UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MARCELO DE GOMENSORO MALHEIROS

# The Mechanochemical Basis of Pattern Formation

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science.

Advisor: Prof. Dr. Marcelo Walter

Porto Alegre
July 2017

**CIP — CATALOGING-IN-PUBLICATION**

Malheiros, Marcelo de Gomensoro

The Mechanochemical Basis of Pattern Formation / Marcelo de Gomensoro Malheiros. – Porto Alegre: PPGC da UFRGS, 2017.

128 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2017. Advisor: Marcelo Walter.

1. Morphogenesis. 2. Reaction-diffusion. 3. Pigmentation patterns. 4. Computer graphics. I. Walter, Marcelo. II. Título.

*Dedicated to Maria Abadia, Pedro, Felipe and Maria Luísa.*

# ACKNOWLEDGMENTS

My very special thanks to Marcelo Walter, for his guidance, patience and confidence.

My heartfelt appreciation to my lovely wife and my happy little kids for their encouragement and invaluable help in making this happen, along all those years of hard work and no vacations, no holidays and few weekends.

Many, many thanks to my mother Maria Lucia for her everlasting enthusiasm and support.

Thanks to my father Carlos Roberto, my sisters Patricia and Sabrina, and my family for their backing.

Thanks to the faculty members of the CG group of INF, at UFRGS, and all the PPGC staff.

Thanks to my colleagues at Univates and all the CETEC staff.

Many thanks for the thesis committee, namely professors Luiz Henrique de Figueiredo (IMPA/RJ), Alessandro de Lima Bicho (FURG), João Luiz Dihl Comba (UFRGS) and Carla Maria Dal Sasso Freitas (UFRGS) for their valuable suggestions for this research.

And also thanks to Przemysław Prusinkiewicz at the University of Calgary and Leah Edelstein-Keshet at the University of British Columbia for their warm reception and much helpful scientific advice.

## ABSTRACT

This doctoral thesis describes a novel model for coupling continuous chemical diffusion and discrete cellular events inside a biologically inspired simulation environment. Our goal is to define and explore a minimalist set of features that are also expressive, enabling the creation of complex 2D patterns using just a few rules. By not being constrained into a static or regular grid, we show that many different phenomena can be simulated, such as traditional reaction-diffusion systems, cellular automata, and pigmentation patterns from living beings. In particular, we demonstrate that adding chemical saturation increases significantly the range of simulated patterns using reaction-diffusion, including patterns not possible before. Our results suggest a possible universal model that can integrate previous pattern formation approaches, providing new ground for experimentation and realistic-looking textures for general use in Computer Graphics.

**Keywords:** Morphogenesis. reaction-diffusion. pigmentation patterns. computer graphics.

**A Base Mecanoquímica da Formação de Padrões**

## RESUMO

Esta tese de doutorado descreve um novo modelo para o acoplamento de difusão química contínua e eventos celulares discretos dentro de um ambiente de simulação biologicamente inspirado. Nosso objetivo é definir e explorar um conjunto minimalista de recursos que também são expressivos, permitindo a criação de padrões 2D complexos usando apenas poucas regras. Por não nos restringirmos a uma grade estática ou regular, mostramos que muitos fenômenos diferentes podem ser simulados, como sistemas tradicionais de reação-difusão, autômatos celulares e padrões de pigmentação de seres vivos. Em particular, demonstramos que a adição de saturação química aumenta significativamente a gama de padrões simulados usando reação-difusão, incluindo padrões que não eram possíveis anteriormente. Nossos resultados sugerem um possível modelo universal que pode integrar abordagens de formação de padrões anteriores, fornecendo nova base para experimentação e texturas de aparência realista para uso geral em Computação Gráfica.

**Palavras-chave:** Morfogênese. reação-difusão. padrões de pigmentação. computação gráfica.

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| BIC | Biologically Inspired Computing |
| CA | Cellular Automata |
| CISC | Complex Instruction Set Computer |
| DBM | Dielectric Breakdown Model |
| DLA | Diffusion-Limited Aggregation |
| GPU | Graphics Processing Unit |
| GUI | Graphical User Interface |
| NNS | Nearest Neighbor Search |
| ODE | Ordinary Differential Equation |
| PDE | Partial Differential Equation |
| RD | Reaction-Diffusion |
| RISC | Reduced Instruction Set Computer |
| RNG | Random Number Generator |
| SIMD | Single Instruction, Multiple Data |

## LIST OF SYMBOLS

| | |
|---|---|
| $a$ | Cell area contribution for domain packing |
| $c$ | Number of chemical reagents |
| $D_R$ | Diffusion rate associated with $R$ |
| $L_R$ | Concentration limit for $R$ |
| $m$ | Number of iterations |
| $n$ | Number of cells |
| $R$ | Reagent concentration for a single chemical |
| $r$ | Number of rules |
| $s$ | Scale factor for reaction part in Turing equations |
| $\Delta t$ | Discrete time step |
| $\nabla^2 R$ | Laplacian operator over $R$ |

# LIST OF FIGURES

# LIST OF TABLES

## LIST OF LISTINGS

CONTENTS

# 1 INTRODUCTION

七転び八起き

*"Fall seven times, stand up eight."*

— Japanese proverb

Since the dawn of Science in the Antiquity, the diversity of life forms has intrigued humankind. The first Greek philosopher who tried to rationalize and categorize living beings was Aristotle. Several early scientists advanced the field dedicated to the study of life, which was first called Biology by Carl Linnaeus in 1763 (HILDEBRAND; GOSLOW, 2006).

Most of the early work in taxonomies was based on visual similarity, grouping species by their external appearance. After the discovery of the DNA by Watson and Crick, the basis for the study of the form changed radically, and taxonomies were re-engineered taking into account mainly genomic similarity, thus following evolutionary relationships between organisms (NULTSCH, 2000). Although much is now known about the genetic code and the biochemical mechanisms that occur inside living organisms, we still understand little about the actual processes that control growth, determine form and define skin pigmentation (BALL, 2001).

The seminal work of Alan Turing provided a diverse, yet controversial direction (TURING, 1952). An abstract chemical reaction, based on a simple two-reagent system and modeled by a pair of differential equations, gave rise to the hypothesis that biological diversity has its root on very simple mechanisms. Turing elegant speculations and pre-computer calculus brought to light the concept of reaction-diffusion (RD) systems (BALL, 2015). Still being a niche research field, reaction-diffusion later led to several landmark studies on Theoretical Biology (GIERER; MEINHARDT, 1972; BARD, 1981; MEINHARDT, 1982; MURRAY, 2003), Computer Graphics (TURK, 1991; FOWLER; MEINHARDT; PRUSINKIEWICZ, 1992), Chemistry (MAINI; PAINTER; CHAU, 1997; PENA; PEREZ-GARCIA, 2001) and Developmental Biology (SHOJI et al., 2003; KONDO; SHIROTA, 2009; MIYAZAWA; OKAMOTO; KONDO, 2010).

In parallel, the early work of Turing (TURING, 1936) and von Neumann (NEUMANN, 1945) led to the theoretical basis of computing, where abstract machines were proven to have the computational power to run arbitrarily complex algorithms. Cellular automata (CA) were born (NEUMANN, 1951), and together with the beginnings of Artificial Intelligence (RUSSELL; NORVIG, 1995), led to the design of very simple yet powerful rule-based systems, which were able to create complex results. Game of Life (GARDNER, 1970), L-systems

(PRUSINKIEWICZ, 1986), fractals (PEITGEN; RICHTER, 1986) and the Chaos Theory (GLEICK, 1987) soon appeared, aided by the advent of personal computing.

Computer Graphics, in particular, have always strived to simulate the visual patterns and geometric structure of living beings, either for the pure aesthetic appeal or the sake of better representation inside animation systems. In fact, reaction-diffusion systems and cellular automata were employed in early CG as base models to generate pigmentation patterns (WITKIN; KASS, 1991). And further pioneering work on forms (FOWLER; PRUSINKIEWICZ; BATTJES, 1992), growth (FLEISCHER et al., 1995) and simulation (WALTER; FOURNIER; MENEVAUX, 2001) made modeling natural phenomena a diverse and multidisciplinary area.

Nowadays, thanks to the enormous computing power of grid systems and graphics processing units (GPUs), the hardware infrastructure can provide support to an ever-growing scale of simulations. Thus, a new field emerged, called Biologically Inspired Computing, where computer systems try to mimic the functioning of living creatures, like the first full-scale simulation of a live bacterium (KARR et al., 2012). At the same time, we witness the ubiquity of computer simulations, providing a bridge between Theoretical Biology, Applied Biology and Computer Graphics, as in the recent works of (CHRISTLEY et al., 2010; GIURUMESCU et al., 2012; KÜCKEN; CHAMPOD, 2013; AMAT et al., 2014).

Still, the defining and control mechanisms of morphogenesis, and in particular, of biological pigmentation patterns, are yet to be fully understood.

## 1.1 Motivation

It should be reinforced the importance of studying morphogenesis, either for Biology or Computer Graphics. For Biology, the understanding of the actual mechanisms that take place during conception, growth and form definition are intrinsically connected to the comprehension of the genetic code and the inner workings of life. This is possibly the most important goal for biological sciences and has implications for all medical and environmental areas (KONDO; MIURA, 2010). For Computer Graphics, the pursuit of techniques for reproducing biological forms is almost as old as the first photorealistic algorithms. Works like plants (DEUSSEN et al., 1998), fur (KAJIYA; KAY, 1989), cactus and flowers (FOWLER; PRUSINKIEWICZ; BATTJES, 1992), pattern venation (RUNIONS et al., 2005), organic forms (DEROSE; KASS; TRUONG, 1998), muscles (TERAN et al., 2003) and skin deformation (GOURRET; THALMANN; THALMANN, 1989), have provided beautiful results. However, most of the techniques were still very specific to the fields they dealt with, being sometimes *ad hoc* and quite complex.

L-systems (PRUSINKIEWICZ et al., 1996) stood out as a different approach, where very simple rules coupled with a growth model mimicked the macroscale arrangements of trunks and leaves of plants. This hinted to an early notion that, in fact, simple biologically inspired rules repeated over and over again could lead to complex, yet realistic models of living beings. In (WALTER, 1998) an expressive biologically plausible cellular model is proposed, where simple mechanisms of adhesion, specialization and subdivision applied on cells gave origin to complex coat patterns for mammals. Then in (WOLFRAM, 2002) there is a long examination of cellular automata, establishing the hypothesis that observed complexity might be just the result of a large scale of simple interactions, both in space and time. Recently, (KONDO; MIURA, 2010) pairs theoretical reaction-diffusion models, computer simulations and Applied Biology to explain and predict actual zebrafish pigmentation patterns.

These works reinforce a connection between simple underlying biological mechanisms, organism growth and the emergence of complex results. Yet, the study of morphogenesis is currently split among different and apparently mutually exclusive theories, as summarized in (STATHOPOULOS; IBER, 2013).

In this thesis, we seek to combine a few of the cited techniques into a minimalist yet *biologically plausible* cellular simulation and explore the process of diffusion-driven pattern formation. Being biologically plausible means reproducing, yet in a simple form, already known and fundamental mechanisms of living cells. In the same spirit, we should avoid adding features that are either too complex or that cannot be mapped into existing processes. We reinforce that such plausibility is pursued both qualitatively (by replicating basic cell functions) and quantitatively (depending on large numbers of cells to produce an emergent pattern).

Therefore, the main hypothesis of our work can be stated as follows:

**Are simple rules and simple interactions,**
**both biologically plausible,**
**running over a long period of time and on a large number of cells,**
**capable of creating realistic pigmentation patterns?**

It is our goal for this research to provide a simple yet powerful model for pattern generation in Computer Graphics. It could be used to define skin pigmentation for procedurally generated virtual creatures, for example, being able to dynamically adapt to different types of bodies. Or it could be applied to more accurately reproduce fur patterns for real animals, like leopards or zebras. We hope that the proposed simulation approach can also contribute to Computing in general, in the sense that large-scale simulations can follow a similar localized (and therefore parallel and scalable) strategy like the one we employ. We also think that this cell simulation model can provide a simple testbed for biologists to play and evaluate new hypotheses about growth and pigmentation mechanisms.

## 1.2 Objectives

The objectives of this work are briefly described as follows.

### 1.2.1 General objective

Our main goal is to present a simple and biologically plausible model that can account for several types of pigmentation patterns.

We focus on a minimalistic approach, avoiding *ad hoc* solutions and keeping both the system state and genetic rules strictly adhering to existing mechanical, chemical and biological processes. In fact, we use simplicity as an Occam's razor[1], to guide the development of the minimal set of features that can still provide a large diversity of patterns. Our fundamental emphasis is on the *mechanochemical* interaction between cells, mainly chemical diffusion through membranes and damped collision, besides simplified cell division.

We have built a parsimonious yet elegant model, trying to mimic the fundamental workings of growing organisms, and expecting the complexity of the resulting patterns to eventually emerge from the large scale of space and time employed.

### 1.2.2 Specific objectives

Here we list specific objectives that are addressed in this work:

- Perform a literature review of early and recent pattern formation studies both in Computer Graphics and in Biology;
- Propose a simulation model that reproduces fundamental mechanical, chemical and biological mechanisms at a cellular level;
- Design rules and states that govern the simulation, providing the rationale for each choice

[1]Occam's razor is a philosophical, but not formally scientific, principle. It states that, among competing hypotheses, the one with the fewest assumptions should be selected. In other words, simpler theories are preferable to more complex ones because they are easier to be tested.

made;

- Describe a particular implementation of our model, amenable to efficient CPU and eventual parallel GPU implementations;
- Design new patterns, inspired by natural pigmentation of several species;
- Validate this approach against real biological patterns and previous results described in the literature;
- Map current limitations and suggest directions yet to explore in pattern formation algorithms.

## 1.3 Text overview

In Chapter 2 we discuss the related work, giving an overview of important literature and theoretical concepts, covering cellular automata, reaction-diffusion and growth models. Next, in Chapter 3 we give an overview of our model, justifying the design choices and presenting the underlying assumptions. In Chapter 4, we detail the current implementation. The basic building blocks for pattern generation are described in Chapter 5, whereas more involved experiments involving growth are presented in Chapter 6. Our model is further validated in Chapter 7, showing the importance of chemical saturation within RD systems and providing many novel and realistic patterns. In Chapter 8 we present a list of our contributions, provide future research directions and then derive our final remarks.

# 2 RELATED WORK

*"If I have seen further it is by standing on the shoulders of giants."*
— Isaac Newton

In this chapter we address important previous work in several fields. There is an extensive literature on cellular automata and growth models which follow the general assumption that simple rules can produce complex results. Some works pursued simplified cell analogies, like the Game of Life, whereas many growth models explicitly followed a macroscopic perspective, like L-systems.

There are also many works that discuss reaction-diffusion systems in different knowledge areas such as Mathematical Modeling, Physics or Theoretical Biology. In Computer Graphics, RD has been pursued as a general pattern generation mechanism in a few landmark studies, but it is in Developmental Biology that most of the current research relating biological patterns and reaction-diffusion is being done.

We understand that such research embodies the pursuit of mathematical models that can generate patterns and forms. The particular works cited in this chapter are thus essential to our model, either to understand the underlying motivation already discussed in Chapter 1, or to establish conceptual (and experimental) support for the choices of biological mechanisms described in Chapter 3.

## 2.1 Cellular automata

Cellular automata are studied by a specific field within Computer Science that is intrinsically related to the Theory of Computing. In fact, its origins are traced to the pioneering works of Alan Turing and John von Neumann.

Turing established the abstract concept of Turing machines in (TURING, 1936), which albeit being very simple both in structure and state models, were proved to perform arbitrary computations. More important, Turing machines established the building blocks of computability and proved that simple machines with simple rules could provide arbitrarily complex results through programming.

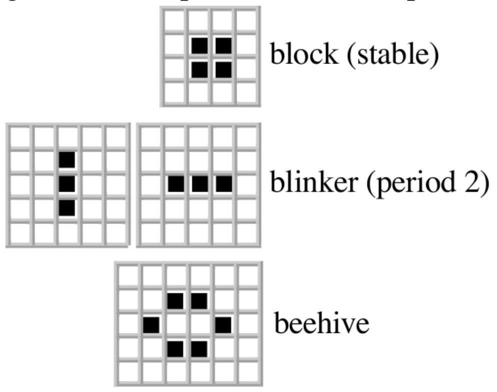
In (NEUMANN, 1945), von Neumann formalized the basic structure of a generic computing hardware, which is known today as the von Neumann architecture. In this architecture the main parts and their functioning were established, like a central processing unit and a memory to store both data and instructions.

Later on, von Neumann theorized what would be the minimum state and instruction code for a functional computing machine, which led to the first cellular automaton (NEUMANN, 1951). The strong similarity between the earlier von Neumann model and this cellular automaton are indeed purposeful. Therefore, this first automaton adopted a matrix-like regular 2D cell arrangement, which is both the simplest mapping to memory cells and also of easiest implementation on computers at that time. It is interesting to note that most cellular automata proposed later followed the simple grid "space", being mostly bidimensional. Several works on cellular automata followed along the years, like the Game of Life, which is significant both historically and as inspiration for our model.

The Game of Life is a simple 2D cellular automaton first presented in (GARDNER, 1970). Its utmost simplicity is based on a biological analogy of "artificial life", which probably added to its popularity. Basically, this automaton runs inside a regular grid of cells that assume only two states: either "dead" or "alive".

The set of rules that provide the outcome from a single cell is also simple and rooted in basic Biology concepts. Each rule takes into account the immediate eight neighbors from a given cell and defines the next iteration of the simulation, or in cellular automata-parlance, *generation*. Figure 2.1 depicts a few simple and non-transitory patterns.

Figure 2.1: Simple Game of Life patterns.



Source: (GARDNER, 1970)

For already established cells, a few rules can be applied. An isolated cell, with no or a single neighbor, is thought to be too exposed to the environment and unable to survive; therefore, it dies. An overly crowded cell, with four or more neighbors, dies of starvation as if there was too much competition for food. On the other side, cells with two or three neighbors can survive and are kept for the next iteration. For empty places in the environment, where there are exactly three neighbors, a new cell is “born”, thus making part of the next generation.

The popularity of Game of Life made it the most studied example of cellular automata. Later research extended it to three and higher dimensions and also to other regular or irregular spaces. There were proposed continuous versions, where the state is no longer binary, but a scalar value (RAFLER, 2011). Several very complex behaviors were observed in Game of Life and sophisticated machines were built within its space. One particular result was that (DURAND; RÓKA, 1999) proved that Game of Life is Turing-complete, and therefore capable of arbitrary computations.

We can use the example of the original Game of Life to establish the overall model for a *simulation loop*, that is, a process which follows the structure sketched by Algorithm 1.

**Algorithm 1** General structure of a simulation loop

```
Initialize world
while $not\,finished$ do
    for each current cell $c$ do
        Examine the neighborhood of $c$
        Determine the next state of $c$ following a matching rule
    Turn the next state into the current state
```

This simulation loop is used to define several key concepts. The *state* is the information associated with a single cell. A *rule* is made of some predefined condition that must be met to trigger some particular action, which then alters the cell state. A *neighborhood* is a formal set of nearby cells that influence the evaluation of a rule condition.

Such cycle represents a design pattern for most computer simulations, including our model. We can identify the current state of the whole system, described by all information associated with a current moment in time of the simulation. In the particular case of Game of Life, it is simply the binary state of the whole 2D world. To provide the next generation, this *current* world is kept unaltered and another world, here called *next*, needs to be constructed. Then each element of the simulation (in this case, a single cell) is processed by the set of rules, and the outcome (in this case, either dead or alive) is placed in the next world. After the processing of all elements, the next world becomes the current world, with marks the transition to a new generation (or equivalently, an *iteration*).

This distinct separation between current and next states/worlds is needed to make the next state deterministic, as the outcome is independent of the evaluation order of cells. The result is that, after the change from next to current state, all changes to the cells seem to occur simultaneously, in an effect equivalent to the age-old back buffer technique from Computer Graphics. This is, in fact, a practical implementation of a discrete time step within a computer system.

As we will discuss later, if the rule evaluation needs only some amount of localized information around each cell, this can lead to the possibility of efficient parallelization. Such locality is particularly useful when relying on modern hardware like GPUs, which are very efficient giving their single instruction, multiple data (SIMD) computation model.

## 2.2 Reaction-diffusion

Reaction-diffusion systems are mathematical models that describe auto-catalytic chemical reactions. They usually involve two reagents, which under Fick's diffusion laws continually diffuse through some solution (GRISKEY, 2005). The reagents are also able to react themselves, changing their concentrations. Such mechanism is commonly formalized through a pair of differential equations that establish the rate of change for the concentrations of each reagent. Let $u$ and $v$ be the concentrations at a single location on a certain domain. Then, the change of concentrations at that location is given by Equations 2.1 and 2.2:

$$\frac{\partial u}{\partial t} = f(u,v) + D_u \nabla^2 u \tag{2.1}$$

$$\frac{\partial v}{\partial t} = g(u,v) + D_v \nabla^2 v \tag{2.2}$$

The right side of each equation can be further split into two components: the *reaction* part and the *diffusion* part. For example, in Equation 2.1, $f(u,v)$ is the reaction part, which produces or consumes some quantity of reagent $u$, depending on the current concentrations of both $u$ and $v$. The $D_u \nabla^2 u$ part is the diffusion term, which takes into account the exchanges of reagent $u$ among nearby locations within the domain. The *diffusion rate* is given by $D_u$, and indicates how fast the diffusion happens. In typical RD systems the reaction functions $f(u,v)$ and $g(u,v)$ and the diffusion rates $D_u$ and $D_v$ are constant for all locations.

The mathematical model for reaction-diffusion was first formalized by Alan Turing in his pioneering paper "The Chemical Basis of Morphogenesis" (TURING, 1952). In this landmark work, one of the last ones he worked on, Turing postulates that the RD mechanism should be investigated as a possible chemical model behind pattern formation in Biology. Although the computation power was very limited at that time, Turing showed that reaction-diffusion systems, running on a regular grid, could generate stable spatial patterns after several iterations.

A important discovery was that the patterns were mostly independent of any previous state that the domain had at the start. In other words, Turing speculated that given a specific set of parameters to a reaction-diffusion system, a pattern of a particular *wavelength*, in his words, would appear spontaneously. The only precondition is that some noise is available in the form of small fluctuations in the equation parameters (or equivalently, concentrations of chemical reagents). That is, in a perfectly homogeneous starting state no such pattern would emerge, but in a real situation, it would indeed.

Turing calls these chemicals *morphogens* (for form generators), making the hypothesis of explicit chemical components that directly controlled the pigmentation patterns in biological organisms.

Despite being a very compelling theoretical model, Turing reaction-diffusion systems did not gain much traction in the biological community until very recently. Kondo and Miura report that search for chemical reagents like $u$ and $v$ did not found fruition, and therefore many applied biologists dismissed Turing hypothesis at first (KONDO; MIURA, 2010). However, along the years particular works validated Turing's original model. In fact, nowadays reaction-diffusion is associated with a broad range of phenomena encompassing auto-catalytic reactions (MIURA; MAINI, 2004), even though such term was coined by Turing in a more restricted sense.

In (GIERER; MEINHARDT, 1972), an independent rediscovery of reaction-diffusion was made by theoretical biologists. Gierer and Meinhardt hypothesized that patterning could be mediated by a short-range *activator* with strong self-enhancing capabilities, coupled to an *inhibitor* of a longer range that suppressed the expansion of the activator in the surrounding areas. The earlier work of Turing was not known by them at that time. In 1981, Bard examined the reaction model of Gierer and Meinhardt and proposed a specific mathematical model as the explanation for spot and stripe patterns on mammals (BARD, 1981). In following works, Gierer and Meinhardt formalized two possible realizations of self-organizing systems with two components: the activator-inhibitor system and the substrate-depletion model (MURRAY, 2003).

Kondo and Miura (2010) argue that the most valuable contribution by Turing was not the hypothesis of the existence of chemical morphogens itself, but the overlaying mechanism. In fact, the most striking insight from Turing was the argument that a mechanism responsible for destroying patterns, diffusion, could actually be part of a system for creating them. Therefore, the combination of *short-range activation and long-range inhibition* can lead to stable patterns very similar to many biological examples. In other words, a very simple set of differential equations can model and explain the dynamic behavior of a complex system. Moreover, even if the underlying biological mechanism is not yet understood, the model given by RD systems is robust and able to provide forecasts for expected results (KONDO; MIURA, 2010).

Several studies of applied and theoretical Biology tried to correlate reaction-diffusion models to patterns of living organisms. For example, the works of (SHOJI et al., 2003; KONDO; SHIROTA, 2009; MIYAZAWA; OKAMOTO; KONDO, 2010; WATANABE; KONDO, 2012; GAFFNEY; LEE, 2013; GARVIE; TRENCHEA, 2014). The work of (VOLKENING; SANDSTEDE, 2015) employs a simulation for zebrafish skin patterns, where individual cells are modeled and interact to generate pigmentation. This model strongly depends on cell migration but otherwise have mechanics similar to our own. However, the focus is on the reproduction of particular biochemical phenomena that matches the real experiments made. Another interesting work correlates the coloration of living lizard scales to a discretized RD system, which is also a particular case of cellular automaton (MANUKYAN et al., 2017).

Reaction-diffusion systems were introduced to the Computer Graphics community by the works of Turk (TURK, 1991) and Witkin and Kass (WITKIN; KASS, 1991). Although published at the same time, each paper focused on different applications and control mechanisms for RD systems. While Turk was concerned with the creation of new textures on 3D surfaces, Witkin and Kass tried to synthesize patterns that followed the geometric characteristic of a form, employing for that local control over pattern scale and anisotropy.

Pearson presented another widely known reaction-diffusion system, the Gray-Scott model (PEARSON, 1993). This set of equations has proved to be useful for the creation of synthetic patterns, providing robust numerical control, a wide range of different patterns and complex dynamic behavior, although with limited empirical evidence of being observed in Nature. In (SANDERSON et al., 2006), the effects of anisotropic diffusion and parameter modulation were further explored to control the direction of stripes and orientation of spot patterns. They were able to synthesize patterns very similar to the ones present in Puffer Fish species. In (MCGRAW, 2008) the Gray-Scott model was further generalized.

Meinhardt's early work on shell pattern formation (MEINHARDT; KLINGLER, 1987) led to a well-known paper on shell modeling (FOWLER; MEINHARDT; PRUSINKIEWICZ, 1992). Later on, a full book on algorithmic models for sea shells was released (MEINHARDT, 2009), with many visually appealing results.

## 2.3 Growth models

All adult biological organisms, either individual or living colonies, are the result of a growth process during its life, starting as a simpler embryo or seed. Therefore, there is a particular line of research that explicitly models organism growth. The study of biology has shown that form and growth are intrinsically connected (HILDEBRAND; GOSLOW, 2006), and these models make it possible to simplify the assumptions on different aspects of organism growth.

Several models have been proposed over the years, with varying goals and assumptions about the underlying biological mechanisms to be simulated. To cite a few: (AGARWAL, 1994; ABELSON et al., 2000; WALTER; FOURNIER; MENEVAUX, 2001; RUNIONS et al., 2005; CICKOVSKI et al., 2005; CAICEDO-CARVAJAL; SHINBROT, 2008; PORTER; MCCORMACK, 2010). For the sake of brevity, we will only discuss two particular approaches that inspired our model: L-systems and Fleischer's growth model.

### 2.3.1 L-systems

One of the earlier and simpler growth models is called L-systems. It is named after the observations from Lindenmayer in (LINDENMAYER, 1968), where the author tries to formalize the organization of cells in a particular type of algae. His model was based on the analogy between the parallel growth of an individual and the expansion of terms in a regular grammar.

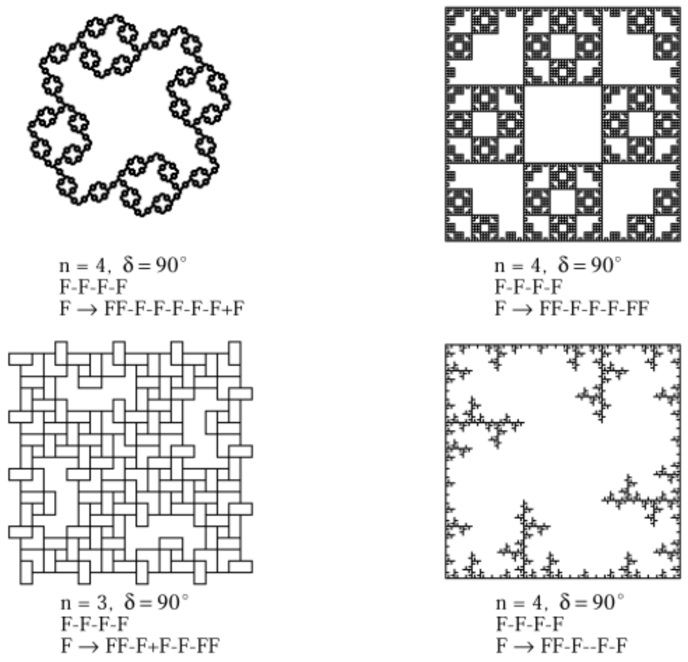
Given a formal regular grammar comprised of an initial state, terminal symbols, non-terminal symbols and transition rules, the abstraction of a growing organism can be built upon the repeated expansion of the initial symbols by iterated application of such rules. For example, let $S$ be the initial symbol (analogous to a seed or to the initial cell). We have $a$ and $b$ as terminal symbols and $a \rightarrow ab$ and $b \rightarrow bb$ as rules. The expansion progresses in several iterations. In a single iteration, all non-terminal symbols are expanded just once, which is called parallel expansion. Therefore, from the initial cell, we would have iterations like $S$, $ab$, $aaba$, $abababa$ and so on.

The rationale for the parallel expansion is the notion that the single cells $a$ and $b$ will grow in parallel by the same amount during an iteration, creating each one the set of cells specified by the right side of a transition rule. That is, after one iteration, a single $a$ would generate $aba$, for all previous instances of $a$.

Lindenmayer proposed this model to explain the specific cellular organization of algae *Anabaena catenula*, which resulted in a unidimensional sequence of cells. Soon that approach was extended into two and three dimensions, providing a much more powerful mechanism for describing complex shapes (PRUSINKIEWICZ, 1986). Then, the initial results were equivalent to simple plants, establishing one of the earliest procedural generation systems (PRUSINKIEWICZ; LINDENMAYER; HANAN, 1988).

Whereas the regular grammar expansion mechanism was kept, thus still consisting of a single sequence of symbols after expansion, what Prusinkiewicz introduced was a particular interpretation of the symbols. It was proposed that each terminal symbol be interpreted as instructions to a two- (and then three-) dimensional drawing language, where some symbols instructed a virtual pen to draw straight lines, whereas others indicated relative changes in direction. For example, in the following 2D L-system, we have $F$ as a command to draw a straight line in the current direction, and $+$ and $-$ as commands to change the current pointing direction to the left and to the right, respectively, without drawing. These schemes follow the well-known "turtle" graphics concept from Logo and provide the basic building blocks for more complex shapes (PAPERT, 1972). Some closed curves are shown in Figure 2.2.

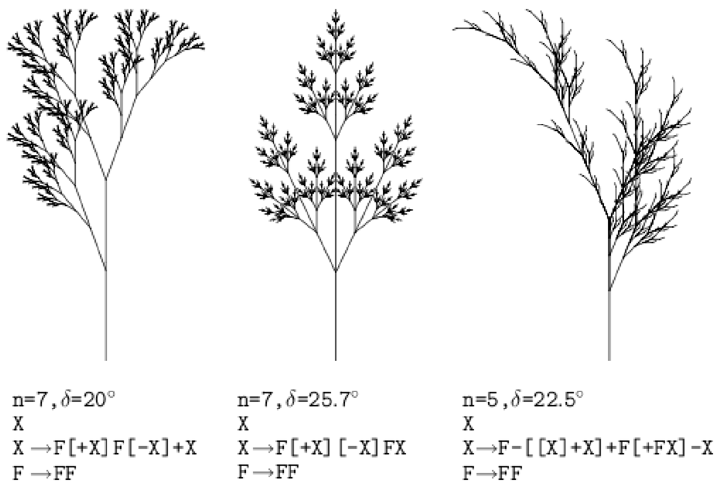
Figure 2.2: Examples of complex closed curves generated by L-systems.



Source: (PRUSINKIEWICZ, 1986)

Although the simplicity of regular grammars also implies their limitation, the addition of a stack mechanism into the drawing (or sequence interpretation) code makes the creation of branches possible. The idea is to explicitly create branches through the use of "state push" and "state pop" operations, which would provide independent growth in branching patterns, exactly as the branches of a living plant. Such systems are called Bracketed L-Systems, and three examples are shown in Figure 2.3.

Figure 2.3: Branching structures generated by L-systems.



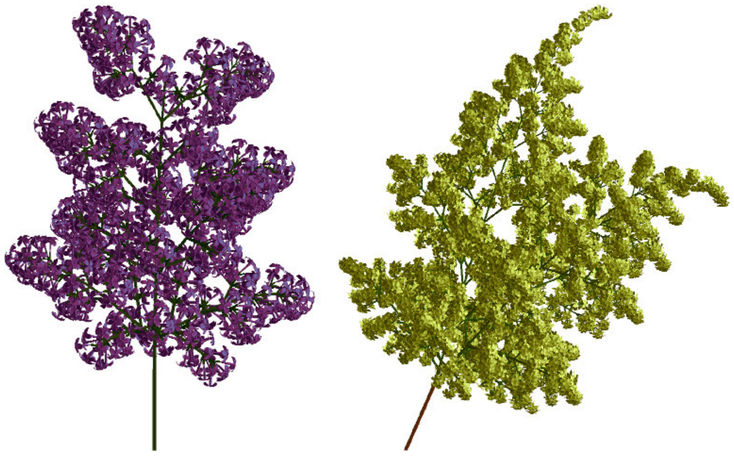
Source: (PRUSINKIEWICZ, 1986)

Further work on L-systems focused on providing more powerful grammars, which in turn made possible more complex and realistic models, as shown in Figure 2.4. In (PRUSINKIEWICZ; HAMMEL; MJOLSNESS, 1993), an L-system was also used to model the leaves and petals of flowers in vegetation. In (PRUSINKIEWICZ, 1998) was proposed the use of context-dependent grammars, enabling the application of rules that involved the local evaluation of symbols, therefore being able to model distinct phases of the development of a plant, where a stochastic growth model was also added to provide diversity. In (PRUSINKIEWICZ; LINDENMAYER, 2012) the explicit modeling of plant organelles was set, including the timed activation of buds and the formation of flowers, still following a formal grammar model.

Although the visual results are striking, the formality of the grammar can be seen as its major drawback. As the expansion is done by the iterative substitution of symbols, the exact design of rules is quite difficult when trying to achieve a particular target form. Also, for simple grammars the self-similarity shape is easily achieved, but in real organisms, parts of the self-similarity are lost due to various reasons, like external factors, a difference in growth speed or even damage. Therefore, simpler L-systems representations lack the context dependence and locality of interaction. On the other side, extended L-systems like (JIRASEK; PRUSINKIEWICZ; MOULIA, 2000) provide a much more complex rule language, which may be difficult to tune to specific results. A later work described the use of high-level user interface elements as guides for the automatic construction of L-systems (ANASTACIO;

PRUSINKIEWICZ; SOUSA, 2008). A previous work of our own has used L-systems to provide simple yet visually pleasing tree models for real-time applications (MALHEIROS; WALTER, 2011).

Figure 2.4: Complex plants modeled by later L-systems.



Source: (PRUSINKIEWICZ et al., 2001)

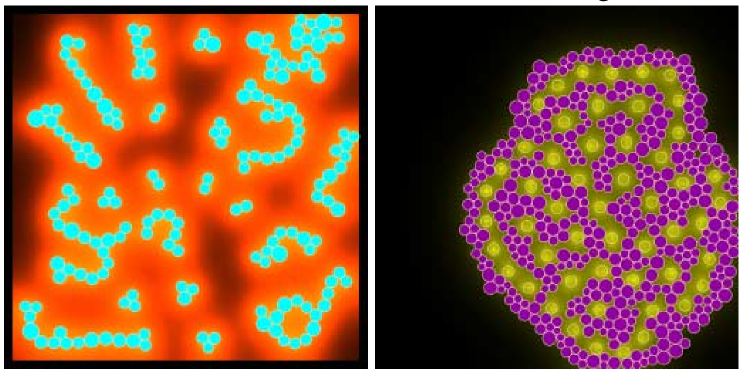
As L-systems provide a form generation framework, they give important insights into the simulation of biological shapes. They also reinforce our perception that the underlying rules for the definition of natural forms and patterns can be described by simple yet iterative computer models.

### 2.3.2 Fleischer's growth model

Although many cell simulation models have been proposed, most of them are either simplistic or highly specialized. In (FLEISCHER; BARR, 1994) it was proposed a general model, strongly inspired by biological mechanisms. In this model there was a concerted effort to simulate not only the physical contact between cells but also their chemical and electrical communication, along with interaction with the surrounding environment and also the explicit support for cell division. Further work on this model was described in (FLEISCHER, 1996), where there was a focus on mechanical collisions and the creation of segmented structures.

What is particularly interesting is that, as far we were able to research, Fleischer proposed the first complete biological cellular model which combined mechanical, chemical, electrical and biological mechanisms. Most previous simulations models focused on just one or two such aspects. The Figure 2.5 illustrates two results.

Figure 2.5: Cell simulation results for Fleischer's growth model.



Source: (FLEISCHER et al., 1995)

Careful analysis of Fleischer's model was invaluable to confirm some design choices for our model, but also to present some pitfalls to avoid. For example, his model employed a set of ordinary differential equations (ODEs) as the response to changes in the state of a cell, being expressed in a high-level language reminiscent of C. We perceived such approach as being too general, unnecessarily increasing the complexity of the design of experiments. It also moved away from biological plausibility when a sinus function, for example, could be added as the response to a given stimulus. Fleischer itself admitted that "writing cell programs can be difficult" (FLEISCHER et al., 1995). In our work we try to reduce the designer burden by keeping a strict biological analogy for all its mechanisms, and also by limiting the choices available.

Interestingly, Fleischer's model incorporated diffusing chemicals and the possibility of running a reaction-diffusion system. Such simulation, however, ran on a standard regular grid that both covered cells and the extracellular space. While (TURK, 1991) was the first to simulate an RD system on the faces of a mesh, (COORE; NAGPAL, 1998) later showed that the same simulation could run on unconstrained and free-moving cells.

## 2.4 Summary

In this chapter, we presented a brief overview of literature that both gave the inspiration to our simulation model and provided invaluable techniques that could be built upon.

By studying many distinct previous approaches, we were able to perceive their limitations. Cellular automata are powerful constructs, but their rigid tying into regular lattices restricts their possibilities. Formal grammars are very useful at a macroscale level, matching the overall form of plants, for example, but they usually lack local control and predictability, which are essential to continuously-varying patterns. Moreover, overly broad description languages may make the reproduction of a specific behavior too difficult, falling into the "curse of dimensionality". That is, by having so many possibilities the exploration of all parameters or enumeration of all hypotheses is impractical.

We have thus identified the opportunity for defining and exploring a biologically inspired cell simulation that pairs at the same level reaction-diffusion systems and growth rules, balancing complexity and generality.

# 3 MODEL

*"Beware of the Turing tarpit in which everything is possible but nothing of interest is easy."*
— Alan Perlis

In this chapter, we discuss the rationale for our model. Here we focus on a more abstract definition, justifying the underlying choices based on previous works of the literature and specific design directions of our own.

Therefore, we address the model from a more theoretical view; the actual implementation details are deferred to Chapter 4. In this way, we can establish parallels with biological concepts to justify particular decisions. The technical details of the simulation, like the choice of RD equations, nearest neighbor search, numerical issues, physics models, parallelization strategies and practical simplifications are therefore drawn in the next chapter.

## 3.1 Overview

By design, our model is both simple and biologically plausible. We have focused on a minimalist approach, avoiding *ad hoc* solutions and employing mechanisms that strictly follow occurring chemical, mechanical and biological phenomena. In fact, we use the Occam's razor principle to guide the development of a reduced set of features that can still provide a large diversity of patterns. Our primary emphasis is on the *mechanochemical* interaction between simulated biological cells, mainly chemical diffusion through membranes and damped collision, besides controlled cell division.

We are particularly interested in pigmentation patterns which are mostly influenced by cell interaction on the organism skin. Thus, our model works at a mesoscale level, abstracting the living cells as indivisible entities. Therefore, we opted to simulate cells as compact units with an internal homogeneous state. A set of rules defines the genetic programming of a cell, triggered only by simple conditions, and which result in just simple cell events. All simulated cells bear the same genetic code, or in other words, are governed by the same set of rules. Cells can differ regarding their internal state, which will then trigger different events at different times during the simulation.

Although most growth processes happen in three dimensions inside living organisms, our focus in pigmentations patterns justifies the constraint of having cells on a bidimensional domain, analogous to the surface of the skin. Furthermore, a very important design guideline is that cell interaction must be strictly local. That is, a cell only interacts with nearby cells. Again, we follow the biological inspiration of cell membranes touching. In fact, as we will detail later, interaction is restricted only to the mechanical collision between cells and chemical diffusion through membranes. Such locality has the added benefit of enabling an easily parallelizable implementation, which can run entirely on a GPU.

### 3.1.1 Scale

Biology studies living organisms and their complex interactions at many distinct scales. Major scales involve meters, hundreds of meters or even a planet in its entirety. This is typically addressed by the study of Ecology, and this area is considered to be already quite mature (NULTSCH, 2000). In a lower scale, we can think of symbiotic behavior among living creatures, which still deals with distinct individuals and their local environment (GULLAN, 2012).

Inside a single organism, we have a very complex set of organ systems: integumentary, skeletal, nervous, circulatory, endocrine, muscular, respiratory, excretory, reproductive, digestive and immune. We may say that, relative to a single living organism, this is the *macroscale* level. Each system is typically very complex in its structure and interrelation with the other organ systems.

A single system is still subdivided into several organs and organic structures, each one comprised of thousands, millions or even billions of cells. We may then call this the *mesoscale* level, where cells can be seen as cohesive units, functioning together to provide the building blocks and metabolic mechanisms for a single organ.

Of course, a particular cell can be analyzed in detail, studying its internal organization of nucleus, membrane, and other subcellular components. This involves the further subdivision of structural elements and chemical components, giving rise to the *microscale*, where the proteins and signaling mechanisms have focus. Still, a *nanoscale* level still lies beneath, where the actual structure of organic molecules can be studied and their complex reactions mapped.

As we are particularly interested in pigmentation patterns, which are mostly influenced by cell interaction on the organism skin, it is natural to focus our model into the mesoscale level, thus abstracting the living cells as indivisible entities.

### 3.1.2 Genetics

Each living cell has genetic material in its nucleus, comprising of long strings of deoxyribonucleic acid (DNA). Thus, the blueprints to all organism processes are coded in the DNA, which is typically the same for each cell of an individual. Several earlier works simulating growth systems either explicitly or implicitly defined genetic information. Some used hard-coded rules (AGARWAL, 1994), others formal grammars (PRUSINKIEWICZ et al., 1996), mathematical formulas (RUNIONS et al., 2005) or even interpreted source code from a standard programming language (FLEISCHER; BARR, 1994).

Although powerful, such high-level approaches may appear conceptually disconnected from the actual biological entities they strive to simulate. And there is always the risk of excessive complexity, driving away simpler and, perhaps, more natural explanations for natural phenomena.

As we strive for utmost simplicity of our model, we defined as genetic information only a set of basic rules — rules that are biologically plausible at a mesoscale cellular level. Therefore, it would not make sense to evaluate a sinus function for some scalar value, but it does seem to be likely to initiate a cell division if the concentration of some chemical reagent rises above some predefined threshold. Rules are triggered by simple conditions, and which would result in just simple cell events.

Following the biological analogy, all simulated cells should bear the exact same genetic code, or in other words, be bound by the same set of rules. Cells can differ regarding their internal state, which will then trigger different events at different times during the simulation. What comprises the state of a single cell is discussed in Section 3.2. We detail the set of possible events in Section 3.3.

We can establish an analogy between our minimalistic approach to rules and the design of a CPU instruction set. Whereas more complicated instructions, like in a complex instruction set computer (CISC) architecture, lead to more compact executable code, this comes at the expense of more complex design for the CPU circuits themselves. On the other hand, a RISC architecture makes possible a more streamlined chip design, although creating longer programs. It is now established in Computer Science that either CISC or reduced instruction set computer (RISC) approaches lead to equivalently powerful processors, thus not affecting the expressibility of programs that run on them (TANENBAUM, 2001).

The choice for a basic set of rules does not directly imply that the patterns being able to be simulated are limited. In fact, we expect exactly the opposite: by keeping the model simple and tied to already known biological concepts, we can drive our experimentation by expected natural behavior, and focus on the smallest set of rules first. Moreover, the set of rules stays the same during all of the simulation. Eventually, the possibility of some kind of mutation could be evaluated, changing an arbitrary rule at some point in time, but this is not the current focus of our work.

From now on we will use the term *experiment* to describe a particular setup to be simulated. An experiment specifies the initial state of all cells and the rules which will be evaluated on them after the simulation starts.

### 3.1.3 Environment

We choose a 2D domain mainly for its simplicity, following the analogy of the development of cells on the surface of the skin. Therefore we could factor out all the complexity which would be needed in keeping cells over an evolving 3D surface, a topic which has already been studied extensively in the literature (TURK, 1991; WALTER; FOURNIER; MENEVAUX, 2001; CARVALHO; ANDRADE; VELHO, 2012).

It can be argued that surface curvature can affect the creation of patterns, but we opted to have a simpler environment; curvature could still be eventually taken into account by activating different rules based on localized concentrations of some reagent, for example.

Another important point is that the main focus of experimentation is on dense arrangements of cells, like a part of tissue. Despite our model being able to create elongated and branching forms, as reported in Chapter 6, most of the biological pattern formation occurs on a continuous and dense patch of cells.

We devised the extracellular space as empty, with no cells but with plenty of space for growth. Hard limits could be imposed to the cell simulation if needed, constraining the growth to a fixed bounding box to represent cell compression. The only allowed explicit geometric placement happens before the actual simulation, when the initial cells are defined. After the simulation starts, there is no direct control of absolute cell positioning. They only move as result of a collision with other cells, which in its turn is triggered by cell division, as described in Section 3.4.

## 3.2 Cell state

A cell within the simulation is modeled as a homogeneous entity, represented geometrically as a circle with unit radius. The numerical attributes for a single cell are summarized in Table 3.1.

Table 3.1: Attributes of a single cell.

| Attribute | Description | Type |
|---|---|---|
| position | $x$ and $y$ coordinates | floating point |
| $R$ | concentration for reagent $R$ | floating point |
| $D_R$ | diffusion rate for reagent $R$ | floating point |
| polarity | unit vector, with $i$ and $j$ components | floating point |
| birth | iteration number the cell was born at | integer |
| fixed | marks the cell as immovable | boolean |
| neighbors | number of nearby cells | integer |

Source: own work.

Being part of a 2D domain, each cell has a pair of real-valued coordinates. Each cell is thought to be round, containing some intracellular fluid and protected from the exterior by a cell membrane. The cell membrane is thought of having varying porosity, thus being permeable or not to the diffusion of one or more reagents. The diffusion rate (or permeability) $D_R$ for each reagent $R$ is individualized and per cell. Therefore, each cell has a pair of non-negative real values for the concentration and diffusion rate associated to each chemical reagent. While reagent concentrations change as the effect of several mechanisms, the diffusion rates can only be changed explicitly by rules activated on the same cell.

Our model neither explicitly establishes different cell types nor discrete cell states. Thus, any functional specialization by cells can only be the result of different reagent concentrations, which in effect can activate different rules. The actual meaning for any reagent is up to the design of the rules; no assumptions are made on reagents. In other words, all reagents are processed in the exact same way by the simulation. This design decision was made to keep the model generic, yet close to a simplified real cell.

We borrow the term *prepattern* from Embryology, to represent the initial spatial and chemical configuration of cells. Although our model enables extensive control over the definition of the prepattern, after the simulation starts each cell state is only affected by the evaluation of the shared set of rules and the limited interactions with nearby cells.

In fact, the prepattern is actually very important to achieve a specific result, but many spatial forms can be achieved by growth in earlier steps. A more radical approach to pattern generation would always start at a single cell, and provide all the rules to build the desired form and pattern from there.

Albeit cells are modeled here as perfect circles, biological cells may have *polarity*. This term comes from Biology and does not involve electrical charges. It means that a cell gains a preferred direction when executing cellular mechanisms (GOEHRING; GRILL, 2013). We represent this orientation by a unit vector. Polarity is a critical feature of our model, as we have a way to establish and maintain directionality during either division or anisotropic diffusion. Cell polarity can be defined explicitly in the prepattern, as a direct result of cell division or by the activation of a rule that aligns the cell's polarity to the concentration gradient of a given chemical reagent. It should be stressed that each cell has only one polarity direction at a given time, so we follow the analogy to the biological case.

An integer value is present on each cell to store at which iteration it was born. Cells defined in the prepattern have birth time set at zero; all others have birth time equal or greater than one.

A simple boolean value is defined for each cell which prevents it from moving after collisions. This is provided to fix cells in the prepattern so they can be held part of a large structure, like a bone or hard tissue. This is a simple feature that makes possible the exploration of growth regions that are not displaced by the creation of new cells.

Finally, each cell has a counter for how many neighbors it has. Neighbors are the cells whose membranes are closest, and which therefore can interact with this particular cell. This counter is dynamically updated at each simulation iteration and gives a simple mechanism to evaluate whether a cell is under strong compression or whether it is on a tissue border. It is also used to prevent excessive division. No information is kept about which are the nearby cells, or where they are located at.

## 3.3 Cell events

Here we give an overview of the discrete cell events that can happen within the simulation. Any such event is triggered when a specific condition is met, as described by the set of rules that comprises the genome for a particular experiment.

We have opted to implement a reduced set of biologically-inspired events, for the sake of simplicity of simulation. The actual choice of events ranges from the simplest, chemical related ones, to the more common biological mechanisms. We tried to cut down any event too specific or that would stray from plausibility at such mesoscale environment. We have thus tried to balance simplicity and expressibility, although knowing that more complex emergent behavior could require a higher number of reagents and rules.

Each rule has a *condition* and a corresponding *action*. The conditions are restricted to testing for either the iteration counter or simple comparisons between current cell attributes and real values. When a condition is met, the action is immediately performed, which results in the occurrence of one of the cell events described in Table 3.2. Actions only affect the next state of the cell for which they were activated; the current state is kept unaltered during an iteration to ensure that the results are not dependent on the order of cell evaluation.

Table 3.2: Cell events within the simulation.

| Action | Cell event |
|---|---|
| change | produce or consume reagent $R$ |
| change | increase or decrease the diffusion rate of reagent $R$ |
| react | modify concentration according to an RD equation |
| divide | perform instantaneous cell division |
| polarize | change polarization along a concentration gradient |

Source: own work.

The simplest event is the production or consumption of a chemical reagent within the cell. By design, an action cannot set the internal concentration of a reagent to a specific value. It is only possible to increase it or decrease it by a given amount. This decision was made based on two arguments. First, although we are simulating discrete cells with discrete time steps within the simulation, we are modeling real processes where concentrations change continuously. So it makes sense to either produce more or consume part of the chemical diluted inside the cell. Second, as reagent concentration is constantly changing through chemical diffusion, such change must be added to the production or consumption of a given reagent.

There is no restriction on the amount of reagent either produced or consumed in a single action. However, chemical concentration cannot be negative and can be bounded above by a certain saturation amount. Such saturation level is optional, being globally defined for each reagent, and cannot be changed during the simulation. As discussed in Chapter 7, we have discovered that chemical saturation plays a crucial role in the formation of natural patterns, which is a phenomenon not explored at all in previous works.

Besides the change in a reagent concentration, actions can also increase or decrease for some amount the current diffusion rate for a specific reagent.

A critical part of our model is the reaction, which is the equivalent part of a reaction-diffusion system. When a rule activates reaction, two or more reagents inside the current cell are modified through the application of a specific set of RD equations. As said before, whereas diffusion happens continuously, the reaction part is discrete in time and thus must be explicitly initiated. Here we are proposing a decoupling that is not usual in RD systems, but which actually makes sense in living organisms: most chemical reactions are controlled processes and thus do not occur at all times.

Another event is cell division. When triggered, mitosis occurs and a new cell is instantaneously created. It is identical to its parent, except for two attributes: birth and polarity. Its birth is set to the next iteration, whereas the polarity for a child cell is defined relative to its parent's polarity, following a given angle as parameter for this action. A division event is usually tied to a probabilistic condition, where the chance of happening is explicitly set.

We have also defined a simple mechanism to prevent excessive division. Biological tissues have complex growth controls to regulate division and thus maintain its integrity. We have opted to employ a simple approximation for cell compression that prevents division above a certain limit: a global parameter can be used to prevent cells from dividing if they have more than a given number of close neighbors.

It is interesting to note that no explicit mechanism for cell death is needed in our model, as the combination of a uniform chance of mitosis and a maximum division limit seems to be enough so far to provide consistent growth of tissue-like structures.

The current model does not permit a cell to explicitly modify its own orientation. It seems that such operation would be biologically implausible, as cells have no knowledge of their surrounding environment. Moreover, polarity in Biology is normally motivated by external means, either when a division occurs or when cells are affected by chemical gradients (GOEHRING; GRILL, 2013).

We thus have a "polarize" event that orients a cell towards the higher concentration of a given chemical reagent. As diffusion is a process that happens for all chemical reagents, their concentrations are usually continuous. Therefore, we can derive a local chemical gradient by only examining the concentrations of the closest cells, providing a simple but powerful control mechanism, where many cells can be consistently oriented by setting up a single cell producing a chemical reagent. Such approach makes possible the exploration of several hypotheses related to the theory of morphogen gradients (STATHOPOULOS; IBER, 2013).

## 3.4 Cell interaction

This section describes the main interaction mechanisms between cells: chemical diffusion through membranes (either isotropic or anisotropic) and mechanical collision.

We have defined that cell interaction is strictly local during the simulation since it is a biological fact that most signaling happens when cells are touching their membranes. Thus, by using local interactions, the state change for a given cell depends only on its nearest neighbors. This assumption both simplifies implementation and provides more predictability, without loss of expressivity. Furthermore, such locality also makes possible the simultaneous and independent evaluation of cells, thus being amenable to parallel processing on a GPU.

It follows directly from these choices that there is no global control after the simulation begins. Cells only move when subject to a collision, caused by cell division. The reagent concentrations only change when subject to chemical diffusion, synthesis, or consumption. The only global mechanism that is explicitly allowed is the passage of time, in the form of activation of rules dependent on the iteration count since the simulation starts. This is important to achieve fine control of the moment actions occur. In fact, even the passage of time could be implemented as a chemical decay within each cell, from the same initial concentration of all cells. We opted to tie it explicitly to the iteration count only as a matter of convenience for the experimentation.

We have consciously avoided individualized cell rules. Thus it is not possible to have a rule that affects only the cell with id 1428, for example. This would go against the generality principle of a genome. Nevertheless, such fine control is still possible with our current design: that cell can be initialized with a distinct reagent concentration in the prepattern, which later triggers a specific rule during the simulation.

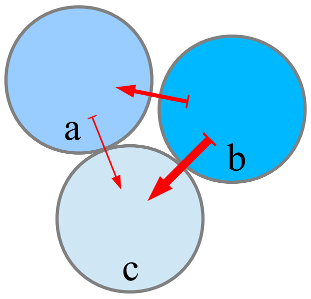
Furthermore, we have defined that there is no possibility of direct user intervention after the simulation has started. In other words, cells are subject only to the shared rules, their internal state and their immediate neighbors.

### 3.4.1 Isotropic diffusion

A fundamental premise of our model is to decouple reaction from diffusion. Diffusion is seen as a simple chemical process, where a given substance spreads within some substrate. Therefore, diffusion occurs continuously among cells, at every simulation step. On the other hand, reaction is seen as a per-cell condition-triggered event, and therefore must be explicitly initiated by some rule.

Diffusion is usually *isotropic*, that is, it operates the same way in all directions. This is the most common situation, and isotropy is assumed for most RD models described in the literature (MURRAY, 2003). Isotropic diffusion occurs through the cell membranes, being controlled by the diffusion rate of each cell involved in the process. As the actual diffusion is computed as a summation of pairwise cell contributions, its rate between two cells in contact is a combination of their individual diffusion rates (detailed in Section 4.3). If one of such cells is impermeable, no diffusion will occur between them. Figure 3.1 illustrates isotropic diffusion between three cells. Note that cells (a) and (b) need not to have their membranes touching to perform diffusion; they just need to be close enough. Such proximity criterion is detailed in Section 4.2.

Figure 3.1: Isotropic diffusion: *b* has highest concentration and *c*, lowest. The amount of diffused reagent is indicated by the width of the red arrows.



Source: own work.

Being empty, the extracellular space does not allow diffusion, which is equivalent to a zero-flux or Neumann boundary condition. Therefore, each group of nearly placed cells can be thought of a closed system, functioning independently from other connected groups of cells within the same simulation.

Our approach is quite different from the direction taken by (FLEISCHER, 1996), where the chemical environment is modeled by a regular grid where cells move over. In Fleischer's model, the chemical concentrations are therefore tied to specific positions in space, and not to the freely moving cells. We believe Fleischer took this route out of simplicity, reproducing the typical RD implementations available at that time. Moreover, in his model the cells are affected by the chemical field, but no examples were shown of them being able to modify it. However, a very interesting analogy can be established with the Lagrangian and Eulerian specifications of a flow field (VERSTEEG; MALALASEKERA, 2007). The Lagrangian specification follows each portion of fluid as it moves through space and time, mimicking cells in our model with their internal chemical concentrations. Then, Fleischer's model may be viewed as a simplification of the Eulerian specification, where chemical concentrations are mapped into discrete parts of the domain, being sampled and possibly modified by the moving cells. Of course, a non-empty extracellular space could eventually be simulated in our model by adding pseudo "fluid" cells, but we think that such extra complexity does not give us foreseeable benefits.

It could be argued that employing just diffusion is a limiting choice and that such decision would prevent more complex phenomena to be modeled, thus restricting the achievable patterns. For example, we are not explicitly addressing inter-cell signaling, which is also another standard biological cell mechanism (MURRAY, 2002).

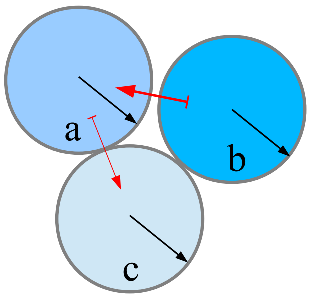
We justify our model based on the premise that although the biological pattern generation mechanisms in the literature are quite varied on their internal processes, many recent works follow the outcome of the simpler conceptual model of "short-range activation and long-range inhibition" (MEINHARDT, 1982). That is, even if the chemical and biological mechanisms are intrinsically different, the net result is very similar: the establishment of Turing patterns and stationary waves (KONDO; MIURA, 2010). So it is valuable to employ a generic yet theoretical reaction-diffusion model as the basis for pattern generation in Computer Graphics, both for its simplicity and the breadth of previous studies.

Another important consideration is that diffusion and direct signaling models can be seen as numerically equivalent (HAMMER, 1998). This means that some (but not all) signaling mechanisms can be mapped into an equivalent chemical diffusion process. So part of the expressibility of a general model of signaling is already possible by only employing diffusion. Moreover, signaling networks are in fact quite complex, comprising several types of biochemical mechanisms, involving gap-junctions, receptor proteins and signaling molecules. Modeling such mechanisms would delve into the microscale level, drifting us away from the simplicity pursued in this work.

### 3.4.2 Anisotropic diffusion

A very important feature of our model is the activation of *anisotropic* diffusion for one or more reagents. That is, for a given reagent its diffusion can be constrained to happen primarily with neighbors on the same direction given by the cell polarity. Therefore, the actual amount transferred between cells is modulated by their relative position and orientation, as shown in Figure 3.2.

Figure 3.2: Anisotropic diffusion: cell polarity is indicated by the black arrows and amount of diffused chemical is indicated by the width of the red arrows.



Source: own work.

The control of diffusion through anisotropy makes possible the creation of locally oriented patterns, as discussed in Chapter 5. In fact, anisotropy has been used for directionality control in several previous works involving reaction-diffusion systems (WITKIN; KASS, 1991; SANDERSON et al., 2006; KIM; LIN, 2007; CHI; LIU; HSU, 2016), but also in other texture synthesis techniques (ITOH; MIYATA; SHIMADA, 2003). Only very recently a paper was presented also coupling gradients and anisotropy (HISCOCK; MEGASON, 2015), albeit based on the Swift-Hohenberg equation and incorporating those features into a single partial differential equation (PDE).

It should be noted that in all those works a fixed and regular cell arrangement was used. As far as we could research, our model is the first to employ anisotropy on an irregular arrangement of cells.

### 3.4.3 Collision

Following the biological analogy, we suppose that our cells live within a limited environment. They are either part of a tissue or immersed in a viscous extracellular fluid (which is not simulated). Therefore, their movement is highly constrained.

This led to the use of damped kinetics for cells. We have opted to not model forces or velocities, as this type of dynamics would not be useful in a dense arrangement of cells. As already known in Biology, there are types of movements done by unicellular organisms, usually called chemotaxis (MURRAY, 2002). But even such movements are slow and self-impulsed. So there is almost no linear momentum to conserve in a typical pattern simulation.

Cells, however, do collide after a cell division. We have opted to use a simple repulsion scheme, which pushes cells back proportionally to the amount of their overlapping. This translates into “soft” collisions, where overlapping cells are gradually moved far from each other. The motion stops only when their membranes are touching. More details are given in Section 4.4.

## 3.5 Summary

In this chapter we have presented an overview of the proposed simulation model, focusing on the rationale behind the design decisions and their biological analogies. As stated before, simplicity is one of our more important goals. A description of the underlying simulation techniques and the chosen implementation strategies is done in the next chapter.

# 4 IMPLEMENTATION

*"It is suggested that a system of chemical substances, called morphogens, reacting together and diffusing through a tissue, is adequate to account for the main phenomena of morphogenesis."*
— Alan Turing

This chapter gives an overview of the current implementation of the model proposed in Chapter 3, discussing technical decisions and programming details.

## 4.1 Simulation loop

A typical experiment for our model has two parts: first, it creates cells and defines their initial state, thus building the prepattern, and then it proceeds to the actual simulation. As discussed before, simulation runs with no external intervention, typically being stopped either manually by the user or when reaching a defined iteration count.

Most of the processing time is thus spent during the simulation loop. It is comprised of eight major steps, performed at each iteration. Those steps are:

1. Define domain size: cells are restricted to lay within a 2D rectangular region, which is normally fixed during the simulation. However, we have implemented a *packed* domain, which has dimensions that gradually increase based on the number of cells. This constrains all cells into a dense arrangement and is used to make growth uniform (instead of a patch of tissue with unconstrained boundaries). Therefore, in this step, the domain size is adjusted if needed, as discussed in Section 4.2.
2. Apply rules: sequentially evaluate conditions and perform actions on each cell, using its *current* state. Changes are accumulated in a separate *next* state, so there are no side effects caused by the order of cell evaluation.
3. Perform Nearest Neighbor Search: find the closest neighbors for each simulated cell. This is both the more complex and time-consuming step, which is described in Section 4.2.
4. Calculate diffusion: change the concentration of the current cell based on the sum of relative concentration differences among closest neighbors. Here the accumulated diffusion is computed for each reagent, which can be either isotropic or anisotropic. The actual implementation is detailed in Section 4.3.

5. Test for collisions: runs for the same set of closest neighbors. If there is some overlap between cells, a penalty-based repulsion offset vector is accumulated, which is discussed in Section 4.4.
6. Limit concentrations and positions: after the net diffusion is added to each reagent and the repulsion offset is accumulated for each cell, both chemical concentrations and 2D positions are adjusted to valid values. That is, cell positions are clamped to lay within the current rectangular domain. Similarly, negative reagent concentrations are set to zero and concentrations above a saturation limit (if defined) are clipped to it.
7. Add newborn cells: when cells divide during rule evaluation, new cells are just added to a list, but not immediately created. Here those new cells are created at once, effectively being added to the simulation.
8. Switch cell states: the final step is the switch from *next* state to *current* state[1]. After that, global cell statistics are updated, too.

We have posed the specific requirement for the implementation to be scalable, in the sense that it must be able to handle hundreds of thousands of cells. In fact, we pursue a directly parallelizable simulation loop, amenable to be run entirely on the GPU in the future. This demand reflects several design choices already described before and is also reinforced by several implementation strategies discussed in this chapter. This also implies the choice of a very simple representation for rules, presented in Section 4.8, instead of a much more complex interpreted runtime environment as it would be possible by using Lua, Python or other modern scripting languages.

## 4.2 Nearest Neighbor Search

Nearest Neighbor Search (NNS) can be stated as finding the point in a given set that is closest to a given query point. It is a classic topic from Computer Science and has many standard algorithms and plenty of literature about.

However, given the dynamic nature of the simulation and the desire to run it efficiently in both CPUs and GPUs, this led us to a search on specific algorithms that would be better suited to our model.

The main problem of traditional NNS algorithms is that they typically depend on a global

[1]This action is similar to the double buffering technique from Computer Graphics, where the reference to two memory regions, one visible and one hidden, is simply swapped to provide instantaneous visual update.

data structure, which is often inefficient to implement on a GPU because of the necessity to lock memory regions during updates. Thus a technique amenable to parallelization is imperative in our work.

Moreover, as cells move dynamically, we also need a data structure that can be continuously updated. That is, instead of rebuilding the data structure from scratch, at each iteration, we would like to resort to the spatial coherence of cells between two iterations to save both memory and computing time.

We have researched the little-known concept of *spatial sorting* and devised an Approximate Nearest Neighbor Search technique that is both easily parallelizable on a GPU and that accepts dynamic changes on the point set (MALHEIROS; WALTER, 2015; MALHEIROS; WALTER, 2016). In fact, we have evaluated that spatial sorting is optimal for our particular case, being also memory efficient. Its major limitation is being an approximate NNS method, so care must be taken for the trade-off between precision and performance. Therefore, we should be careful to employ it in situations where the arrangement of cells is dense, to reduce misses when locating the closest neighbors. We have thus implemented two other NNS techniques to cover particular situations.

First, we have added a 2D *k*-d tree implementation, based on the Nanoflann[2] library. It is an exact NNS, so it is being used both for benchmarking purposes and for the analysis of precision against spatial sorting, as discussed in Section 4.9. Moreover, this *k*-d tree technique can be used for arbitrary configurations of cells, that is, neither regular or dense arrangements. Second, we have added a very fast and simple NNS, for the particular situation where cells are arranged in static rectangular grids following a square lattice. As this is a common setup for traditional simulation of RD systems, such specialized NNS is indeed very useful. In fact, most experiments that do not depend on cell division can be run using this configuration.

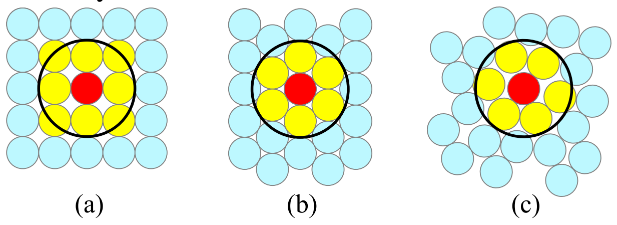
The actual NNS technique used is selected at runtime, before the simulation starts, based on the experiment input file. If there is a single rectangular grid and no division is employed, the square lattice NNS is chosen. If there is cell division and a packed domain is used, then spatial sorting is activated. Otherwise, the implementation falls back to the *k*-d tree NNS.

It should be noted that NNS algorithms typically address two problems: either locating the *k* nearest neighbors or locating all neighbors within a fixed distance *d*, which is called *range search*. In our model, we have the latter case, where cells have varying numbers of neighbors. For example, a cell on a border has fewer neighbors that a cell in the middle of the tissue. Therefore this distance, which we call *influence radius*, must be chosen carefully.

[2]<http://github.com/jlblancoc/nanoflann>

For our implementation, we have fixed an influence radius $d = 3.0$, centered at each cell. As each cell has a unit radius, this enables the location of either the immediate eight neighbors in a typical Moore neighborhood (defined on a square lattice) or six neighbors (in a regular hexagonal lattice). In fact, most compact yet non-regular arrangements of cells result in an approximation for the hexagonal case, which matches the best unit circle packing in the plane. Examples of these neighborhoods are shown in Figure 4.1.

Figure 4.1: Neighborhoods: (a) square lattice, (b) hexagonal lattice, and (c) general situation. The black circumference marks the influence radius for the red cell. Cells with their centers within this range are shown in yellow.



Source: own work.

That seemed to be the best choice, as it enables both typical regular grid configurations and a more general one within the same $d$ setting. Moreover, we have briefly experimented with varying influence distances, and the used value seems to be a good trade-off: too small $d$ prevents diffusion through nearby cells, whereas too large $d$ interferes with diffusion continuity and nullifies the typical RD effect of "short-range activation and long-range inhibition".

In fact, we have checked that most available implementations of RD systems employ a neighborhood with eight adjacent cells, running on a square lattice grid. For example, this is the default implementation for the discrete Laplacian operator on the OpenCL kernels in the Ready package (HUTTON et al., 2017). The use of a neighborhood with five cells, also known as Von Neumann neighborhood (TOFFOLI; MARGOLUS, 1987), seems to be much less common, albeit can lead to similar Turing patterns. In this latter case, because of the reduced connectivity, reagents take more iterations to spread, therefore diffusion rates need to be adjusted accordingly.

Domain packing works by simply setting the dimensions of a square domain based on the current number of cells, so that a more uniform compression can be achieved. The computed area $A$ of the desired domain is given by $A = na$, where $n$ is the current number of cells and $a$ is the individual area contribution of each cell. In theory, $a$ could vary from $\pi$, which is the area of a unit circle, to 4, which is the area of a unit square and matches the space taken by cells in a regular square lattice. For a perfect circle packing on the plane we might have

$a = 2\sqrt{3} \approx 3.4641$, which generates the hexagonal arrangement. In practice, a tighter domain is more useful, as cells are amenable to some overlap during compression. We have used $a = 3.2$ by default, which can be altered per experiment.

## 4.3 Reaction and diffusion

Although there were many RD systems proposed over the years, as previously discussed in Section 2.2, we opted to use just a single one throughout this research.

We chose one of the simplest, which is a slight variation of the two equations proposed by Turk in (TURK, 1991). It is interesting to note that Turk's system is itself based on the original reaction terms proposed by Turing in (TURING, 1952) (page 65), which are shown *ipsis litteris* in Equations 4.1 and 4.2. In these equations, $X$ and $Y$ are the reagent concentrations, whereas $\beta$ is per-location equation parameter.

$$f(X,Y) = \frac{1}{16}(16 - XY) \tag{4.1}$$

$$g(X,Y) = \frac{1}{16}(XY - Y - \beta) \tag{4.2}$$

In fact, Turk followed the general formulation for the RD differential equations given by Equations 2.1 and 2.2, replacing the constant $\frac{1}{16}$ by the scale parameter $s$, which modulates the intensity of the reaction part. By changing $s$ it is possible to alter the intrinsic size of the features that appear when the system stabilizes. Turk's resulting equations are shown as Equations 4.3 and 4.4.

$$\frac{\partial u}{\partial t} = s(16 - uv) + D_u \nabla^2 u \tag{4.3}$$

$$\frac{\partial v}{\partial t} = s(uv - v - \beta) + D_v \nabla^2 v \tag{4.4}$$

Our equations differ from Turk's in two small changes. First, we have replaced the constant value of 16 by the $\alpha$ parameter. Second, both Turk's and Turing's systems had a per-location parameter $\beta_i$, which had a slight variation around the fixed value of 12. This was the source of randomness that made patterns appear, as the initial concentrations of both reagents were all set as 4. Choosing a different set of $\beta_i$ values thus would result in a distinct pattern. We opted to follow the more traditional approach of making $\beta$ a fixed parameter, where the variations are driven by randomness in the initial concentrations. This also makes the implementation simpler, as a per-location $\beta_i$ attribute no longer needs to be stored. Our equations are shown in Equations 4.5 and 4.6.

$$\frac{\partial u}{\partial t} = s(\alpha - uv) + D_u \nabla^2 u \tag{4.5}$$

$$\frac{\partial v}{\partial t} = s(uv - v - \beta) + D_v \nabla^2 v \tag{4.6}$$

Because several works in the literature spatially modify equation parameters to achieve more complex patterns, we thought it would be useful to control both $\alpha$ and $\beta$ in our experiments. The actual discussion of attainable results and a mapping of the parameter space are drawn in Section 5.2.

The current design of our model makes it trivial to add other equations, as the only required computation is the reaction part. But we have opted to use only Equations 4.5 and 4.6, which we will call from now on the *Turing equations*, mainly for the sake of simplicity. By dealing with a single set of equations, we aim to reduce the space of possibilities and to assess how far we can go with a particular RD system. Moreover, by having a simple system, we reduce the number of parameters that need to be explored. Finally, we opted for the particular Turing equations because they are at the same time very simple and well studied[3].

Moreover, we understand that the Turing equations are already very expressive, being able to generate the fundamental pattern features we have seen on most RD systems: spots and labyrinths. In this work we opted to explore how just this basic features can be combined with other mechanisms, like growth and anisotropy, to generate more complex patterns.

As discussed in the design of our model, the reaction calculation is a rule-triggered action, which simply results in the evaluation of the quantities $s(\alpha - uv)$ and $s(uv - v - \beta)$ for a given cell, which alters the amount of reagents $u$ and $v$. Although we use $u$ and $v$ along the text, they refer to any two reagents.

The diffusion part of the equations is continuously calculated, at each iteration of the simulation. The calculation follows Fick's second law, by using a numerical implementation of the discrete Laplacian $\nabla$ operator over the nearest neighbors. In fact, the integration scheme is similar to a generalized five point stencil, where the center cell has the same weight as its neighbors (COORE; NAGPAL, 1998). We have analyzed that using weights, either fixed or based on the actual distances between cell centers, gives very little benefit: the resulting patterns

[3]At the time of writing, Turing's original RD paper has more than 10.700 citations in Google Scholar, even more than the paper on Turing machines (9.326 citations) and the Turing test (9.441 citations).

still demonstrate the same overall appearance, albeit with slight variations. Moreover, using fixed weights is computationally cheaper.

For isotropic diffusion of a reagent $R$, each pairwise contribution between two cells is multiplied by the combined diffusion rate $D_R$ given by Equation 4.7, where $D_{Ri}$ and $D_{Rj}$ are the respective per-cell diffusion rates for $R$ on cells $i$ and $j$. Typical RD systems have a single diffusion rate for a given chemical, whereas in our model we have a per-cell diffusion rate. To be compatible with previous systems, we have added the multiplication by 2 so that when $D_{Ri} = D_{Rj}$ we have $D_R$ assuming the same value[4].

$$D_R = \frac{2D_{Ri}D_{Rj}}{D_{Ri} + D_{Rj}} \tag{4.7}$$

Anisotropic diffusion is achieved by first modulating individual diffusion rates by the absolute value of the dot product between the normalized relative direction vector and the polarity of the respective cell. The relative direction vector is defined by the line segment that connects the centers of cells $i$ and $j$. Then Equation 4.7 is used to get the combined diffusion rate between these two cells. The overall effect is that full diffusion occurs when cells are aligned along the same direction as their polarity, having gradually weaker diffusion when cells get side-by-side, as discussed in Section 3.4.2. Equation 4.8 shows the modulated diffusion rate $\hat{D_{Ri}}$ for a single cell, where the polarity vector is given by $\vec{p}_i$ and the relative direction vector is given by $\vec{d}$.

$$\hat{D_{Ri}} = \left|\vec{d} \cdot \vec{p}_i\right| \frac{D_{Ri}}{||\vec{d}||} \tag{4.8}$$

For the numerical integration, we have evaluated several alternatives, and settled on the simplest approach: a straightforward Euler integration, as usually done in previous works (SANDERSON et al., 2006; MIYAZAWA; OKAMOTO; KONDO, 2010). We employ a fixed $\Delta t$ during the simulation, which multiplies both the reaction and diffusion parts of the Turing equations. The default value is 1, but it can be changed on a per-experiment basis. Higher-accuracy numerical methods also gave similar results, albeit with more computational cost. The overall strategy was to simply decrease $\Delta t$ if numerical instabilities appeared; still, most of the experiments were carried out with $\Delta t = 1$.

It is interesting to note that Equations 4.5 and 4.6 run adequately on single-precision floating-point calculations, so all simulation code uses only **`float`** type variables. We have determined that these equations need at least six significant decimal digits to actually create

[4]We thank Dr. Przemysław Prusinkiewicz for pointing to the correct equation, as our implementation was previously using the simpler but biologically inaccurate $D_R = \min(D_{Ri}, D_{Rj})$ calculation.

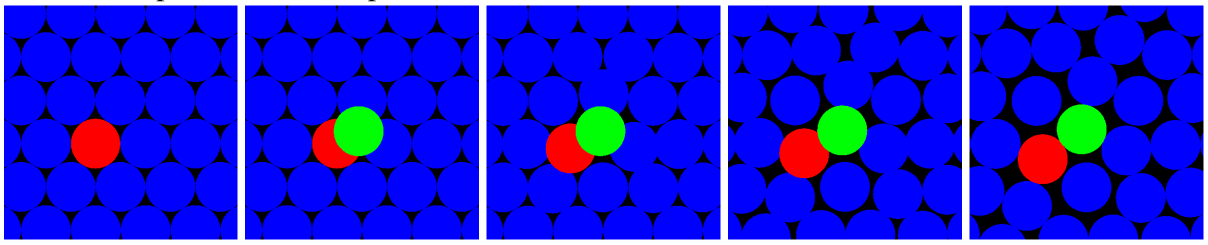
patterns and reach stable states. Therefore, the use of a `half` type is not applicable, and `double` type would consume twice the storage memory with little overall effect on patterns. Moreover, current consumer-level GPUs only have hardware-accelerated `float`-based computing pipelines.

## 4.4 Division and collision

We suppose that cells live within a limited environment. Since they are part of a tissue or immersed into a viscous extracellular fluid, their movement is highly constrained. Therefore, cells only move as a result of collisions, and collisions only happen after a division.

We have evaluated several possible mechanisms to model division, but have opted for the simplest one: division happens instantaneously, by creating a new cell at a unitary distance apart from the parent (distance measured between centers). Right after division, there is always some overlapping of the newborn cell with its parent and other nearby ones, which then causes collisions. Figure 4.2 illustrates the process.

Figure 4.2: The red cell divides and the green cell is created. In the following iterations collision occurs and spreads all cells apart.



Source: own work.

Collision testing runs for the same set of closest neighbors as diffusion, given by the fixed influence radius. When cells overlap we employ an impulse offset $\vec{o}$ that makes them move apart. This scheme is a simplified version based on (CLAVET; BEAUDOIN; POULIN, 2005), as we do not need to explicitly compute resulting forces or maintain momentum for the cells. The actual offset is computed from the summation of impulses given by nearby cells and is proportional to how much overlap there is. The overall effect is that cells slowly push others apart, taking possibly several iterations until they are not overlapping anymore. The actual calculation is given by Equation 4.9, where the relative direction vector between the current cell and its $i$-th neighbor is given by $\vec{d_i}$. It should be noted that impulse is only calculated if $||\vec{d_i}|| \in (0,2)$, as cells have unitary radii.

$$\vec{o} = \sum_i \left(0.25 - \frac{0.5}{||\vec{d}_i||}\right)\vec{d}_i \tag{4.9}$$

The child is identical to its parent, except for two attributes: birth and polarity. Its birth is set to the next iteration (because new cells appear in the simulation after the current iteration ends). The polarity for a child cell is defined relative to its parent's polarity. By default, the child polarity is the same but can receive an angular offset when created. For example, a new cell can be created with an orientation tilted by 10 degrees relative to its parent.

For simplicity, we have opted to maintain the size and concentrations of the child cell. A more accurate model would make two smaller cells, with reduced volume and thus the same concentrations. Cells would need to grow to match a target size, either draining external reagents through diffusion or producing enough quantities to compensate the change in volume. The side effect of this is that it would take more iterations to reach a local cell configuration similar to the one before mitosis. As we seek to have reagent concentrations stable unless explicitly modified by rules, it seems better to prevent the division from affecting concentrations. Moreover, cell division is so common in experiments that a less expensive implementation is preferred.

## 4.5 Chemically induced polarization

As discussed in Section 3.3, we have a specific action to establish the polarity of cells based on the current concentration of a given chemical reagent $R$. This mechanism works by evaluating the concentrations of nearby cells (akin to isotropic diffusion) and numerically computing the spatial derivative of the scalar field given by such concentrations. The resulting $\vec{p}$ vector is then normalized and assigned to the polarity of the current cell on the next iteration, thus not affecting the anisotropic diffusion for the current iteration. Equation 4.10 shows the actual calculation, where $R_i$ is the concentration of the $i$-th neighbor and $\vec{d}_i$ is the associated relative direction vector.

$$\vec{p} = \sum_i (R_i - R)\frac{\vec{d}_i}{||\vec{d}_i||} \tag{4.10}$$

## 4.6 Code organization

The current simulator is called Pattern Explorer. The full source code and all experiment files are publicly available[5] under the permissive MIT License.

It is implemented in C++ on a Ubuntu 14.04 Linux system. The code is organized into several modules. The more important ones are the NNS algorithms and the simulation loop itself. The simulation loop is quite compact, being about 500 lines of C++ code.

The simulation can be called from an API, enabling the compilation of text-mode standalone programs. In fact, all timing measurements were done with this setup. We have also implemented a simple graphical user interface (GUI), using OpenGL, FreeGLUT and AntTweakBar libraries. This interface is described in Section 4.7.

To enable a fast cycle of experimenting different ideas for pattern formation, we have defined a simple text-based language, for specifying both the prepattern and the rules to be followed. Each experiment file is usually self-contained. Section 4.8 gives a brief overview of this language.

Other modules are available for output in different formats. We have an automated screenshot feature for capturing the evolution of patterns, an export for the cell positions and colors in SVG vector format and a high-quality image interpolation. This interpolation makes possible the generation of high-resolution textures that can be applied to 3D models. The interpolation algorithm itself is based on Natural Neighbor Coordinates, as provided by the CGAL 4.9 library.

## 4.7 Interface

The GUI allows real-time visualization of the simulation, thus being able to monitor the pattern formation process and track the internal state of single cells. It first reads a text file with the commands for a single experiment and then opens an interactive window.

On the right of Figure 4.3 there is a canvas where cells are drawn as filled circles. Visible line segments indicate the current cell orientation. On the left, there is an information panel that also enables some parameters to be manually changed. In this panel, the simulation status (running or paused) is shown, besides the current iteration. Collapsible subpanels show several statistics for the simulation, including the number of cells. The Geometry subpanel shows maximum and minimum $x$ and $y$ coordinates and maximum, average and minimum number of

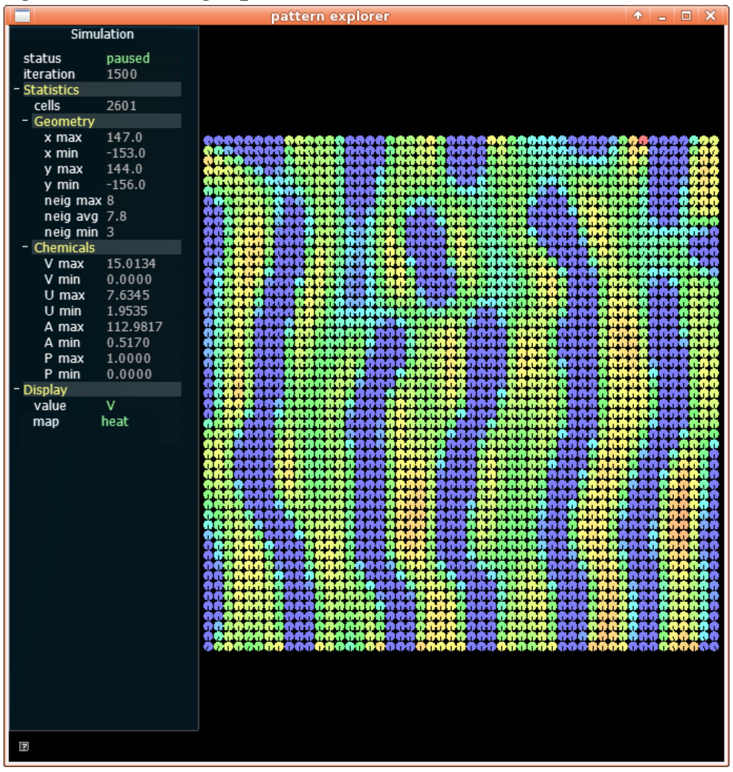
[5]<http://github.com/mgmalheiros/pattern-explorer>

neighbors. The Chemicals panel shows maximum and minimum concentrations for all defined chemicals. There is also a Display panel, controlling the color coding of cells, which maps the concentrations for the selected chemical reagent to a configurable colormap[6].

Additional control is possible through keyboard shortcuts and direct interaction using the mouse on the canvas, like panning, zooming or individual cell selection.

Figure 4.4 shows an additional subpanel, called Cell, which displays the state of the currently selected cell. The cell is identified by a circle drawn around it, showing the influence radius, which is used to select nearest neighbors. The nearest cells are also show filled in white. The subpanel shows the cell id, birth, age, number of neighbors, *x* and *y* coordinates, polarity (described as a unit vector with components *px* and *py*) and the current concentrations and diffusion rates for all reagents.

Figure 4.3: The graphical user interface for the simulation.

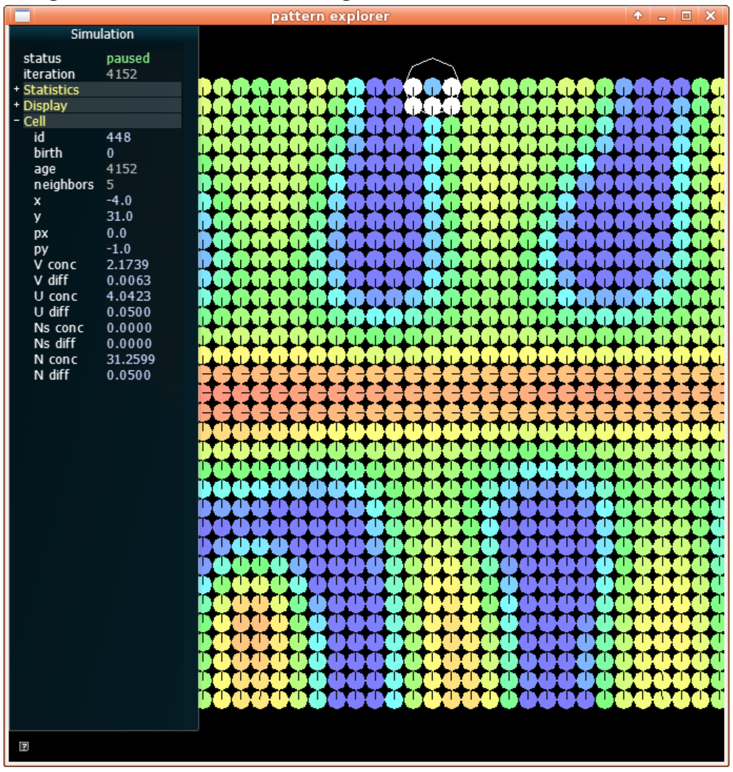


```
Simulation
status          paused
iteration       1500
- Statistics
  cells         2601
 - Geometry
    x max       147.0
    x min       -153.0
    y max       144.0
    y min       -156.0
    neig max    8
    neig avg    7.8
    neig min    3
 - Chemicals
    V max       15.0134
    V min       0.0000
    U max       7.6345
    U min       1.9535
    A max       112.9817
    A min       0.5170
    P max       1.0000
    P min       0.0000
- Display
  value         V
  map           heat
```

Source: own work.

[6]We understand the limitations of using a heat or rainbow colormap for conveying information through colors, as this practice is not recommended in the Scientific Visualization field. However, we are not concerned with the precise identification of specific chemical concentrations, instead just wanting to discern their overall variations on the simulated cells. Of course, a perceptually uniform colormap could be used as better default.

Figure 4.4: Panel showing individual cell information.



```
Simulation
  status      paused
  iteration   4152
+ Statistics
+ Display
- Cell
    id          448
    birth       0
    age         4152
    neighbors   5
    x           -4.0
    y           31.0
    px          0.0
    py          -1.0
    V conc      2.1739
    V diff      0.0063
    U conc      4.0423
    U diff      0.0500
    Ns conc     0.0000
    Ns diff     0.0000
    N conc      31.2599
    N diff      0.0500
```

Source: own work.

## 4.8 Text-based specification

Each experiment is comprised of a set of commands and rules, which are described by a simple text file. The following sections give a brief overview of the simulation language. This is exactly the same language used to generate the results presented throughout this work.

As usual, we strived for simplicity by having one statement per line and using a simple grammar, directed by English words. There is currently no explicit limit for the number of cells, chemicals or rules within a given experiment. Lines preceded by `//` are considered comments. A typical experiment is comprised of three sections: the simulation parameters, the definition of the prepattern and the genetic rules. An example is shown in Listing 4.1.

**Listing 4.1** Example experiment input file.

```
define chemical U
define chemical V

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.04
use chemical V conc 4 dev 2 diff 0.01
create sqr_grid 100 100

// -------- rules

rule always react U V scale 0.01 turing alpha 16 beta 12
stop at 10000
```

The simulation parameters define global settings that directly affect the simulation, and therefore are defined before it starts. The prepattern gives the initial chemical and positional configuration of cells, and therefore is also set before the beginning of the simulation. The rules are encoded into a compact numerical representation and stored into arrays, then being evaluated at runtime for each cell at each iteration.

### 4.8.1 Simulation parameters

Global simulation parameters are declared by **`define`** commands. The commands shown in Listing 4.2 define general properties. The **`define chemical`** command defines the name of a new chemical reagent, and optionally both an upper limit for its concentration (chemical saturation) and if its diffusion is anisotropic (relative to the polarity of each cell). The **`define division_limit`** command sets the maximum number of neighbors a cell must have to be able to perform division. The **`define domain`** command defines either a fixed domain with given width and height or a square packed domain (that grows based on a cell area contribution, which defaults to 3.2). The **`define time_step`** command alters the integration step for the diffusion calculation, which defaults to 1.

**Listing 4.2** Simulation parameters.

```
define chemical <string> [limit float] [anisotropic]
define division_limit int
define domain [int int | packed [area float]]
define time_step float
```

### 4.8.2 Default values

There are **`use`** commands to define default values for several cell attributes, depicted in Listing 4.3. Such values are used when issuing a **`create`** command to generate a new group of cells. The attributes can be changed again, affecting new cells created later on.

The **`use chemical`** command defines the initial concentration of a given chemical (and respective variation) and its default diffusion rate (and variation). Further calls to this command simply change the default values to be used during initial cell creation. The command **`use polarity`** sets a default orientation for cells, specified in degrees. Eventually, a non-polarized state can be specified with **`use polarity none`**. The **`use seed`** command sets the initial value for the random number generator. A value of zero means to use the current operating system time as the seed, yielding different patterns each time an experiment is run.

**Listing 4.3** Commands for setting default values.

```
use chemical <string> [conc float [dev float]] [diff float [dev float]]
use polarity [none | float [dev float]]
use seed int
```

The prepattern is created by the placement of groups of cells (Section 4.8.3) and by the subsequent explicit definition of particular cell parameters (Section 4.8.4). After the simulation starts, these commands cannot be issued.

### 4.8.3 Cell placement

Listing 4.4 shows commands for direct cell placement. The **`create cell`** command makes a single new cell at the given position. The **`create sqr_grid`** command defines a square grid of some number of columns and rows, centered at a given position. The **`create sqr_circle`** command defines a circle with given radius and made with cells following a square grid. The **`create hex_grid`** command creates a hexagonal grid of cells, and likewise, the **`create hex_circle`** creates a circle with cells following a hexagonal arrangement. Finally, the **`create shape`** command creates cells using positions given by a text file (each alphabetic character is a cell, any other character is empty space), following a square grid arrangement.

**Listing 4.4** Commands for creating and positioning cells.

```
create cell float float [fixed]
create sqr_grid   int int [at float float] [dev float] [fixed] wrap
create sqr_circle int     [at float float] [dev float] [fixed]
create hex_grid   int int [at float float] [dev float] [fixed]
create hex_circle int     [at float float] [dev float] [fixed]
create shape "file.txt"   [at float float]             [fixed]
```

The optional **`dev`** parameter indicates the amplitude of deviation from the positions actually set, whereas the optional **`fixed`** parameter marks cells not to be affected by collisions. The **`wrap`** keyword is specific for the square grid and indicates that a typical toroidal wrapping should be performed on the square borders (to make the resulting pattern tileable, for example).

### 4.8.4 Explicit parameter setting

There are also explicit commands to alter the concentration, diffusion rate or polarity for specific cells. Such instructions provide a finer control for altering the cells placed by the **`create`** commands. The **`set cell`** and **`set cells`** commands are shown in Listing 4.5, which modifies the cells associated with a given index or index range. The index is an integer that sequentially increases, and is given to each cell after its placement. The GUI makes possible the visual selection of cells and exhibits their corresponding indexes.

**Listing 4.5** Commands for altering chemicals for particular cells.

```
set cell int chemical <string> [conc     float [dev float]]
                               [diff     float [dev float]]
                               [polarity float [dev float]] [fixed]
set cells int to int chemical <string> [conc     float [dev float]]
                                       [diff     float [dev float]]
                                       [polarity float [dev float]] [fixed]
```

### 4.8.5 Rules

Rules are initiated by the keyword **`rule`** and are comprised of two parts: the condition and the action. The condition has an optional part that defines the initial and final iteration counts where such rule must apply, as depicted in Listing 4.6. These iteration counters are simply integer values that follow the **`from`** or the **`until`** keywords. The remaining of the condition is either comprised of the **`always`** keyword, a numerical comparison or the **`probability`** keyword. As expected, the **`always`** keyword is always true, whereas the numerical comparison must be evaluated each time to be known. The **`probability`** keyword is followed by a real number between zero and one, which is the probability of being true (a random number is drawn for each occurrence of this condition). The overall effect is that the associated action is performed only when the current iteration is within the specified interval and when the condition is shown to be true.

**Listing 4.6** Types of conditions for a rule.

```
rule [from int] [until int] always ...
rule [from int] [until int] if <par0> == <par1> ...
rule [from int] [until int] if <par0> != <par1> ...
rule [from int] [until int] if <par0> <  <par1> ...
rule [from int] [until int] if <par0> <= <par1> ...
rule [from int] [until int] if <par0> >  <par1> ...
rule [from int] [until int] if <par0> >= <par1> ...
rule [from int] [until int] if <par0> in <par1> <par2> ...
rule [from int] [until int] probability <par0> ...
```

Currently, the comparison parameters, here indicated by **`par0`**, **`par1`** and **`par2`**, can be real values or refer to the current cell state. Therefore is possible to refer to the concentration or diffusion rate of a given reagent by its name or the name of a mapping (described later on). Moreover, it is also possible to use its birth iteration (**`BIRTH`**), its current age (**`AGE`**) or the number of close neighbors (**`NEIGHBORS`**).

Note that the conditions involving iteration counts are not strictly necessary, as a similar control mechanism could be set within each cell by a specific reagent that does not diffuse, and that is increased by a tiny amount each iteration. We have thus opted to include the explicit references to the iteration count and cell age for convenience when designing patterns, due to the necessity of timed actions.

Almost all actions map to the cell events described in Section 3.3. Listing 4.7 gives a brief overview of each one. The **`change`** rule modifies a concentration or diffusion rate for the given chemical value, which optional variation. As said before, only the given amount to be added (or subtracted) to the current state is specified: it is not possible to absolutely set to a given level. The **`react`** action performs the reaction part of a reaction-diffusion equation on two reagents specified by the given strings. The optional **`scale`** parameter permits controlling the overall size of stationary waves by scaling the magnitude of the reaction for both reagents. The **`divide`** action can trigger a cell division, which actually happens after the current iteration is finished. Optionally can be defined an offset direction and variation (in degrees) relative to the current cell polarity.

**Listing 4.7** Types of actions for a rule.

```
... change <par0> float [dev float]
... react <string> <string> [scale <par0>] turing alpha <par1> beta <par2>
... divide [direction float [dev float]]
... map <par0> float float to <string> float float
... polarize <string>
... and
```

The **map** action is just a convenient mechanism for converting real quantities. It does not have any direct effect on the simulation: it simply defines a computed value that is associated with a string. This string can then be used as a parameter in other conditions or actions. The idea is to linearly map the given real interval from **par0** to another interval. The computed values are always restricted to be in the output interval (that is, the value is clamped when over the lower or upper limits). The **polarize** action adjusts the current cell polarity to follow the local gradient of the given chemical name. Finally, the **and** keyword combines the condition for this rule to the condition of the following rule, making a logical "and". Therefore, both conditions must be true for the following action to be triggered.

### 4.8.6 Utility commands

Listing 4.7 gives a brief overview of the utility commands implemented so far. Most are not directly related to the simulation itself, but they map concentration values to output colors, automate taking snapshots of the pattern or interrupt the simulation at a given iteration.

**Listing 4.8** Utility commands.

```
colormap select [heat | striped | gradient]
colormap slot int use color "<string>"
colormap slot int use rgb int int int
snap at int [...] [then exit]
snap from int repeat int step int [then exit]
stop at int
texture size int int
zoom level float
```

The **colormap** commands define a colormap, which can be used to map the concentration of a reagent to a set of given colors. The **heat** type is the traditional hue transition from blue (zero) to red (one). The **striped** and **gradient** types create a colormap based on colors placed on 100 slots, either by simple repetition or by linear interpolation, respectively. The color themselves can be specified by a CSS color string, a hex triplex or by individual RGB byte values. The **snap** commands generate automated screenshots to image files in the PNG format, at the given iteration counts. The **stop** command simply interrupts the simulation at the given iteration. The **`texture size`** command defines the output resolution for the high-quality interpolated texture. Finally, the **`zoom level`** command sets an initial GUI magnification.

### 4.8.7 Language overview

In this section we briefly summarize the commands of our text-based language. In Table 4.1 we list all statements, grouped by their category. We then link each command to the corresponding section where it is defined. Where applicable, we also add a reference to the section that explains the related model design rationale.

Table 4.1: Summary of language statements.

| Statement | Category | Description | Rationale |
|---|---|---|---|
| `define chemical` | simulation parameters | Section 4.8.1 | Section 3.1.3 |
| `define division_limit` | | | |
| `define domain` | | | |
| `define time_step` | | | |
| `use chemical` | default values | Section 4.8.2 | Section 3.2 |
| `use polarity` | | | |
| `use seed` | | | |
| `create cell` | cell placement | Section 4.8.3 | |
| `create sqr_grid` | | | |
| `create sqr_circle` | | | |
| `create hex_grid` | | | |
| `create hex_circle` | | | |
| `create shape` | | | |
| `set cell` | explicit parameters | Section 4.8.4 | |
| `set cells` | | | |
| `rule always ...` | rule conditions | Section 4.8.5 | Section 3.3 |
| `rule if ...` | | | |
| `rule probability ...` | | | |
| `...  change` | rule actions | Section 4.8.5 | |
| `...  react` | | | |
| `...  divide` | | | |
| `...  map` | | | |
| `...  polarize` | | | |
| `...  and` | | | |
| `colormap` | utility commands | Section 4.8.6 | |
| `snap` | | | |
| `stop` | | | |
| `texture` | | | |
| `zoom` | | | |

Source: own work.

## 4.9 Performance analysis

Simulation time is dependent on five major factors: number of iterations, number of cells, number of chemicals, choice of rules and choice of NNS algorithm. That is, for each iteration and each cell we must evaluate all rules and then locate the cell's neighbors. If we consider a static regular grid arrangement, with cells having a fixed number of neighbors which can be located at constant time, we have an expected time complexity of $O(mn(r+c))$, being $m$, $n$, $r$, and $c$ the number of iterations, cells, rules, and chemicals respectively.

If cell division happens, a regular grid can no longer be used, and we must either use an exact NNS, like $k$-d tree, or an approximate one like spatial sorting. Then we must take into account that these NNS techniques need two distinct operations: overall setup and per-cell query. During setup, we must either build the $k$-tree from scratch or spatially sort cell positions, which are both done in each iteration. Then, for each cell, we have to perform a query to locate the actual neighbors. We can suppose a fixed upper limit for the number of nearest cells because in practice we employ a limited influence radius and an explicit control to prevent excessive cell division.

The worst case time complexity for the $k$-d tree setup is $O(n\log n)$, whereas a single query operation is $O(\log n)$. The overall complexity of the whole simulation using $k$-d tree is $O(mn(\log n+r+c))$. We have deeply analyzed spatial sorting in (MALHEIROS; WALTER, 2016) and found an experimental complexity of $O(n\log n)$ for sorting, whereas the query can be done in constant $O(1)$ time. Thus, when using spatial sorting, we estimate the worst case time complexity also as $O(mn(\log n+r+c))$.

In fact, when running an experiment, most of the time is spent by the neighborhood location code (data structure setup and query operation) and cell interaction (diffusion and collision). Detailed timing information for all experiments discussed in the following chapters is given in Appendix A.

## 4.10 Summary

In this chapter we have discussed the current implementation of our model, focusing on simulation issues and on the textual language that drives the generation of patterns.

# 5 BUILDING BLOCKS

*"As you know, shibumi has to do with great refinement underlying commonplace appearances. [...] In art, where the spirit of shibumi takes the form of sabi, it is elegant simplicity, articulate brevity."*
— Trevanian

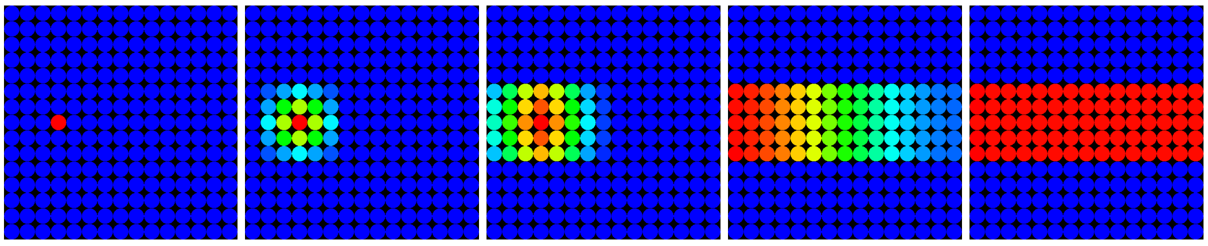
This chapter gives an overview of most features available in our model, using simple experiments to demonstrate them. As they can be combined in innumerable ways, we understand them as *basic building blocks* for more involved experiments, and therefore, for generating complex patterns. As cell division and chemical saturation augment the possibilities for pattern creation significantly, we discuss them separately, in Chapters 6 and 7, respectively.

## 5.1 Diffusion and gradients

An experiment is shown in Figure 5.1, where we define a single chemical reagent *R*. We define that cells will start with zero concentration of *R*, specified by the command **`use chemical A conc 0`**. We also define that the diffusion rate is zero, which means the cells are impermeable to the chemical *R*, which is specified by **`diff 0`**. Then, we create cells arranged in a square lattice of 15 by 15, which are implicitly initialized by the current concentration and diffusion rates.

Figure 5.1: Diffusion limited by impermeable cells, on iterations 0, 2, 5, 50 and 1000 (from left to right).

```
define chemical R

// -------- prepattern

use chemical R conc 0 diff 0
create sqr_grid 15 15
set cells 75 to 149 chemical R diff 0.1
set cell 108 chemical R conc 75
```



Source: own work.

However, as we are still defining the prepattern, we can explicitly set the diffusion rates of the cells inside the middle segment to 0.1. Moreover, we also set a single cell to have $R$ concentration of 75. There are no further rules, only a utility command to capture screenshots at the few iterations shown. When the simulation starts, the chemical concentration initially set in a single cell will spread among nearby permeable cells, until about 1000 iterations all such cells will reach the exact same concentration of 1 for $R$.

This is a simple experiment but it demonstrates important points about our model. First, diffusion rates can be set (and then altered) for each cell, which generalizes traditional RD models which typically employ a fixed rate for each reagent. Second, we show that by definition the simulation is a closed system, where total chemical quantities are preserved, at least until an explicit production, consumption or reaction rule is activated. Last, we create visual patterns by simply translating chemical concentrations to a colormap. In the experiment shown, we map the lowest concentration of the displayed chemical to blue, following a rainbow hue ramp until the maximum value is mapped to red.

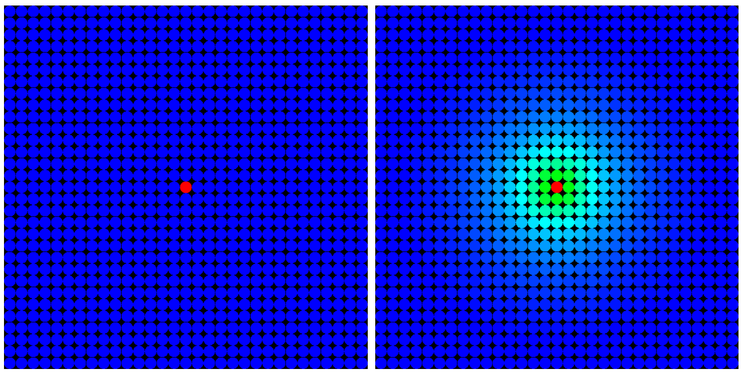
As diffusion is, by design, a continuous process, any pattern created by the distinct concentration of cells would eventually be "smoothed-out" and disappear into a constant concentration. Therefore, it makes sense to actively keep one or more cells producing more of a given reagent, so a continuous flow is achieved. However, as the simulation world is closed, the amount of chemicals would increase unbounded. Thus, a simple and interesting strategy is to define rules to both produce and consume a given reagent by specified quantities each iteration. When reaching the equilibrium, a steady gradient of concentrations would be formed.

Figure 5.2 shows an experiment that defines a stable radial gradient, defined by two chemicals, $R$ and $P$. The first defines the concentration gradient itself and the second is used to mark a single producer cell, being non-diffusible. The marked cell (where $P = 1$) continuously produces some amount of $R$, whereas all other cells (where $P = 0$) consume it by a smaller amount. By adjusting the production and consumption rates, we can define how far the gradient can reach. The resulting concentrations for $R$ stabilize around iteration 4000.

It is interesting to note that isotropic diffusion works similarly as a low-pass filter, akin to an image filter like the Gaussian blur, as already noted in (TURK, 1991). Therefore, the chemical concentrations of $R$ along such radial gradient approximate a 2D Normal distribution, centered at the producer cell. Therefore, by using this simple approach, it is not possible to construct linearly decreasing concentrations along the spatial derivatives of the gradient.

Figure 5.2: Stable radial gradient created from a single producer cell: chemical *P* is shown in the left and chemical *R* is visualized in the right.

```
define chemical R
define chemical P

// -------- prepattern

use chemical R conc 0 diff 0.05
use chemical P conc 0 diff 0
create sqr_grid 31 31
set cell 480 chemical P conc 1

// -------- rules

rule if P conc == 1 change R conc  0.8
rule if P conc == 0 change R conc -0.001
stop at 4000
```



Source: own work.

## 5.2 Standard Turing patterns

As discussed in Section 4.3, we have chosen a slight variation of Turing's original RD system as implemented by Turk in (TURK, 1991). The Equations 4.5 and 4.6 already give several parameters to control, so we have analyzed the generative possibilities of this RD system within our model and established a strategy to reduce its dimensionality.

We will use the terms *activator* and *inhibitor* loosely to distinguish the chemicals involved in Equations 4.5 and 4.6. Although there is a whole class of equations categorized by Meinhardt as Activator-Inhibitor systems (MEINHARDT, 1982; MEINHARDT; KLINGLER, 1987; MEINHARDT, 2009), we think that these terms are simple and didactic enough to refer to the *U* and *V* chemicals. Moreover, we will use the definition given by Turk where "in two-chemical reaction-diffusion systems the inhibitor is always the chemical that diffuses more rapidly". That is, *U* is the inhibitor and *V* is the activator.

An experiment for creating a standard Turing pattern is shown in Listing 5.1. Note that the chemical declared first in an experiment file is displayed by default. This can be changed interactively in the GUI.

**Listing 5.1** Experiment for a standard Turing pattern.

```
define chemical V
define chemical U
define time_step 1.0

// -------- prepattern

use seed 1
use chemical U conc 4 dev 2 diff 0.06
use chemical V conc 4 dev 2 diff 0.01
create sqr_grid 80 80

// -------- rules

rule always react U V scale 0.01 turing alpha 16 beta 12
stop at 5000
```

Albeit simple, this experiment implies the choice of 16 parameters that affect the visual outcome of the simulation. These parameters are summarized in Table 5.1 and discussed in the following sections.

Table 5.1: All parameters affecting a simple Turing pattern experiment.

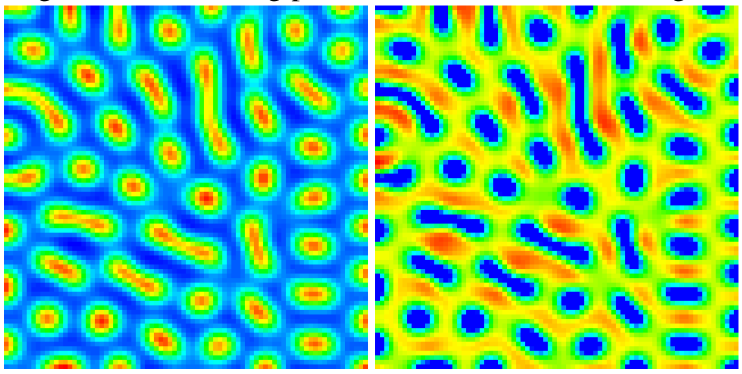
| Parameter | Dim. | Type | Description |
|---|---|---|---|
| shown reagent | 1 | boolean | which chemical to exhibit |
| time step | 1 | floating-point | discrete numerical integration step |
| random seed | 1 | integer | initial value for Random Number Generator |
| initial concentrations | 4 | floating-point | initial mean and deviation for reagents |
| diffusion rates | 2 | floating-point | diffusion rates for chemicals |
| grid type | 1 | boolean | choice of regular lattice |
| grid dimensions | 2 | integer | number of rows and columns |
| scale | 1 | floating-point | $s$ factor for RD equations |
| alpha and beta | 2 | floating-point | $\alpha$ and $\beta$ reaction parameters |
| iterations | 1 | integer | number of iterations to be simulated |

Source: own work.

### 5.2.1 Shown reagent and time step

The first parameter is the definition of which reagent to display during simulation. Turing patterns appear on the concentrations of both reagents, as the stability is reached because of their mutual interference. The choice of which reagent to exhibit is important because the overall results are similar but mostly inverse: at the regions where $U$ gets its highest value, $V$ gets its lowest and vice-versa. Thus, this choice depends mostly on aesthetic aspects because concentrations can be mapped to any colors as fit. Figure 5.3 shows the result of running the experiment of Listing 5.1, both for $U$ and $V$. As the diffusion rate for the inhibitor $U$ is always higher, is pattern features tend to have smoother outlines, whereas the concentration transitions are steeper for the activator $V$.

Figure 5.3: Resulting patterns, for $U$ (left) and $V$ (right).



Source: own work.

The second parameter is the time step. As we are not particularly interested in the precision of the numerical integration, we have used $\Delta t = 1$ for most simulations. It is large enough to reach stable patterns with only a few thousand iterations but robust enough to not introduce numerical errors. The value of 1.0 is the default, so it is explicitly set to a lower value only on the experiments that have high diffusion rates, for example. Another use is to slow down the combined reaction and diffusion, by using a smaller time step. This is very useful combined with cell division, making possible to separately fine-tune the growth rate and the velocity of reaction-diffusion.

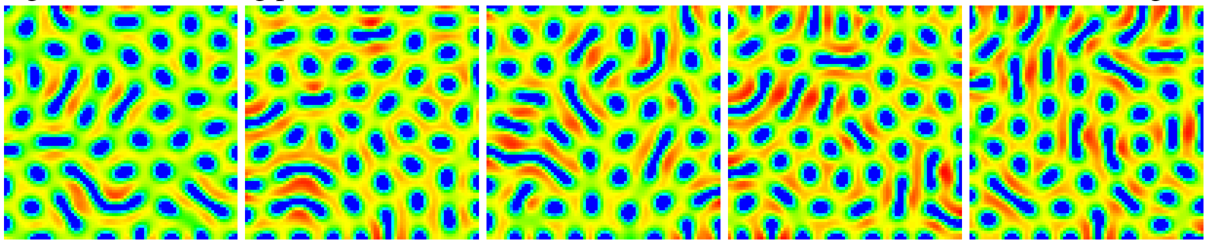
### 5.2.2 Random seed and initial concentrations

Another parameter is the random seed. By default, the simulation seeds the random number generator (RNG) with 1, which produces values following a uniform distribution. Such value can be changed to any other positive integer, thus generating deterministic yet distinct values when random numbers are needed[1]. This has a deep effect on the final pattern, because not only the initial concentration values are set using random numbers, but many other values in the definition of the prepattern and during the simulation itself. Most variation in our model is controlled by a mean value and the amplitude of its deviation, given by the keyword **`dev`**, as

[1]As discussed in Section 4.8.2, a seed of 0 initializes the RNG with the current time, generating a distinct pattern on each run of the same experiment.

discussed in Section 4.8. Figure 5.4 shows the result of running the base experiment with five distinct seed values.

Figure 5.4: Resulting patterns on $V$, with random seeds set to 1, 2, 3, 4 and 5 (from left to right).



Source: own work.

The definition of initial concentrations actually comprises four parameters: the mean value and its deviation, both for the inhibitor and the activator. After experimentation, we can say that only changing the mean value does not affect the resulting pattern but can take a longer time to stabilize. Keeping the same mean value but changing the amplitude of variation leads to similar patterns, and again affects convergence time: smaller amplitudes take longer. We have empirically found that a mean value of 4 and a deviation of 2 usually result in a fast stabilization, so we have used those parameters in most experiments. Interestingly, we later found that those are the same values suggested by (TURK, 1991).

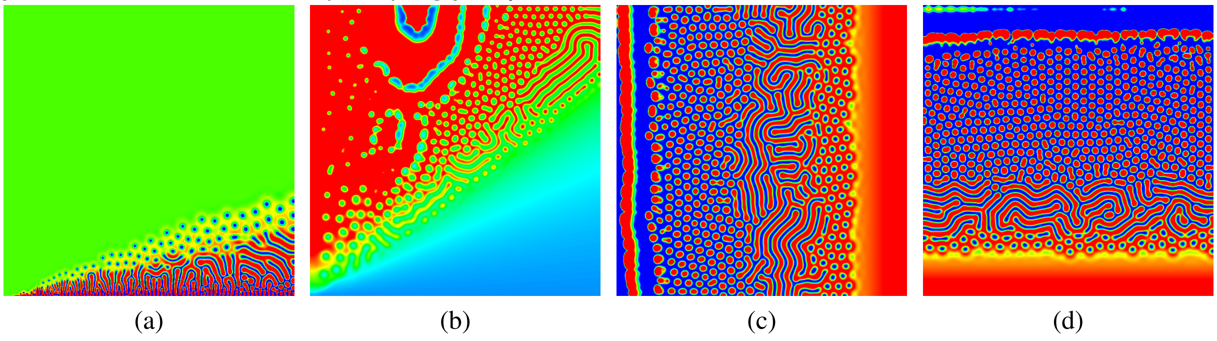
Initial variation does not need to be present on both reagents. If variation only happens for the activator $V$, a distinct pattern appears, albeit taking a similar time to stabilize as with both reagents. On the other hand, if the variation is restricted to only $U$, another distinct pattern appears, now taking longer to converge. Finally, if initial concentrations for both $U$ and $V$ are constant, no pattern is created, as expected.

### 5.2.3 Diffusion rates, scale, alpha and beta

Although RD systems have been thoroughly studied, we have not found a detailed mapping of the particular parameter space of Equations 4.5 and 4.6. We are aware of the work of (MIYAZAWA; OKAMOTO; KONDO, 2010), but their mapping was done for the linear RD equations, which have nine parameters just for the reaction part.

We have used the Ready package (HUTTON et al., 2017) to create parameter maps, where for each location in a large square lattice we continuously vary one or two parameters, thus summarizing into a single image a wide range of results. Parameter maps thus have helped to infer the specific relations between $D_u$, $D_v$, $s$, $\alpha$ and $\beta$. Four of the created maps are shown in Figure 5.5.

Figure 5.5: Parameter maps on $V$: (a) $D_u$ against $D_v$, (b) $\alpha$ against $\beta$, (c) horizontally varying just $\alpha$ and (d) horizontally varying just $\beta$.


(a) (b) (c) (d)

Source: own work.

We have found that even though $\alpha$ and $\beta$ directly change the nature of the equations, the usual pattern features can be achieved by adjusting only $D_u$ and $D_v$. Moreover, a stable pattern is usually achieved when $\alpha \in [10, 17]$ and simultaneously $\beta \in [10, 20]$. Parameters outside these intervals either do not generate patterns, leading to almost constant concentrations or drive the simulation into chaotic and unstable states. So we have fixed for all experiments $\alpha = 16$ and $\beta = 12$.

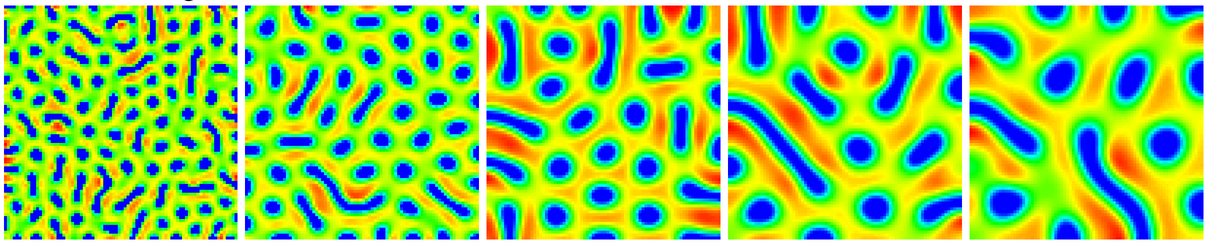
In fact, diffusion rates are the most important parameters, as together they specify both the kind of features that will evolve and the overall scale of the pattern. Moreover, they can be directly mapped to cell membrane permeability, which is a biological parameter much more plausible to vary in a living organism than the equation coefficients $\alpha$ and $\beta$.

Figure 5.5 (a) shows that there is a linear relationship between $D_u$ and $D_v$. Thus, if both diffusion rates are multiplied by the same factor, the overall pattern behavior stays the same, but the size of the features increases accordingly. This is illustrated in Figure 5.6, for five distinct multiplication factors and using the base experiment again. For the 3 and 4 factors we had to reduce the time step to 0.8 and 0.6, respectively.

Interestingly, using the parameter $s$ from the Turing equations has a similar, yet inverse, effect. This parameter modulates the reaction part of Equations 4.5 and 4.6 and can be seen as a coupled control of reaction strength or speed. We understand that the usage of this scale parameter is still biologically plausible, in the sense that it does not alter the actual behavior of the chemical process, but just controls the rate of production or consumption of $U$ and $V$ due to self-regulation.

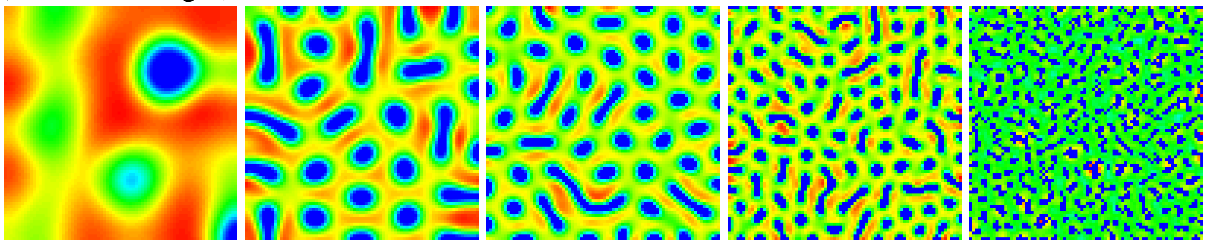
We show the effect of the change in $s$ in Figure 5.7, for five distinct values. In the base experiment and most simulations we use the reference value $s = 0.01$. Naturally, lower scale values reduce the amount of reagents produced and consumed, which will typically lead to longer convergence into stable patterns. As it can be seen, the results are very close, but not exactly equal, to the ones achieved through multiplication of diffusion rates.

Figure 5.6: Resulting patterns on $V$, with $D_u$ and $D_v$ multiplied by factors 1/2, 1, 2, 3 and 4 (from left to right).



Source: own work.

Figure 5.7: Resulting patterns on $V$, with $s$ assuming values 0.001, 0.005, 0.01, 0.02 and 0.04 (from left to right).

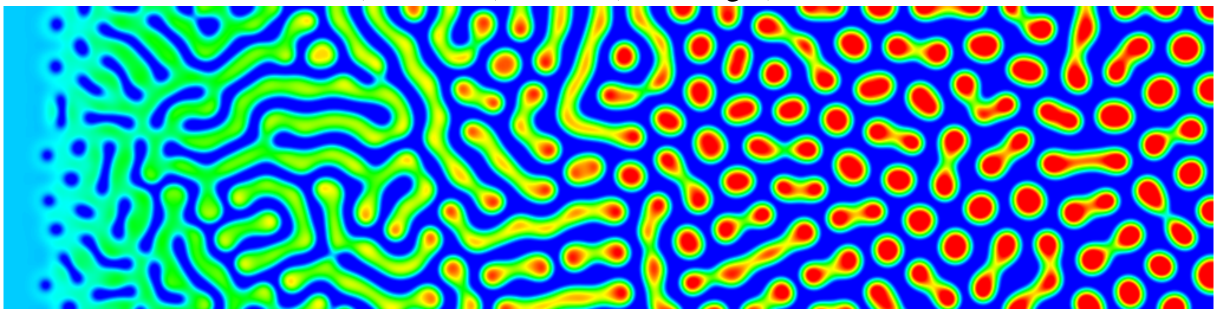


Source: own work.

As diffusion rate is a per-cell attribute, whereas the $s$ parameter is explicitly defined by the **`react`** action, we can combine both mechanisms. To control the overall pattern wavelength we can use $s$, which is globally defined in a given experiment. Local control of the pattern size can be achieved modulating both $D_u$ and $D_v$ with the same ratio.

By setting all these choices of parameters, we can now focus on the selection of the kind of pattern to be generated. By fixing both $\alpha$ and $\beta$ we just need to choose the correct ratio between $D_u$ and $D_v$. Moreover, by fixing $D_v = 0.01$, we have just one parameter controlling the overall features of the Turing pattern. Figure 5.8 shows a parameter map made with Ready, that shows a continuous range of the kind of patterns we would like to reproduce, letting $D_U$ vary within the interval $[0.0, 0.8]$ in the horizontal direction. The effect is the continuous change from negative spots to labyrinthine patterns to larger spots.

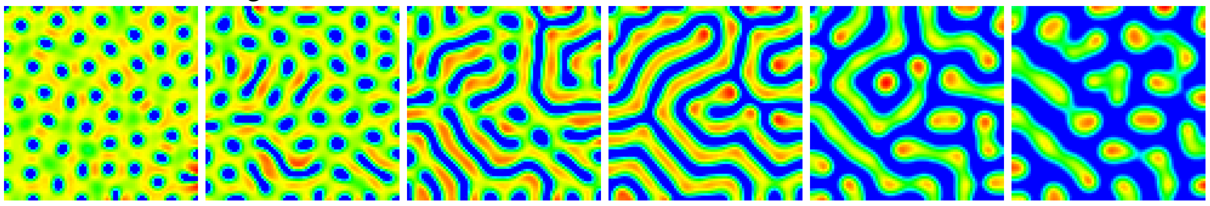
Figure 5.8: Parameter map for the Turing equations, showing the concentration for $V$. Diffusion rate for $U$ varies from 0.00 (on the left) to 0.80 (on the right).



Source: own work.

Figure 5.9 shows the result of running the base experiment with six distinct $D_u$ values. For the 0.40 and 0.80 diffusion rates we had to reduce the time step to 0.4 and 0.2, respectively. The color differences in relation to the parameter map are due to our implementation mapping the full concentration range into the heat colormap, whereas Ready uses a fixed interval.

Figure 5.9: Resulting patterns on $V$, with $D_u$ assuming values 0.04, 0.06, 0.10, 0.15, 0.40 and 0.80 (from left to right).



Source: own work.

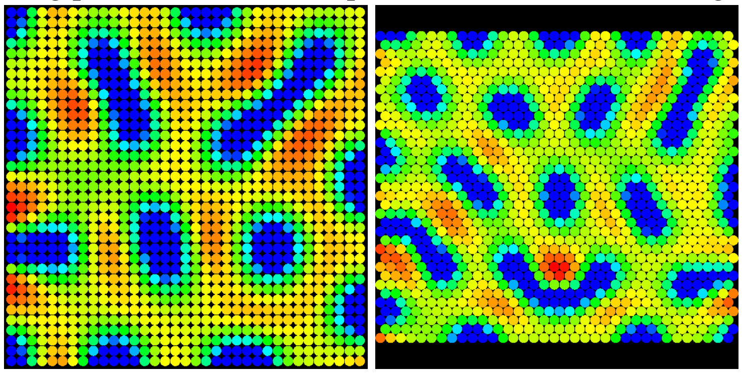
### 5.2.4 Lattice type and grid dimensions

Three other parameters are related to the initial disposition of cells in the prepattern: the regular lattice type and its dimensions, given by the number of columns and rows.

As discussed in Section 4.2, we implement both the regular square lattice and the regular hexagonal one. It is desirable to run simulations over this arrangement because it is compatible with most previous work involving reaction-diffusion. Moreover, if cell divisions do not occur, such fixed neighborhood makes possible the use of very specialized NNS routines, causing the simulation to run faster.

The actual choice is a matter of the experiment focus. The overall pattern does not depend on the arrangement but is still affected by it. In a regular square lattice most cells have eight neighbors, which helps to spread reagents in all directions. In a hexagonal lattice, most cells have six neighbors: as there are fewer neighbors, the overall diffusion for a given cell will be lower than in the square lattice, as the cell-to-cell transport is driven by the diffusion rate. This lower diffusion in its turn decreases the wavelength of the pattern. Moreover, as there are fewer relative directions between cells, the hexagonal lattice will produce increased directionality of borders. Figure 5.10 shows the resulting pattern for both lattice types, setting 40 columns and 40 rows and keeping all parameters the same for both.

Figure 5.10: Resulting patterns on $V$, for a square lattice (left) and a hexagonal lattice (right).



Source: own work.

### 5.2.5 Iteration count

The final parameter is the iteration count, which sometimes is explicitly defined in a **`stop at`** command. The number of iterations to be simulated is largely dependent on the experiments, as some patterns could take several tens of thousand iterations to converge.

We have added the possibility of detecting chemical stability when cell concentrations for all reagents do not change between iterations. In practice, we accept very small differences between the concentrations of the current iteration and the previous one. This simple feature is disabled by default, but it can be enabled by a command-line flag to automatically stop as soon as possible any simulation. This is particularly useful when running automated generation of experiments.

## 5.3 Parameter modulation

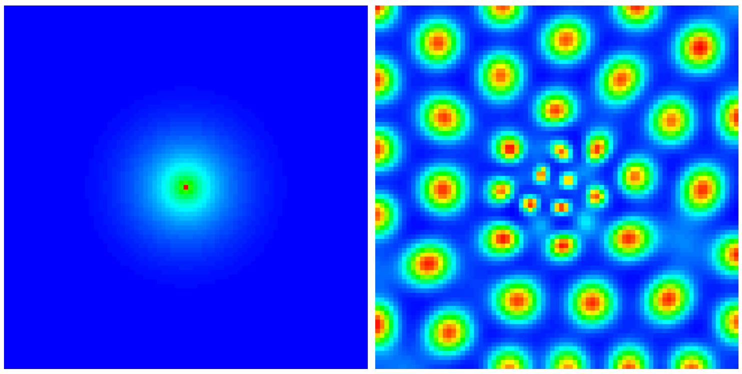
Another feature of our model is called *modulation* and refers to the spatial control of a parameter based on local chemical concentration. It can be achieved by simply creating a mapping of real values through the **`map`** action. For instance, this is the method used in (WITKIN;

KASS, 1991) for their specific space-varying patterns.

In Figure 5.11, we show a radial gradient on the *R* reagent and a Turing pattern on the *U* reagent. The gradient was created by a single producer cell at the center, which is then used to control the *SCALE* parameter. This parameter is then used to modulate the $s$ scale factor for the Turing reaction involving $U$ and $V$. This results in a continuous transition of spot size from center to borders.

Figure 5.11: Parameter modulation: radial gradient (left) inducing the variation of the scale parameter for a Turing pattern (right).

```
define chemical U
define chemical V
define chemical R
define chemical P

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.04
use chemical V conc 4 dev 2 diff 0.01
use chemical R conc 0       diff 0.05
use chemical P conc 0       diff 0
create sqr_grid 81 81
set cell 3280 chemical P conc 1

// -------- rules

rule if P conc == 1 change R conc  2.5
rule if P conc == 0 change R conc -0.001
rule always map R conc 0.2 3 to SCALE 0.005 0.03
rule always react U V scale SCALE turing alpha 16 beta 12
stop at 10000
```

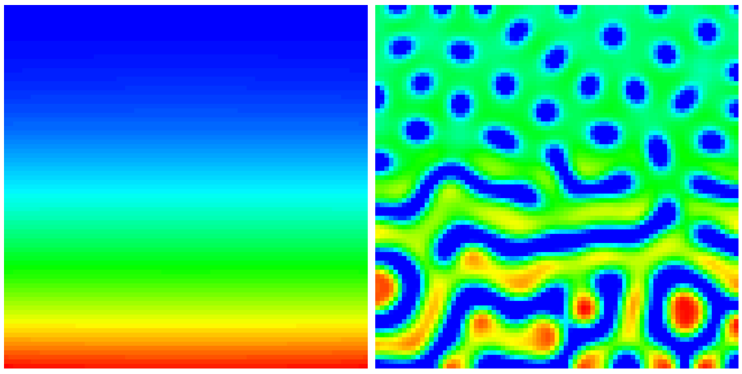

Source: own work.

Although parameter modulation is a mechanism commonly used to directly control features of a larger pattern, we have used it sparingly, mostly to map reagent quantities during experiments. In fact, we consciously avoided modulating the $s$, $\alpha$ and $\beta$ parameters for the Turing equations, because it is not biologically plausible that a given chemical reaction functions differently at different parts of a rather small domain, at a mesoscale level. In fact, as discussed in Section 5.2, most effects of change in $s$, $\alpha$ and $\beta$ can be reproduced by adjusting just the $D_u$ and $D_v$ diffusion rates.

As another example, in Figure 5.12, we show a vertical gradient on the $R$ reagent and the associated Turing pattern on the $V$ reagent. The gradient was created by a row of producer cells at the bottom, which is then used to setup the initial diffusion rates $U$ and $V$. The experiment first waits until the gradient stabilizes at iteration 5000, then sets at once the diffusion rate for $U$ based on the concentration of $R$. A fixed diffusion rate of 0.01 for $V$ is also set at this time; if done earlier, the initial random concentrations would have been diluted and pattern formation would take much longer. The Turing reaction involving $U$ and $V$ is also started at this iteration. This results in a continuous transition from stripes to spots, akin to a traditional parameter map.

Figure 5.12: Parameter modulation: vertical gradient (left) inducing a continuous pattern feature variation (right).

```
define chemical V
define chemical U
define chemical R
define chemical P

// -------- prepattern

use chemical U conc 4 dev 2 diff 0
use chemical V conc 4 dev 2 diff 0
use chemical R conc 0       diff 0.1
use chemical P conc 0       diff 0
create sqr_grid 81 81
set cells 0 to 80 chemical P conc 1

// -------- rules

rule if P conc == 1 change R conc  2
rule if P conc == 0 change R conc -0.02
rule always map R conc 0 100 to RATE 0.04 0.15
rule from 5000 until 5000 always change U diff RATE
rule from 5000 until 5000 always change V diff 0.01
rule from 5000 always react U V scale 0.01 turing alpha 16 beta 12
stop at 10000
```


Source: own work.

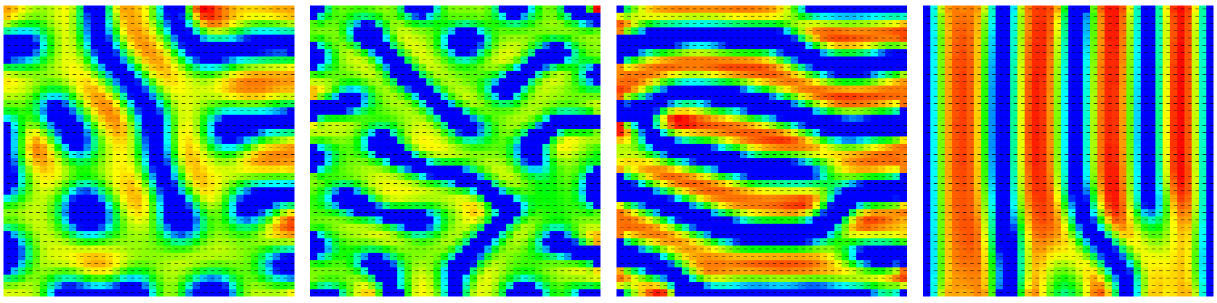
## 5.4 Anisotropic diffusion

In our model anisotropy is enabled per-reagent. As discussed in Section 4.3, it restricts the diffusion along the direction given by the polarity of each cell. If a given cell is not polarized, its diffusion is isotropic. Even if another cell has polarity, diffusion is normally run for chemical reagents where anisotropy is not enabled.

Although several prior works employed anisotropy, (KIM; LIN, 2007) was the first to report a particular non-obvious behavior: when paired with an RD system, anisotropy works best when applied to a single chemical. That is, instead of modulating the diffusion of both $U$ and $V$, it is more effective to enable it to either $U$ or $V$.

Figure 5.13: Anisotropy: (a) both chemicals are isotropic, (b) both are anisotropic, (c) only $V$ is anisotropic and (d) only $U$ is anisotropic.

```
define chemical V //anisotropic
define chemical U //anisotropic

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.08
use chemical V conc 4 dev 2 diff 0.01
use polarity 0
create sqr_grid 40 40

// -------- rules

rule always react U V scale 0.01 turing alpha 16 beta 12
stop at 5000
```



Source: own work.

For example, take the simple experiment shown Figure 5.13. All cells are polarized into the same horizontal direction, and all configurations start from the same random concentrations of $U$ and $V$. The pattern shown in (a) is the typical Turing pattern achieved when both reagents have isotropic diffusion. When anisotropy is enabled for both $U$ and $V$ in (b), by adding both **`anisotropic`** keywords, the result is not the expected one: the overall pattern does not gain directionality. In fact, the combined effect of anisotropy in both reagents seems to cancel out.

The only perceptive effect is the small reduction in the size of the features, giving that diffusion is now more limited for both chemicals.

Again in Figure 5.13 (c), if we enable anisotropy only to $V$, the results are striking: stripes are formed along the polarity direction. So this is a simple mechanism that can be used for creating oriented stripes: just enabling anisotropy for the activator chemical ($V$ here). But a more interesting phenomenon occurs if we enable anisotropy for the inhibitor ($U$): stripes appear perpendicularly to the polarity direction, as shown in (d).

This complex behavior is still compatible with the notion of simplicity: in fact, having just one reagent diffusing anisotropically is conceptually simpler than both of them having anisotropic diffusion at the same time. When we actively control the polarity of cells, this behavior can be used in many interesting ways, particularly in reproducing pigmentation patterns found in Nature.

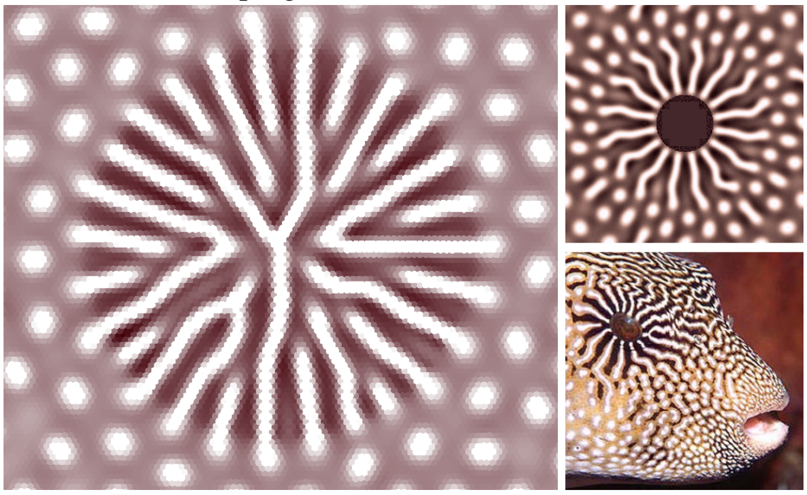
## 5.5 Polarization

We understand that polarization is an important biological mechanism (GOEHRING; GRILL, 2013), and a crucial one for our model: we must have a way to specify the direction of growth and the orientation of stripes, for example. In fact, we presume that most directionality from biological patterns comes either from explicit chemical polarization driven by morphogen gradients (STATHOPOULOS; IBER, 2013), as described here, or from the division process during growth, which is explored in Chapter 6. Note that anisotropic diffusion directly depends on cell polarity to have a preferred direction for reagent transport between cells.

We can use a circular gradient at the center of the pattern to induce cell polarization. Figure 5.14 shows the result of enabling anisotropic diffusion only for $V$, where cells are oriented along the spatial derivative of $R$. The effect is that stripes orient along radial lines. Here we employ a regular hexagonal lattice of cells. In the same figure we show the equivalent result from (SANDERSON et al., 2006), which employed explicit modulation of the Turing RD equation parameters: $\beta$ was controlled by the relative angle to the center of the spot, whereas $s$ was driven by the distance to the same center. The experiment file is shown in Listing A.1.

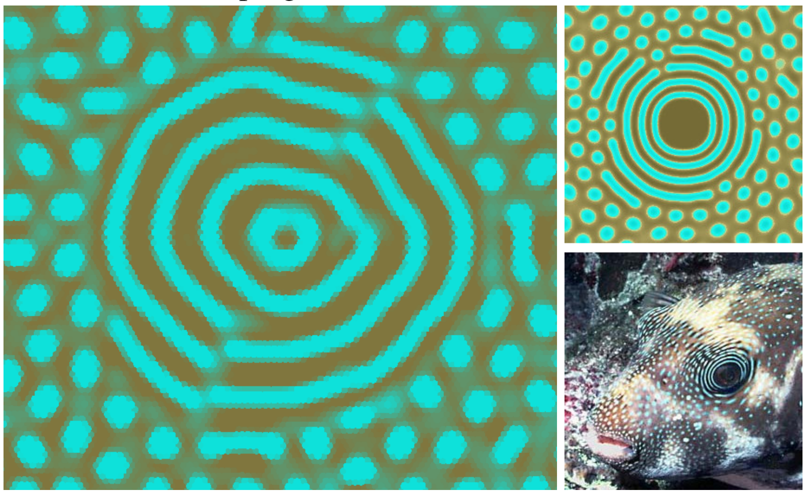
If we now set anisotropy only for the $U$ reagent, the RD system creates concentric, yet slowing moving, stripes, as seen in Figure 5.15. Note that the parameters for the Turing equations were only slightly altered from the previous experiment, yet we can achieve a very similar result to the one shown in the right column, which still used direct modulation of parameters. The experiment file is shown in Listing A.2.

Figure 5.14: Effect of anisotropic diffusion for *V* reagent only (left) and similar previous result, made by parameter modulation (top right).



Source: own work (left) and from (SANDERSON et al., 2006) (right column).

Figure 5.15: Effect of anisotropic diffusion for *U* reagent only (left) and similar previous result, made by parameter modulation (top right).



Source: own work (left) and from (SANDERSON et al., 2006) (right column).

We note that, in both situations shown previously, equivalent results were obtained through a simpler combination of mechanisms than in (SANDERSON et al., 2006). Which are, perhaps, more biologically plausible explanations for such natural pigmentation patterns.

## 5.6 Summary

In this chapter we have shown the results of simple experiments, both to demonstrate features of the proposed model and to illustrate the diversity of patterns that can be already generated.

# 6 GROWTH

*"It is very exciting to see that structures that we used to think of as very complex turn out to be very simple in principle."*
— Przemyslaw Prusinkiewicz

Cell division is one of the major features of our model, as it enables the actual growth of the simulated tissue and the resulting pattern evolution over a growing domain. Although division has been proposed before in prior works, as discussed in Section 2.3, our approach is one of the very few to couple division with reaction-diffusion in a biologically plausible way.

In this chapter we present in detail experiments involving cell division, which make possible the creation of a broad range of patterns, either matching early works or novel ones. Although the results were deemed satisfactory, we had at first the expectation that complex natural patterns like the ones from a zebra or a giraffe would be easy to attain, once growth and reaction-diffusion were combined. But early experiments in this direction failed, which led us to look deeply into the problem of pattern enlargement, which is described in Section 6.6. The knowledge obtained led us to experiment with chemical saturation, later shown to be the missing piece of the puzzling mechanism for creating natural patterns, which is then discussed in Chapter 7.

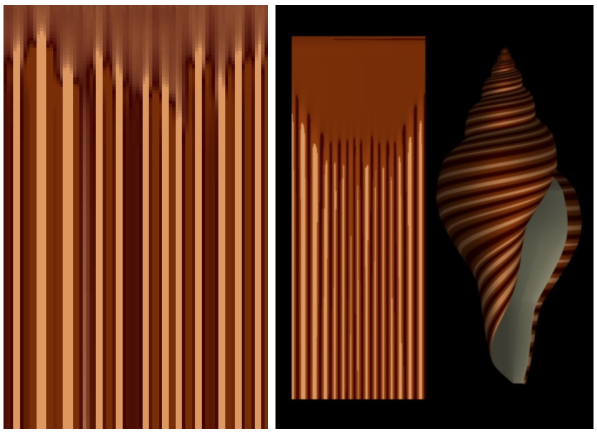
## 6.1 Growth front

We use the term *growth* in a broad sense, where any kind of cell division occurs. As cell division is a discrete event, governed by one or more explicit rules, we can use it far more diversely than just simulating the growth of a dense tissue.

The first experiment creates a growing front of cells, at one of the edges of an already existing pattern. The idea is to reproduce the main mechanism that defines pigmentation of a mollusk shell. After an initial row of cells is established, they divide creating below a new row of cells, and this process is repeated only for the newborn cells. At the same time, we freeze the chemical concentration on the parent cells, making the diffusion rate drop to zero, whereas diffusion continues throughout the bottom row. We also keep reaction happening on the cells in the bottom row, so that the concentrations can change over time. The result is a frozen reaction-diffusion system in time, where each row represents a particular iteration. The final result is shown on the left of Figure 6.1, which can be compared to the simulation done by (FOWLER;

MEINHARDT; PRUSINKIEWICZ, 1992). The strict division control is done by conditions depending on the age of the current cell, as given by the **AGE** variable. The experiment file is shown in Listing A.3.

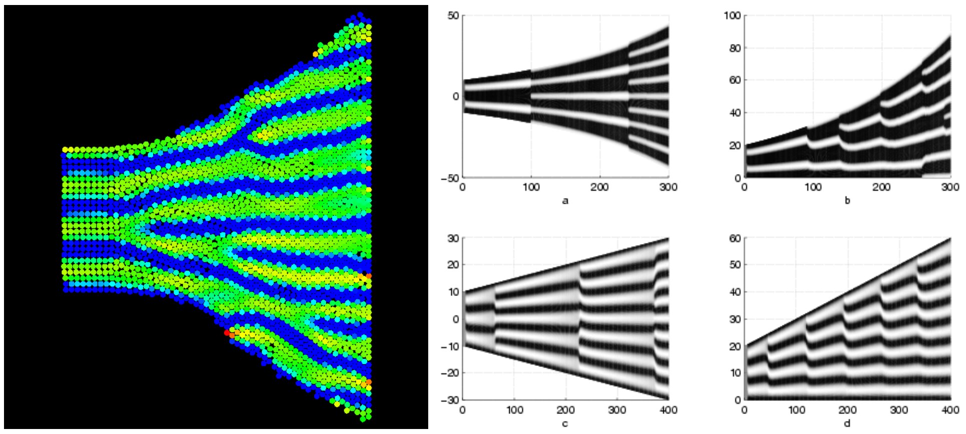
Figure 6.1: Growing front that freezes the pattern generated in each row (left) and prior results (right).



Source: own work (left) and from (FOWLER; MEINHARDT; PRUSINKIEWICZ, 1992) (right).

Another experiment is shown in Figure 6.2, where the effect of expanding domain is simulated by a continually growing front that reaches the limits of the simulation area. We can observe the appearance of new stripes as the tissue gets larger, maintaining the usual spatial wavelength of periodic stripes, as already observed in (PAINTER, 2001). Note that the stripes emerge from both the directionality of cell division and the use of anisotropy on $V$. The dimensions of the simulation area are set with the **define domain** command. The experiment file is shown in Listing A.4.

Figure 6.2: Effect of expanding domain generated by constrained growth (left) and prior results (right).

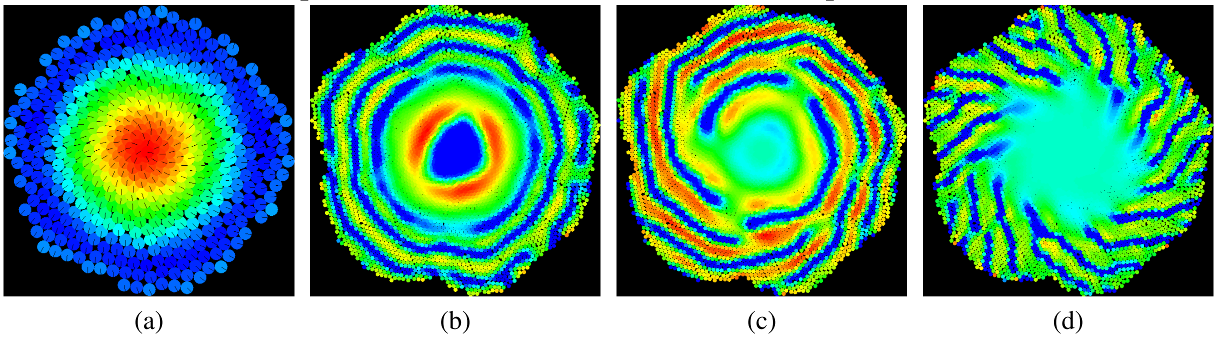


Source: own work (left) and from (PAINTER, 2001) (right).

## 6.2 Center growth

A particular structural pattern emerges when we keep dividing only the last child cell and enforcing that the newborn cell is at exactly 137.5 degrees from its parent. The result is that the divisions occur mostly at the same spot, in the center of the pattern, evenly spreading older cells in an almost perfectly regular arrangement, similar to the natural phenomenon of *spiral phyllotaxis*, shown in close-up in Figure 6.3 (a). It is interesting to note that this is achieved differently from (FOWLER; PRUSINKIEWICZ; BATTJES, 1992), where new cells were placed on the outside of the shape. And it is still different from the mechanism described in (PENNYBACKER; NEWELL, 2013), which employs a specific RD model to create spots matching the correct positions. In Figure 6.3, from (b) to (d) we show distinct patterns formed with the same growth parameters, but with distinct $D_u$ and anisotropy settings. The experiment file is shown in Listing A.5.

Figure 6.3: Close-up detail (a) after only 1000 iterations. Pattern (b) has $D_u = 0.08$, (c) has $D_u = 0.08$ and anisotropic $U$ and (d) has $D_u = 0.04$ and anisotropic $V$.



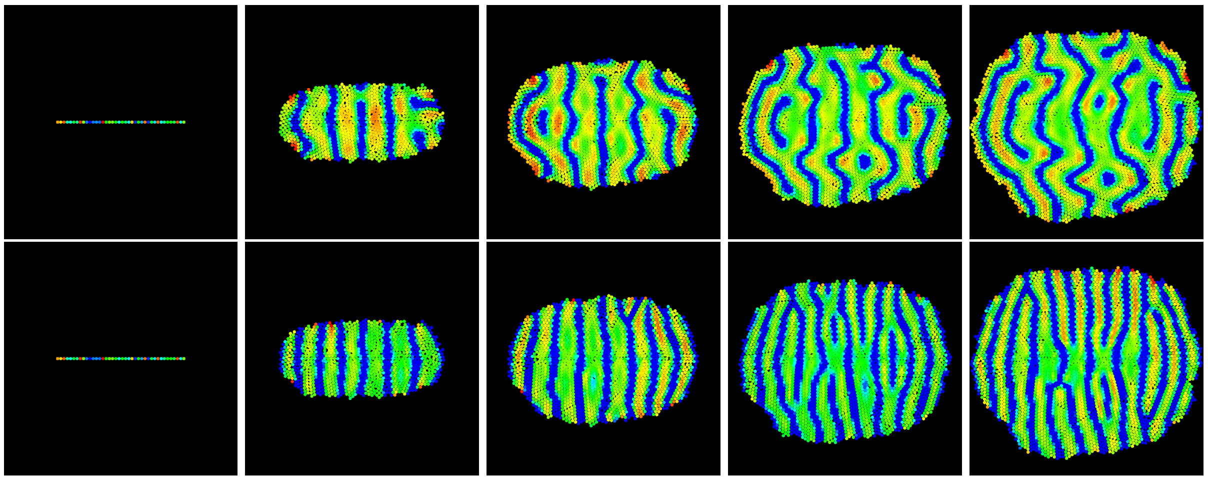
Source: own work.

Another experiment is depicted in Figure 6.4, where an initial “spine” of 40 cells is defined. We want to explore the hypothesis from Biology where the neural crest[1] induces the pigmentation patterns in a newly formed organism. Those cells are marked as **`fixed`**, so they are not displaced by collisions. Moreover, the experiment defines a probability of $4/1000$ of mitosis for each one of these initial cells, creating a child either right above or right below. Each new cell is prevented from dividing, so older cells are pushed apart from the center by other newer ones. Here we explicitly prevent excessive compression by setting **`define division_limit 8`**, that is, only cells with eight or fewer neighbors can divide. Normal reaction and diffusion are enabled for all cells. A few initial iterations are enough to establish an alternating pattern on the

[1]The neural crest is a temporary group of cells that arise in vertebrate embryos, being responsible for the formation of many types of cells, including melanocytes (which are the cells that produce skin pigmentation).

spine, which then serves as the generator for stripes. Note that diffusion is isotropic for the first run of this experiment. If we enable anisotropy for the $V$ reagent we get much more uniformly oriented stripes. The experiment file is shown in Listing A.6.

Figure 6.4: Patterns created by a fixed row of cells at iterations 0, 4000, 8000, 12000 and 16000 (from left to right). Top row employs isotropic diffusion, whereas bottom row has anisotropy enabled for $V$.

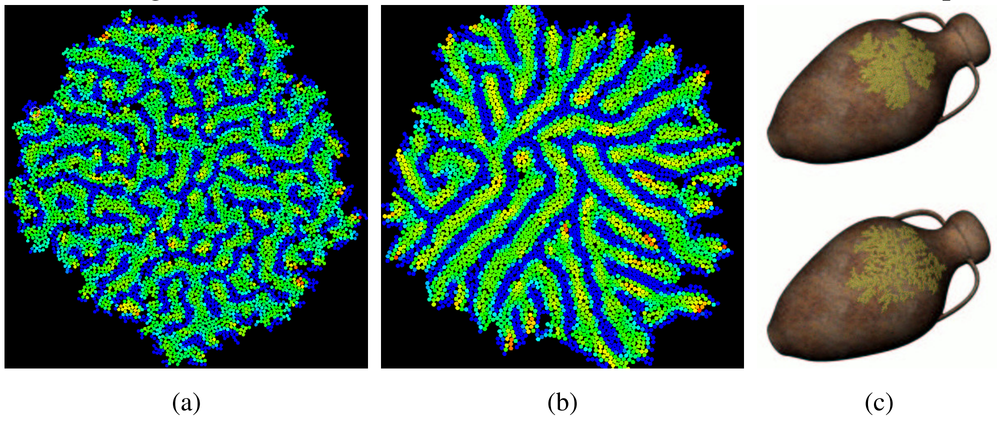


Source: own work.

## 6.3 Border growth

Here we briefly discuss the situation where growth is constrained to occur only at pattern borders. Thus, we can use the number given by the **NEIGHBORS** counter to initiate cell division. In this way, the tissue expansion occurs only at its edges. By affecting the polarity of the newborn cell, it can result in either directional or random growth.

Figure 6.5: Border growth: (a) random division, (b) directed division and (c) two prior results.



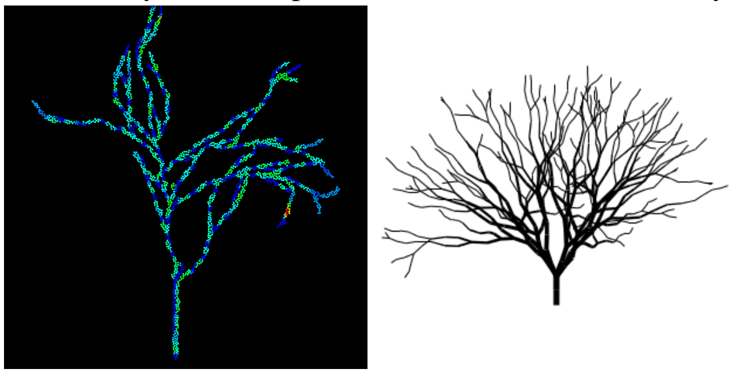
Source: (a) and (b) own work and (c) from (DESBENOIT; GALIN; AKKOUCHE, 2004).

Figure 6.5 shows two variations of the same experiment, where cells divide only if they have three or fewer neighbors. We start from a single cell. Anisotropic diffusion is kept for one of the two reagents. A reaction is then performed at each iteration to induce oriented patterns. On the left, the division spawns cells with arbitrary polarity, whereas on the right, each new cell diverges exactly 10 degrees from its parent. This is similar to the general process of Diffusion-Limited Aggregation (DLA), used for example when modeling the growth of lichen (DESBENOIT; GALIN; AKKOUCHE, 2004). The experiment file is shown in Listing A.7.

## 6.4 Branching growth

Analogously to the rules of a stochastic L-system (PRUSINKIEWICZ; JAMES; MĚCH, 1994), we may constrain division to occur when a cell has two or fewer neighbors. As this restricts division to very loose cells, it tends to generate filaments in an approximate linear growth. The experiment depicted in Figure 6.6 adds another criterion for subdivision: a less probable directional division of 20 degrees from its parent. This results in a bifurcation, which then creates a branched structure. As division is restricted to cells on the "tip" of each branch, as soon as they collide with other structures the neighborhood gets denser and division stops locally. The experiment file is shown in Listing A.8.

Figure 6.6: A randomly branched pattern (left) and a stochastic L-system (right).



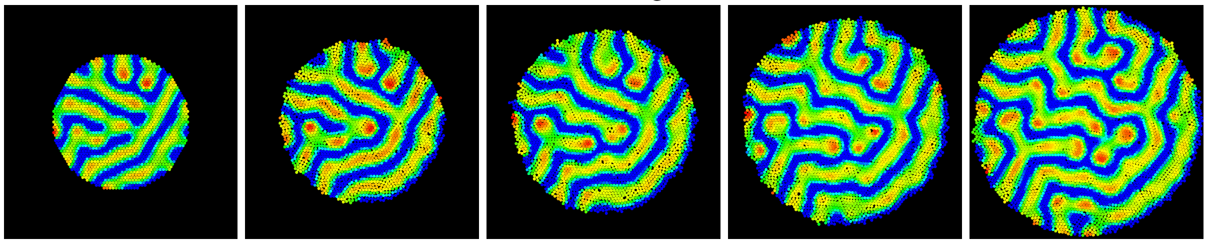
Source: own work (left) and from (PRUSINKIEWICZ; JAMES; MĚCH, 1994) (right).

## 6.5 Uniform growth

In this section we discuss two experiments involving uniform growth, where divisions occur with equal probability for each cell. Although this is the most common case of observed tissue growth in Biology, the underlying regulatory mechanisms are still not completely understood (SHRAIMAN, 2005).

The first experiment is shown in Figure 6.7. It creates cells initially packed into a circular region and lets the standard Turing patterns evolve until iteration 2000. After this, a uniform probability of 1/10000 of cell division is applied until iteration 15000. As mentioned earlier, a typical RD pattern can adjust itself to changes in its domain, keeping the wavelength for its features, like the spacing of its spots or width of its stripes. So we can see that, by having the reaction active during the growth process, the pattern adjusts into a larger area, introducing new details with the same wavelength. Therefore the original pattern is significantly modified. The experiment file is shown in Listing A.9.

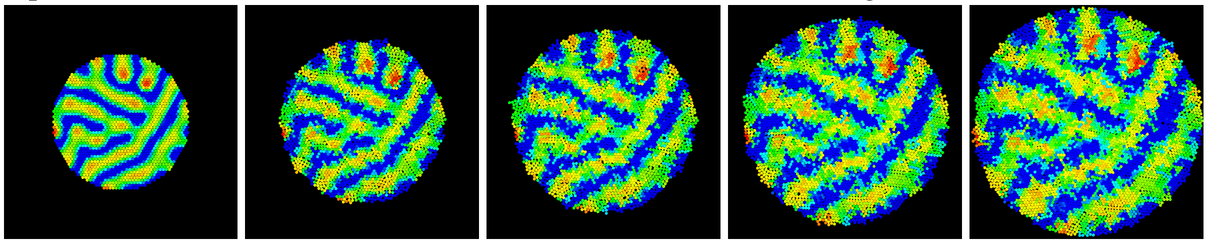
Figure 6.7: Uniformly growth inducing changes in the final pattern. The images depict iterations 2000, 6000, 9000, 12000 and 15000 (from left to right).



Source: own work.

However, we may stop diffusion as soon as the growth starts, to keep the concentrations constant. This is done in the experiment in Figure 6.8. As it can be seen, the division of cells introduces exact duplicates, which then add irregularities to the original pattern. On the other hand, if we only stopped the reaction, diffusion would blur the boundaries between concentration levels, making the pattern fade away. The experiment file is shown in Listing A.10.

Figure 6.8: Uniformly growing a pattern after making chemicals non-diffusible. The images depict iterations 2000, 6000, 9000, 12000 and 15000 (from left to right).



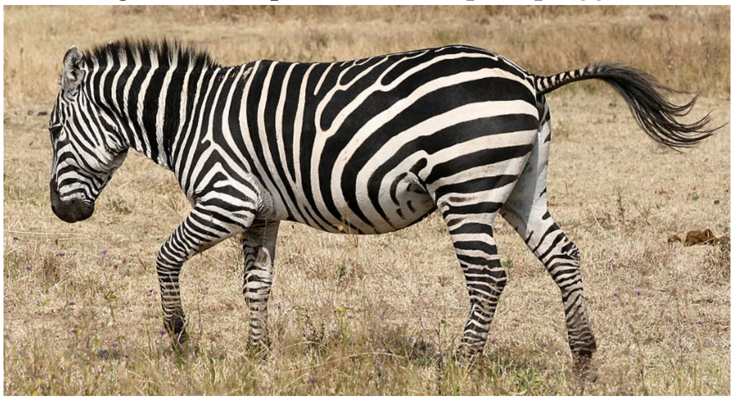
Source: own work.

## 6.6 Pattern enlargement

The results of the experiments shown in Section 6.5 clearly do not match the typical growth process of a fur coating on the back a living mammal as it ages, for example. Therefore, we have studied a particular problem: *after the hypothetical production of melanin-like pigmentation defines a skin pattern, how does it maintain its form during natural growth?*

We have expected that just coupling reaction, diffusion and cell division would lead to stable growing patterns, but the dynamic nature of standard RD equations always disrupts any initially established shape. In fact, we have tried adjusting the growth rates, the time step, the diffusion rates and the scale $s$, with no result[2]. Evidently, a more complex mechanism of pattern maintenance was missing.

Figure 6.9: A plains zebra (*Equus quagga*).



Source: photograph by Muhammad Mahdi Karim (Wikimedia Commons, GNU FDL 1.2).

We have devised several hypotheses of our own for the underlying pattern enlargement mechanism, focusing on the case where pattern borders would be maintained after the reaction has stopped. Note that we have resorted to building possible explanations based on the mechanisms already available in our model, and perhaps new ones. We thus avoided delving into the actual tissue regulation hypotheses from Biology, which depend on much more complex concepts at a microscale level. We have looked in the particular case of the plains zebra (*Equus quagga*), depicted in Figure 6.9. Although the adult individual has very well defined stripes, the most accepted theory says that stripes are formed in the very early days in the womb (BARD, 1981). In summary, the hypotheses were:

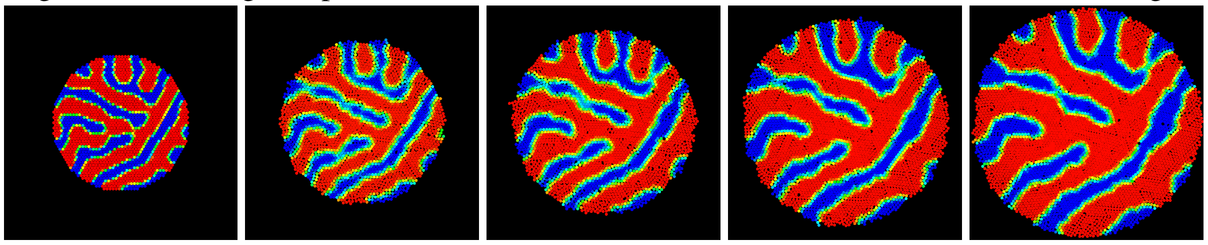
[2]Unsuccessful experiments are also made available, being organized in a folder of our software distribution.

1. Only the inner cells (in relation to black and white areas) would divide, making cell division localized. The side effect would be that borders could get irregular as the pattern grows.
2. Perhaps the pattern is defined on such a large scale (concerning individual cells) that actual division would not affect the final shape of borders. This hypothesis supposes very high diffusion rates.
3. Some mechanism of adhesion and repulsion between cells would keep the black/white border well defined. Perhaps cells with the same color would be attracted and repeal cells of the other color.
4. A conditioned division would create new cells only if the linearity of borders is not affected, maybe controlled by the production of a specific chemical along interfaces between black and white regions.
5. A flexible interface would be defined along borders, where cells establish and maintain torsion resistance along borders.
6. Some type of cell death mechanism is activated (either apoptosis or autophagy), where cells die if subjected to overcrowding or irregularities at the pattern borders.
7. Perhaps the actual division process needs to be changed, so that newborn cells would not get a copy of its parent's concentration, but an average of the concentration of nearby cells. That way inner regions would keep the same color whereas borders would get less irregular when mitosis occurs.
8. After the pattern is established, diffusion continues at a much slower rate and a regulation mechanism takes place, which smooths borders but keeps interiors stable.

Fortunately, before testing all those possibilities we have used once again the Occam's razor principle. By favoring simplicity, we have evaluated first the eighth hypothesis. And it has proven itself a viable maintenance mechanism, which we have since called *reinforcement*. That is, it is both simple and seems to be robust to global or local tissue growth.

Reinforcement works by mapping the concentrations from either $U$ or $V$ to another reagent, here called $R$, at a specific iteration and after the pattern has already established. We map the lowest concentrations for the source reagent into the zero level, and the highest from the source into the 2.0 level. This third reagent is also defined with a low diffusion rate. Then we activate two mutually exclusive rules to reinforce $R$ concentrations close to either 0.0 or 2.0. When the concentration for $R$ for a given cell is within $[0.1, 0.9]$, the first rule gradually decreases $R$ towards zero, whereas when the concentration is within $[1.1, 1.9]$, it increases it towards the upper limit. The result is that these rules oppose the diffusion effect, by keeping the high and low concentration regions mostly unchanged during growth. The net effect is that some minor detail is lost, but the overall pattern is maintained, as is shown in Figure 6.10. The same uniform growth as in the experiments of Section 6.5 is applied. The experiment file is shown in Listing A.11.

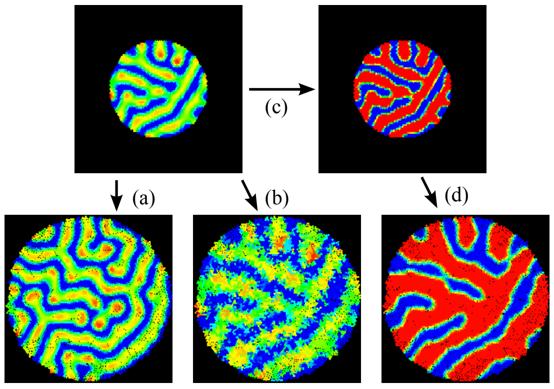
Figure 6.10: Uniformly growing a pattern using a chemical reinforcement mechanism on reagent $R$. The images depict iterations 2000, 6000, 9000, 12000 and 15000 (from left to right).



Source: own work.

It is interesting to note that the diffusion does not have practical effects inside the large red or blue area, but it actively smooths out borders, even when new cells are created. This smoothing is countered by the reinforcement mechanism, which tries to enhance the border, like a high-pass filter from image processing. Again, inside the large and mostly constant red and blue areas, the reinforcement mechanism has little effect. So most activity regarding a significant change in concentration occurs at borders between those regions. A comparison against the other uniform growth experiments is made in Figure 6.11.

Figure 6.11: Distinct behavior during tissue growth. Transition (a) shows the effect of domain growth, whereas (b) shows cell division with inactive reaction and diffusion. The copy of concentrations to a third chemical (c) makes possible to keep the overall pattern under uniform growth (d).



Source: own work.

As a final note, when using reinforcement we have only a fuzzy definition of low and high concentration levels for *R*, which tend to oscillate around 0.1 and 1.9, respectively. An absolute lower level could be achieved letting the concentration drop until reaching zero, as the simulation prohibits negative concentrations. The idea of adding to our model the concept of chemical saturation then occurred naturally, as a plausible way of limiting concentrations into an absolute higher level. The far-fetching implications of this simple new feature are described in Chapter 7.

## 6.7 Summary

In this chapter we have shown the results of a few experiments involving cell division, demonstrating that we can both reproduce prior results from the literature and produce new interesting patterns.

# 7 SATURATION

*"Our experience hitherto justifies us in trusting that Nature is the realization of the simplest that is mathematically conceivable."*
— Albert Einstein

Saturation is a basic concept from Chemistry, which when applied to a solution means the point where the concentration of a dissolved substance can no longer increase. Such feature, in fact, complies with the same requirements we are imposing to the other mechanisms of our model: simplicity and biological plausibility. Moreover, by adding saturation we can impose an upper limit, which adds symmetry to the already existing lower limit, zero. Finally, the implementation of saturation is trivial, being just a $\min(L_R, R)$ operation, where $L_R$ is the maximum concentration of reagent $R$.

From Chemistry we know that the point of saturation for a given solution depends on its temperature and pressure (SILBEY; ALBERTY; BAWENDI, 2004). However, it is reasonable to assume in our model that the saturation limit for a reagent is the same for all cells, which implies that it is defined globally and remains unchanged during the simulation.

Early versions of our model did not incorporate chemical saturation for the reagents. Some RD systems constrain chemical concentrations to be non-negative, and that was our only assumption from the beginning, as it makes sense in a real chemical process. We have since explored rules like the ones used for pattern enlargement and saw that imposing a limit concentration is a plausible and perhaps common mechanism in biological organisms.

We have found no references to the use of saturation limits within a reaction-diffusion system. Several well-known RD systems only impose limits on the reagent production, represented by the $f(u,v)$ and $g(u,v)$ terms in the Equations 2.1 and 2.2, like the ones described by (MURRAY, 2003) and (MIYAZAWA; OKAMOTO; KONDO, 2010). In fact, the few uses of the word *saturation* in RD systems seem to be related to imposing limits on the flux of diffusion, as in (KURGANOV; ROSENAU, 2005), not on the concentrations themselves.

So if chemical saturation is intrinsic to solutions, and RD systems mostly model idealized reactions between reagents present in the same solution, why does such mechanism has not been analyzed before? We suspect that most recent works on the literature, at least on the Biology and Computer Graphics fields, have resorted to employing just the classic RD models. Moreover, a $\min(L_R, R)$ operation lives "outside" the differential equations, and makes the formal mathematical analysis more complex.

In this chapter we will discuss the usage of chemical saturation in three different situations. First, we present how saturation can help reproducing classic cellular automata with our model. Then, we discuss a very simple mechanism to create novel patterns, dubbed reinforcement patterns. At last, we show a few very realistic patterns that emerged from the combination of saturation and reaction-diffusion.

## 7.1 Cellular automata

Although our focus has been on patterns created by reaction and diffusion, our model is also capable of running several types of cellular automata. Chemical saturation is used in these experiments because we typically need to force the concentration of a reagent to an absolute high level.

In fact, we understand that our model falls into the category of *continuous automata*, where valid states are no longer limited to be discrete numbers. Albeit there is no explicit support in our model for identifying who are the neighbors for a given cell (as only the number of nearby cells is available), we can measure and control the diffusion locally as a mechanism to evaluate neighborhoods.

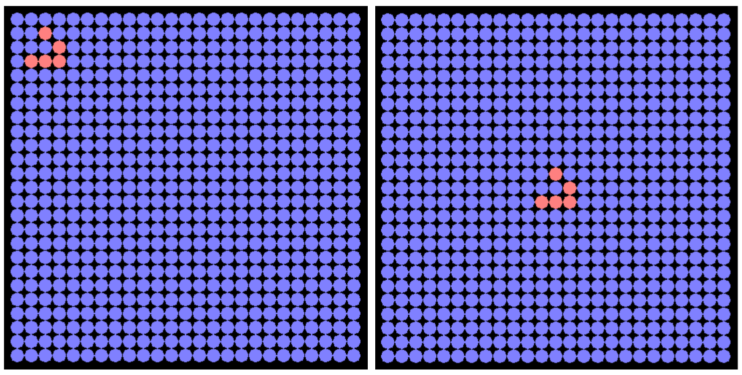
We have reproduced the classic Game of Life (GARDNER, 1970) cellular automaton. The prepattern for this experiment defines the well-known *glider* from Game of Life, made of alive cells, but any other initial configuration would behave correctly. It is a cyclic pattern, that slides one cell down and one cell to the right after eight iterations. After this cycle, it returns to its original form. Figure 7.1 shows the state at iteration zero (left), and then the state at iteration 80 (right).

We first establish a regular square grid where positions can be either “dead” or “alive” in the sense of the automaton. We map these two states to the reagent *R*, being 0 and 1, respectively. As there is no explicit mechanism to sense neighbors, we set the diffusion rate of *R* to be slow, and then apply conditions based on the concentration after just one iteration. For example, a dead position with just one alive neighbor will get a slight increase in its own concentration for *R*, whereas an alive position with four empty neighbors will have a proportional decrease in *R*. Depending on the current concentration of *R* we can either consume all of it, to reach a dead state, or produce more chemical, to reach an alive state. We have set the saturation level for *R* to be 1, by using the `limit` keyword.

There is, however, another issue to be handled: by definition, actions started by rules and diffusion occur simultaneously. Therefore, it is adequate to set reagent levels only at an even iteration, skip an iteration (so only the normal diffusion occurs) and then test chemical levels again. For this process, we have created an artificial “counter” reagent called *C*, which is alternatively set to 0 or 1, marking even and odd iterations. This chemical has zero diffusion rate.

Figure 7.1: Game of Life: starting state (left) and after 80 iterations (right).

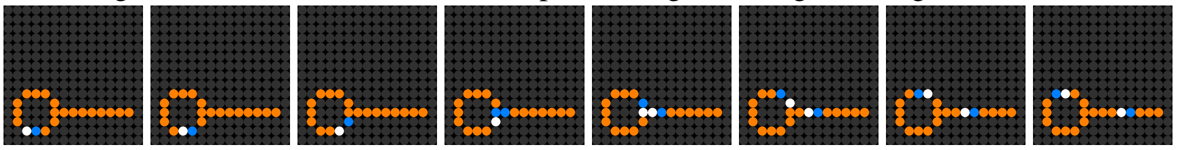
```
define chemical R limit 1
define chemical C

// -------- prepattern

use chemical R conc 0 diff 0.01
use chemical C conc 0 diff 0
create sqr_grid 25 25
set cell 526 chemical R conc 1
set cell 527 chemical R conc 1
set cell 528 chemical R conc 1
set cell 553 chemical R conc 1
set cell 577 chemical R conc 1

// -------- rules
// periodic counter
rule if C conc == 0 change C conc +1
rule if C conc == 1 change C conc -1
// kill all positions by default
rule if C conc == 1 change R conc -1
// a position with three neighbors becomes populated
rule if C conc == 1 and
rule if R conc in 0.025 0.034 change R conc +2
// a position with two or three neighbors survives to next generation
rule if C conc == 1 and
rule if R conc in 0.935 0.954 change R conc +2
stop at 160
```



Source: own work.

Another example is the Wireworld cellular automaton, which assumes four distinct states. We have reproduced it with a more general approach, by using three reagents and cycle of three (controlled by reagent *C*). Figure 7.2 shows the first few iterations of a clock-generating mechanism defined in the prepattern. The color scheme is the same used by the Golly package[1]. The experiment file is shown in Listing A.12.

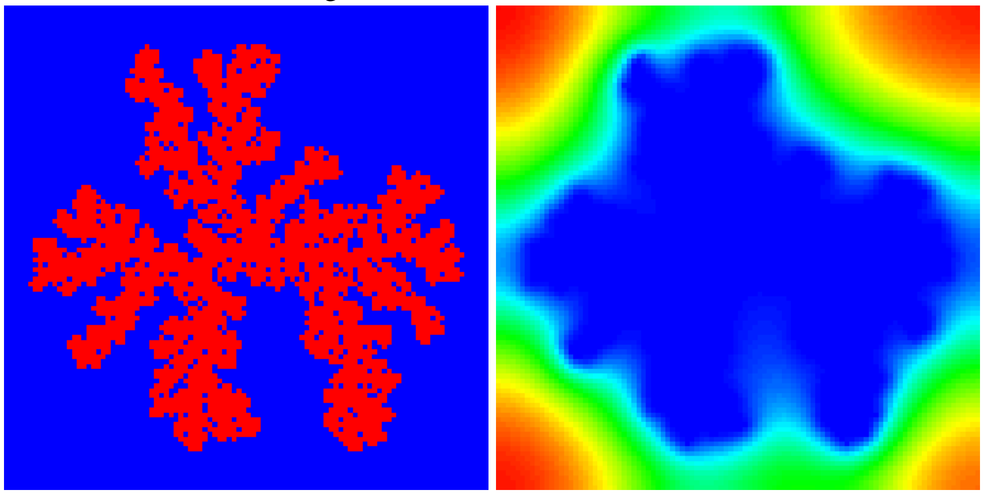
Figure 7.2: Wireworld automata implementing a clock-generating mechanism.



Source: own work.

A more complex experiment can be done by coupling a two-state automaton and an extra diffusing chemical. On the left of Figure 7.3 we show an approximation for a Laplacian Growth process, more specifically the Dielectric Breakdown Model (DBM) technique, described in (KIM et al., 2007). The basic idea is to grow a pattern from an initial group of active cells (shown in red), updating at each iteration a probability field. Only non-active blue cells that are adjacent neighbors of red cells can be activated, and this process is governed by the probability given by the concentration of a reagent that is continually consumed by active cells, shown in the right of Figure 7.3. Therefore, the higher probabilities are far from the active cells, where new fronts are expected to grow. The experiment file is shown in Listing A.13.

Figure 7.3: Approximate reproduction of Laplacian Growth using the DBM technique. On the right the active (red) and inactive (blue) cells. The probability field is shown on the right (blue is the lowest, whereas red is the highest).



Source: own work.

## 7.2 Reinforcement patterns

We have since experimented with the reinforcement mechanism described in Section 6.6, which gave rise to interesting patterns, which we called *reinforcement patterns*. The idea is simple: we initialize each cell in a regular lattice with a random concentration and a constant

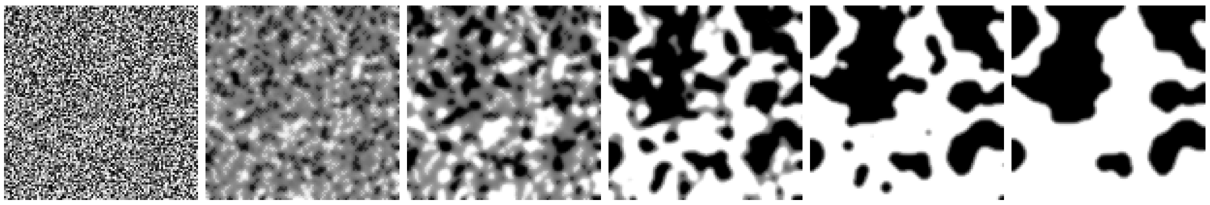
[1]<http://golly.sourceforge.net/>

diffusion rate. However, instead of applying a reaction rule that involves two chemicals, we use the reinforcement mechanism on just one reagent. If we adjust the parameters accordingly, a stable pattern emerges.

Although the reinforcement rules establish concentration intervals as predicates, we now can use saturation to simplify them. Indeed, by having as absolute lower bound the 0 concentration and as absolute upper bound a 1 concentration (for example), we can create mostly binary patterns with thin yet smooth borders. The experiment shown in Figure 7.4 starts with random concentrations for the single chemical *R* and achieve stability under 2700 iterations. The resulting concentration for *R* is then simply mapped to an interpolated grayscale map.

Figure 7.4: Stable reinforcement pattern using chemical saturation. The images depict iterations 0, 50, 100, 500 and 2700 (from left to right).

```
define chemical P limit 1

// -------- prepattern

use seed 2
use chemical P conc 0.5 dev 0.5 diff 0.01
create sqr_grid 100 100

// -------- rules

rule if P conc > 0.7 change P conc +0.01
rule if P conc < 0.3 change P conc -0.01

colormap select gradient
colormap slot 00 use color "black"
colormap slot 99 use color "white"
snap at 0 50 100 200 500 2700 then exit
```

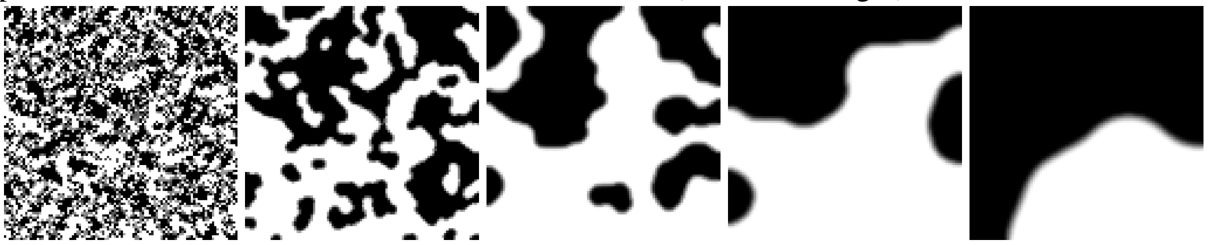


Source: own work.

If we change the diffusion rate, we achieve the same effect of adjusting both diffusion rates in a typical RD system: we affect the pattern scale, as shown in Figure 7.5.

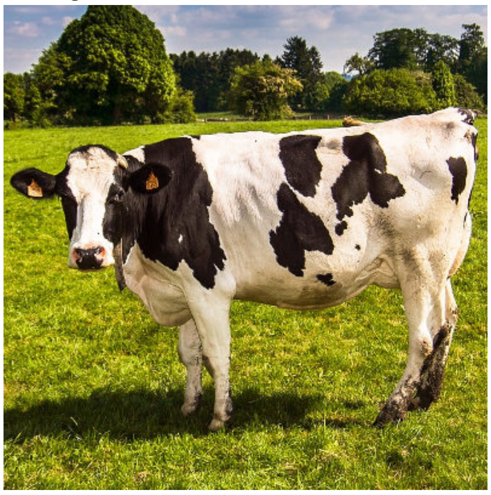
We are not aware of previous studies discussing this mechanism. The closest results are from (YOUNG, 1984), which creates black-and-white patterns and still rely on two chemicals that diffuse and react themselves. This work defines two types of cells, which are selected based on a threshold of one of the reagents. In fact, this work proposes a particular discretization of a traditional RD system, albeit generating stable binary patterns.

Figure 7.5: Stable reinforcement pattern using chemical saturation. The images depict the stable patterns for $D_R$ set to 0.02, 0.05, 0.01, 0.02 and 0.03 (from left to right).



Source: own work.

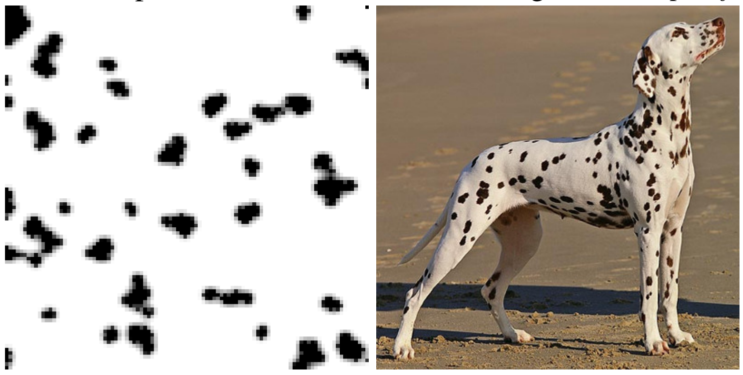
Figure 7.6: A cow (*Bos taurus*).



Source: photograph by Arnaud Liégeois (Pixabay, CC0).

Previous work on contrasting fur coatings (QUEIROZ, 2011) inspired us to find parameters that matched some mammal species. Some similarity from these results can be seen for a typical cow, as depicted in Figure 7.6. But a more striking similarity can be obtained for the irregular spots of Dalmatian dogs, as shown in Figure 7.7. The experiment file is shown in Listing A.14.

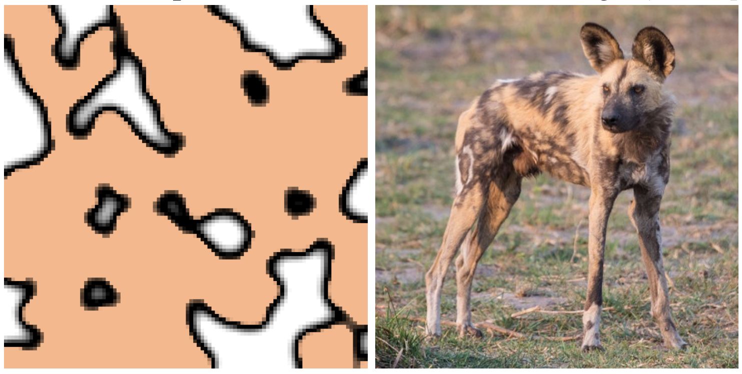
Figure 7.7: Reinforcement pattern (left) and Dalmatian dog (*Canis lupus familiaris*) (right).



Source: (left) own work and (right) photograph by skeeze (Pixabay, CC0).

We can further change the parameters, for example increasing the saturation limit to 3, to get larger transition areas, like the result shown in Figure 7.8. Still, a stable pattern is generated after 2100 iterations. The experiment file is shown in Listing A.15.

Figure 7.8: Reinforcement pattern (left) and African wild dog (*Lycaon pictus*) (right).

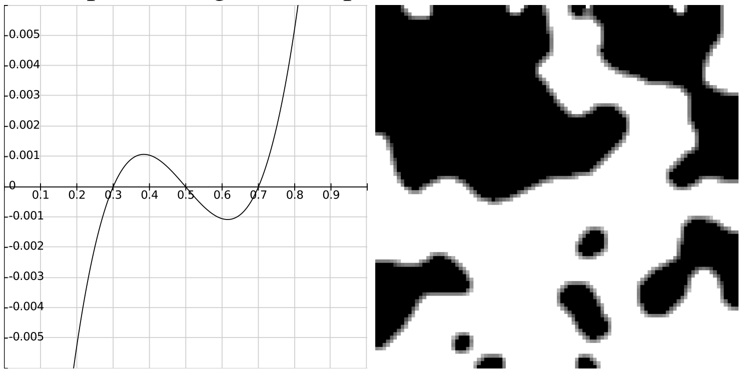


Source: (left) own work and (right) photograph by tonyo_au (Pixabay, CC0).

As a final experiment, we prototyped a new reaction involving just a single reagent $R$. We were trying to mimic the behavior of the two reinforcement rules by a single reaction term, akin to standard reaction-diffusion. The idea is that such equation forces to 0 concentrations below 0.5, whereas concentrations above 0.5 will be pushed toward the saturation of 1. We so far only tested a single formulation, which is shown in Equations 7.1. We added the explicit $s$ parameter in similarity to the scale factor for the Turing equations. A search into the equation parameter space for $a$, $b$ and $c$ found a few triplets that indeed create stable patterns. Figure 7.9 shows a function plot for the reaction term, along with the associated generated pattern. We used the same grid, initial concentrations and diffusion rate as in Figure 7.4.

$$\frac{\partial r}{\partial t} = s(r-a)(r-b)(r-c) + D_r \nabla^2 r \tag{7.1}$$

Figure 7.9: Plotting of reinforcement reaction term as $y = s(x-a)(x-b)(x-c)$ (left) and corresponding generated pattern (right), with parameters $s = 0.35$, $a = 0.3$, $b = 0.5$ and $c = 0.7$.



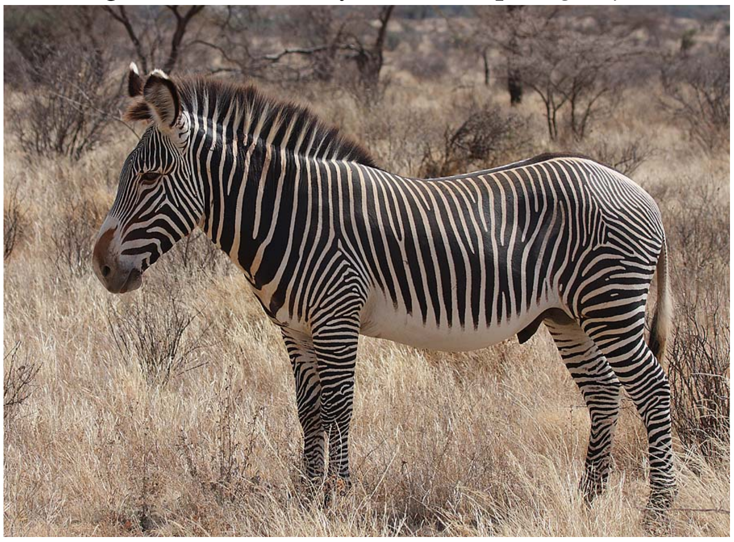
Source: own work.

We can see that a similar result can be achieved, replacing two discrete rules for a continuous reaction. We think there is an interesting venue for further research on this type of patterns. And perhaps a mathematical morphology analysis would be able to assess how statistically close the generated patterns are from real fur coatings.

## 7.3 Saturated reaction-diffusion

As remarked in Section 6.6, we have analyzed many photos of zebras, including different species, like the Grévy's zebra (*Equus grevyi*, in Figure 7.10). The pigmentation pattern is mostly binary, but for us there are three puzzling aspects regarding pattern formation: the very accentuate difference in stripe width, the distinct orientations of stripes along the body and the small and intricate details in specific regions.

Figure 7.10: A Grévy's zebra (*Equus grevyi*).



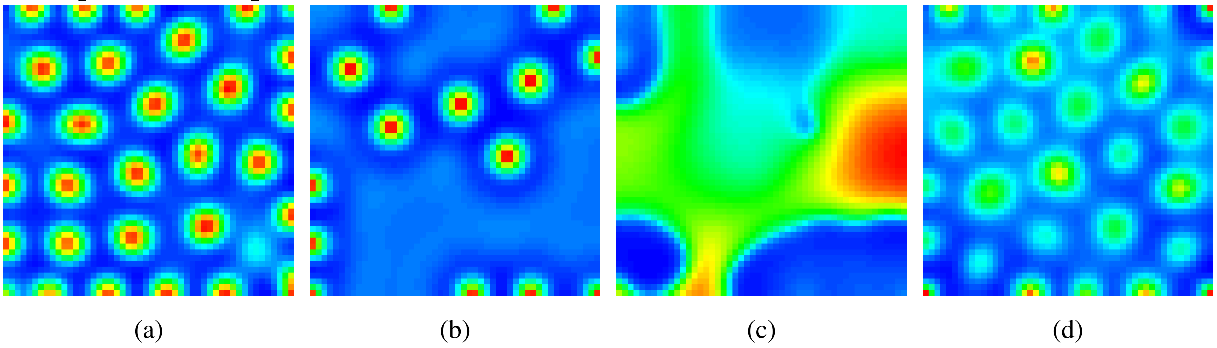
Source: photograph by Steve Garvie (Wikimedia Commons, CC BY-SA 2.0).

Several hypotheses for the variation of stripe widths and orientations have been available since early works of (BARD, 1977) and (MEINHARDT, 1982), and most explanations relate to the distinct times where the basic pattern is formed on the embryo skin and the different growth rates for parts of its body until adulthood. However, even recent work like (GRAVAN; LAHOZ-BELTRA, 2004) has not provided a compelling variation for the generation of stripes.

Moreover, our analysis of minor features in the zebra skin strongly suggests that the pattern definition mechanism is still active during growth, which would explain the emerging small stripes along the dorsal region and the emerging white stripes in the zebra back, as can be seen in Figure 7.10. We have made several experiments trying to reproduce those features, and just the growth control with anisotropic diffusion is not enough to reproduce them. We were convinced that the missing puzzle piece is a mechanism that keeps reaction and diffusion active while still maintaining borders well defined.

We have proposed the reinforcement mechanism as described earlier, where it transferred the pattern to another reagent before growth, in fact stopping the reaction. But the interesting results obtained for reinforcement patterns in Section 7.2 suggested another venue for experimentation: *what would happen if we couple reaction with reinforcement rules?* The addition of reinforcement rules and just explicit production or consumption actions indeed introduced spatial heterogeneity to the Turing patterns, and sometimes led to chaotic configurations. The Figure 7.11 shows a basic spot pattern and the variations induced. The (d) pattern even got spots with oscillatory behavior.

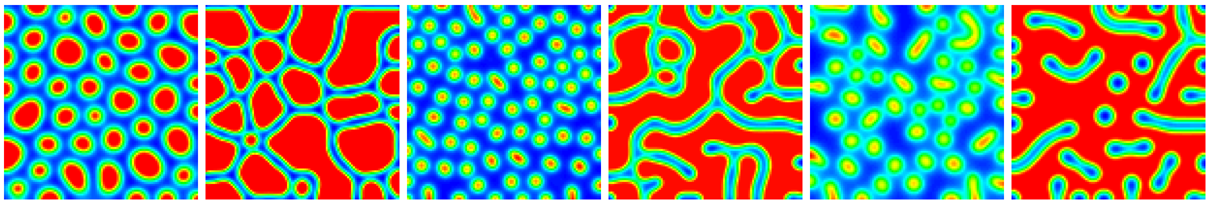
Figure 7.11: Stable spot pattern (a) perturbed by (b) reinforcement, (c) explicit production and (d) explicit consumption.



Source: own work.

We then asked a second question: *what would happen if we couple reaction with chemical saturation?* Some results obtained are illustrated in Figure 7.12. In short, the classic Turing equations gain very compelling and diverse behavior when saturation limits are imposed to one or more reagents. The remaining of this chapter presents the patterns that are now possible with the use of saturation.

Figure 7.12: A few patterns created by adding saturation limits to one or two reagents during reaction and diffusion.

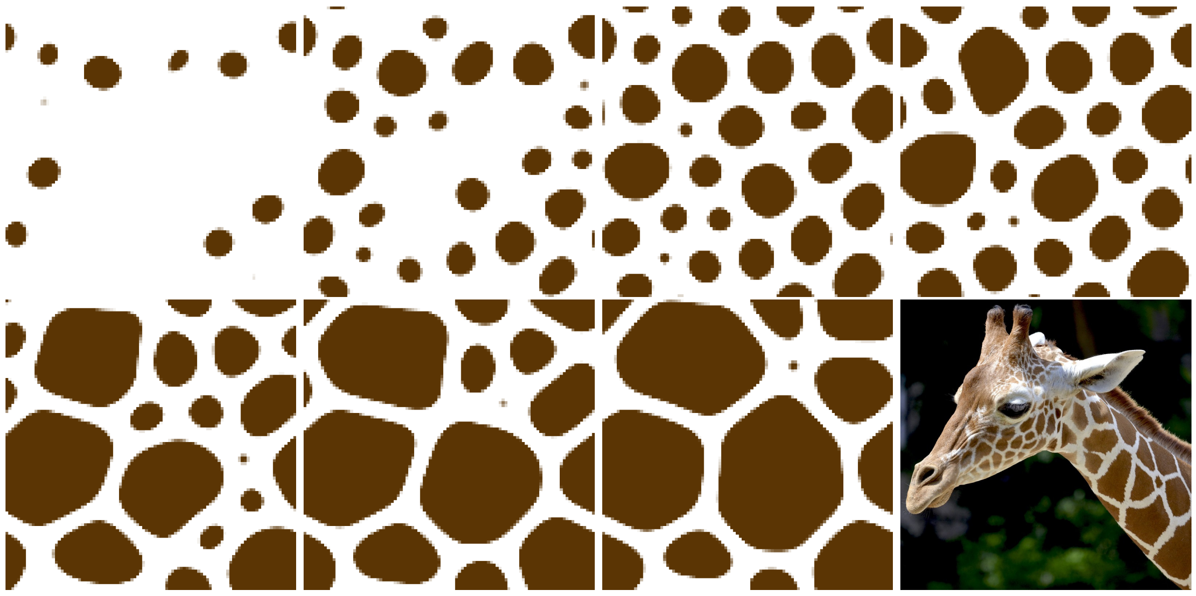


Source: own work.

The experiment shown in Figure 7.13 exhibits the emergence of a stable giraffe pattern from a random amount of initial concentrations for $U$ and $V$ chemicals. The RD system first produces irregularly spaced spots with similar size. Gradually, some spots increase and absorb smaller spots, creating larger regions, and then moving to a stable configuration. For this experiment we have used the **wrap** keyword, enabling a toroidal topology: cells at lattice borders are also adjacent to the ones on opposite sides.

Figure 7.13: Reaction-diffusion where both $U$ and $V$ reagents achieve saturation, with corresponding giraffe (*Giraffa reticulata*) pattern.

```
define chemical U limit 6.3
define chemical V limit 6.3

// -------- prepattern

use chemical U conc 4 dev 3 diff 0.04
use chemical V conc 4 dev 3 diff 0.01
create sqr_grid 100 100 wrap

// -------- rules

rule always react U V scale 0.005 turing alpha 16 beta 12
colormap select gradient
colormap slot 60 use color "white"
colormap slot 75 use color "5b3504"
stop at 38000
```

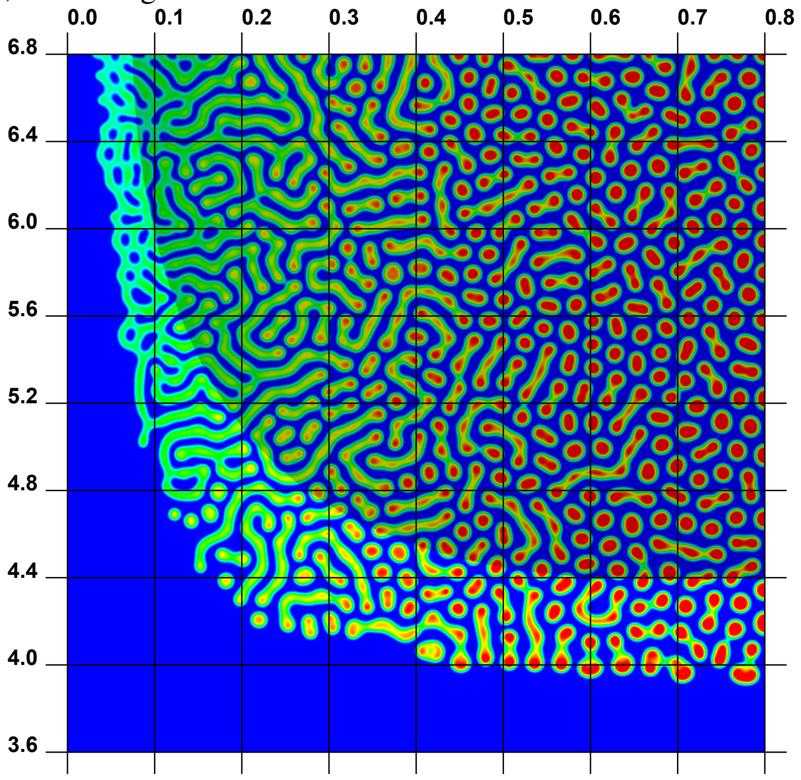


Source: own work and photograph by Andrea Bohl (Pixabay, CC0).

Particularly striking are the dynamics provided by saturation: when one reagent hits its upper concentration limit, the other one is usually forced to zero. The use of saturation tends to create large areas of zero concentration and others of maximum concentration. This behavior is exactly what we were previously trying to achieve for pattern growing. That means that saturated RD is resilient to domain growth, and now makes possible the creation of large areas, thus breaking the fixed wavelength from the standard Turing equations.

When exploring the dynamics of the patterns generated by saturated reaction-diffusion, we found that the most visually interesting and capable of generating realistic biological patterns came from imposing a limit to either $U$ or the same limit to both $U$ and $V$. Figure 7.14 shows the parameter map for the situation where only $U$ is limited.

Figure 7.14: Parameter map showing $V$, where $D_U$ varies the horizontal axis, and saturation limit $L_U$ varies in the vertical axis. In the darkened area the concentrations fluctuate below the upper limit, so there is no difference from the standard RD behavior. No steady patterns appear in the blue areas, as the reagents tend to constant concentration.



Source: own work.

We have run a few thousand simulations with different combinations of parameters for saturation limits (either for $U$, $V$ or for both at the same time), diffusion rate for $U$, initial random seed scale $s$. A particularly unusual set of parameters was later identified to match one species of nudibranch (Hypselodoris iacula), shown in Figure 7.15. One of the distinct features of this pattern are the small dots, which seldom occur but are very stable and do not fade away.

We have also experimented with patterns with local polarization. Figure 7.16 exhibits an experiment where vertically oriented cells are set in the top section of the pattern, whereas cells at the bottom do not have polarity. The distinction between parts is made by simply defining distinct diffusion rates at the prepattern. Note that the lower pattern generates spots of different sizes, as result of saturation. The inspiration came from the poison dart frog (*Ranitomeya* *amazonica*). The image on the left is generated by using the interpolation algorithm discussed in Section 4.6, which was run after the simulation stopped. The colors for all cells are mapped from the same concentration of the $U$ reagent. The experiment file is shown in Listing A.16.

Figure 7.15: Experiment reproducing the pigmentation of a nudibranch (*Hypselodoris iacula*) (left) and a living individual (right).

```
define chemical U limit 5.2
define chemical V

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.08
use chemical V conc 4 dev 2 diff 0.01
create sqr_grid 100 100 wrap

// -------- rules

rule always react U V scale 0.02 turing alpha 16 beta 12
colormap select gradient
colormap slot 20 use color "white"
colormap slot 50 use color "eaaa7a"
stop at 3000
```

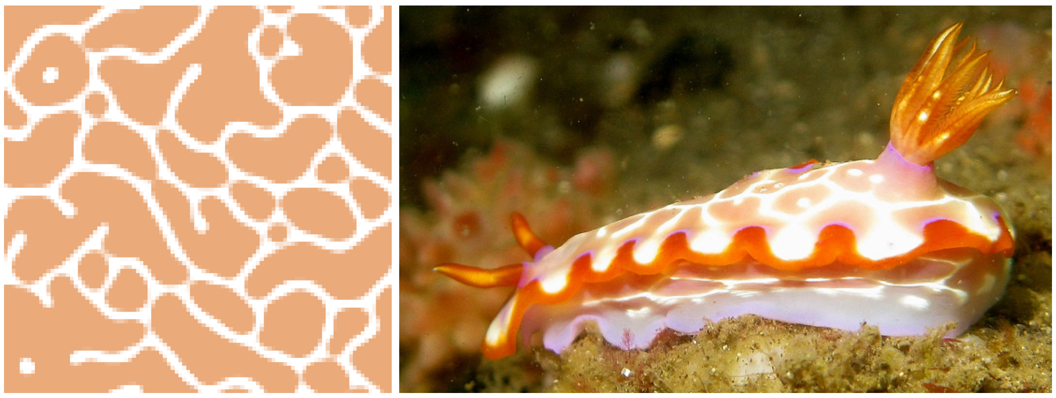
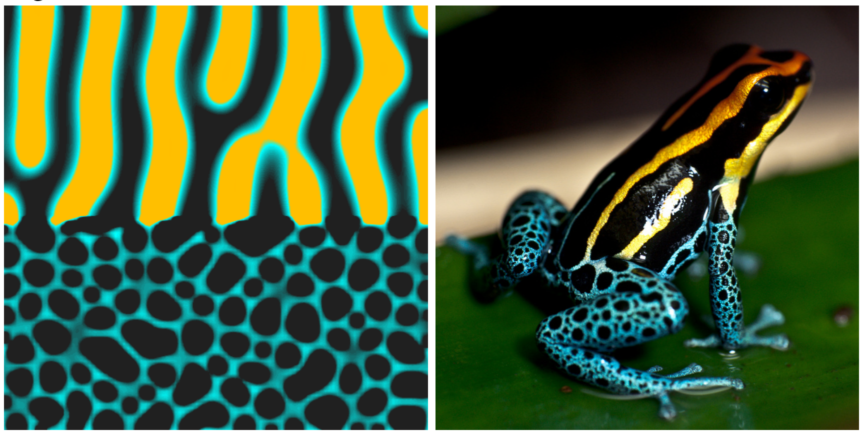
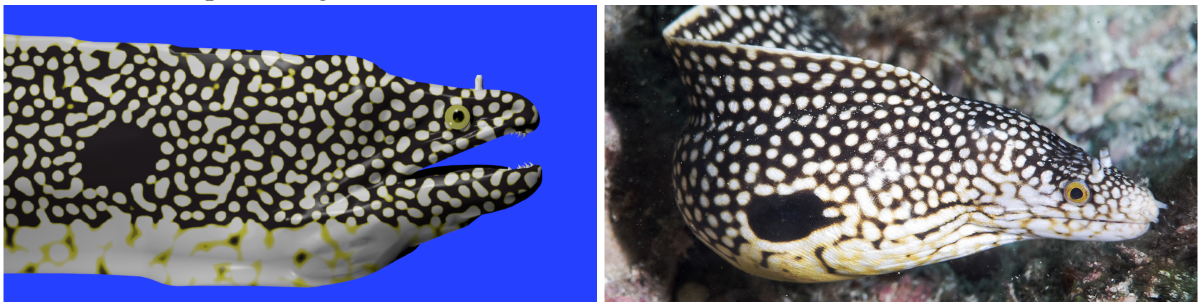

Source: photograph by Steve Childs (Wikimedia Commons, CC BY 2.0).

Figure 7.16: Poison dart frog (*Ranitomeya amazonica*) interpolated texture (left) and a photo of an individual (right).


Source: own work (left) and photograph by Sascha Gebhardt (Flickr, CC BY-NC 2.0).

In Figure 7.17 we show the recreation of the pattern from a specific moray eel species, *Muraena melanotis*, mapped onto a 3D model.

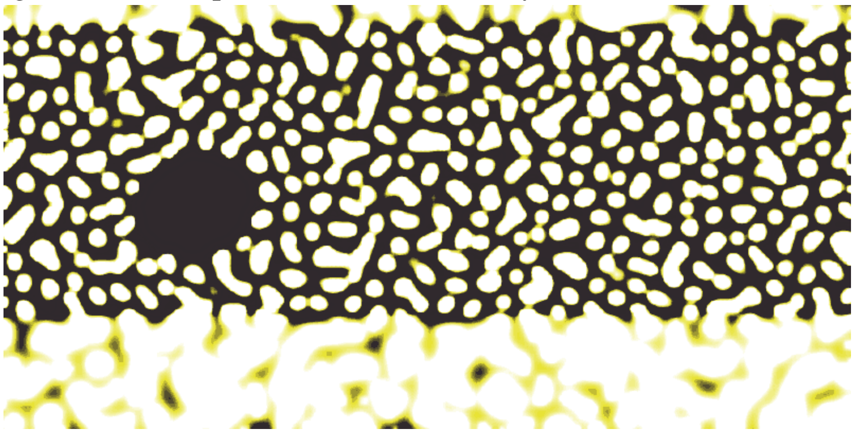
Figure 7.17: The moray eel (*Muraena melanotis*): simulated pattern mapped onto a 3D model (left) and actual photo (right).



Source: own work (left) and photograph by Richard Bowes (Flickr, used under permission).

We again employed saturated RD, limiting $U$ and $V$. For the distinct parts of the body we simply set different diffusion rates for $U$. The closeness of cells ensures the continuity of concentrations in the interface of such parts. For the distinctive black spot, we created an approximate elliptical gradient, based on two producer cells, and let nearby cells affected by this gradient to continually produce $V$, which prevented the emergence of white spots. The final texture is shown in Figure 7.18 and the corresponding experiment file in Listing A.17.

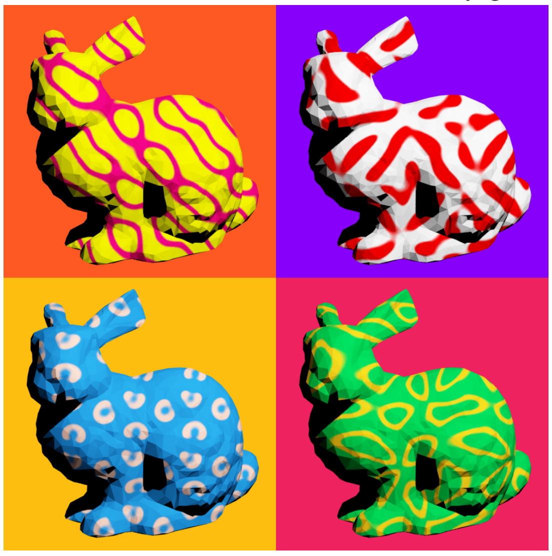
Figure 7.18: Interpolated texture for moray eel (*Muraena melanotis*).



Source: own work.

As a final example, we have selected four randomly generated experiments involving reaction, saturation, gradients, polarization and anisotropy. We manually assigned bright colormaps to reagent concentrations, exporting the resulting patterns as high-quality textures. Those textures were then mapped into low-resolution versions of the Stanford Bunny. The result is shown in Figure 7.19.

Figure 7.19: “Saturated Bunnies”, made from randomly generated experiments.



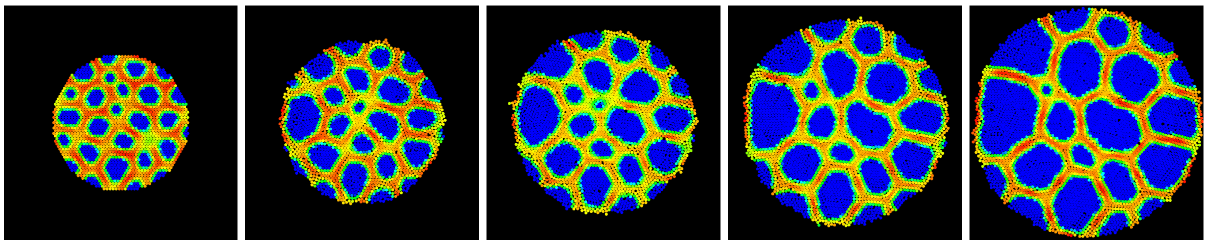
Source: own work.

### 7.4 Growth and saturated reaction-diffusion

If we add cell division, saturated RD is still able to maintain (and perhaps evolve) an initial shape. An experiment similar to the ones from Section 6.5 is shown in 7.20.

Figure 7.20: Uniformly growing a pattern using a chemical saturation on reagent *R*. The images depict iterations 2000, 6000, 9000, 12000 and 15000 (from left to right).

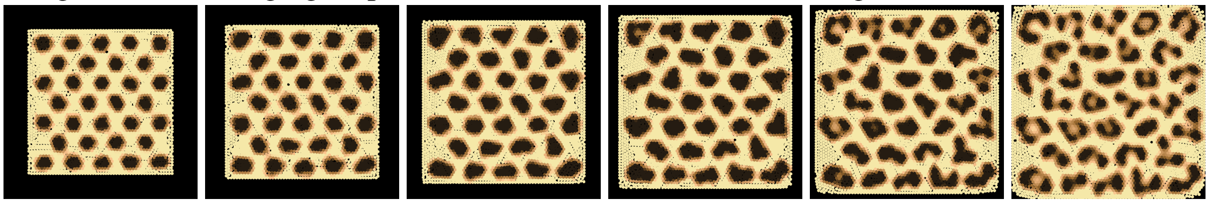
```
define chemical V
define chemical U limit 6.3
define division_limit 6

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.04
use chemical V conc 4 dev 2 diff 0.01
create hex_circle 45

// -------- rules

rule always react U V scale 0.01 turing alpha 16 beta 12
rule from 2000 until 15000 probability 0.0001 divide direction 0 dev 180
snap at 2000 6000 9000 12000 15000 then exit
zoom level 1.303
```



Source: own work.

We also have confirmed the conjecture from (KONDO; SHIROTA, 2009) that leopard rosettes would spontaneously appear from the growth of the tissue, albeit with a simpler mechanism and more accurate results, as shown in Figure 7.21. The experiment file is shown in Listing A.18.

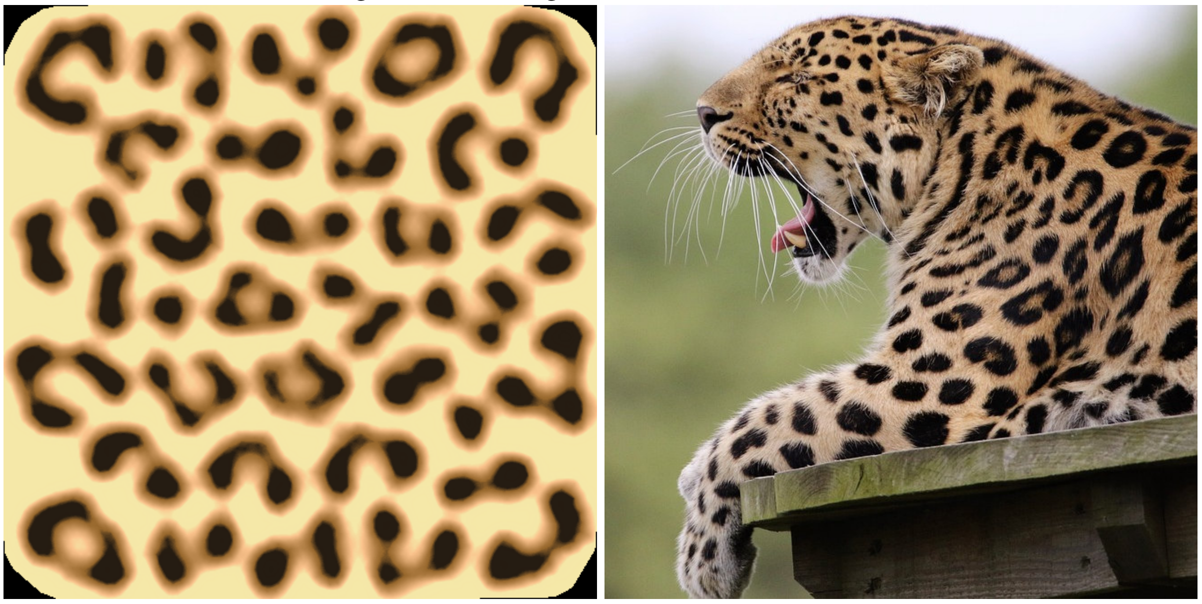
Figure 7.21: Emerging leopard rosettes from combined uniform growth and saturated RD.



Source: own work.

For the leopard experiment we have coupled uniform growth and saturated RD, imposing a limit only to $U$. To make the growth rate consistent over the simulated tissue, we further constrained the pattern into a packed domain, and explicitly seeded the prepattern with uniformly spaced cells. Note that previous results for both giraffe and rosettes were only possible with cascade RD processes or more specialized computations (TURK, 1991; WITKIN; KASS, 1991; WALTER; FOURNIER; MENEVAUX, 2001; LIU; LIAW; MAINI, 2006). A comparison from the final high-quality interpolation and the coating from an individual is shown in Figure 7.22

Figure 7.22: Simulated leopard (*Panthera pardus*) rosettes from combined uniform growth and saturated RD (left) and living individual (right).



Source: own work and photograph by Kdsphotos (Pixabay, CC0).

## 7.5 Summary

In this chapter we have shown a few results of the addition of chemical saturation to our model. It seems that this simple feature opens many possible venues for further research. First, we were able to reproduce a few classic cellular automata, showing that a general approach is possible for representing many other ones. Then have closely matched many distinct natural patterns with either reinforcement mechanisms or by usage of saturated reaction-diffusion. As we employed only biologically plausible mechanisms recreating them, we believe in the probable existence of a saturation-like phenomenon in the real pattern formation processes.

# 8 CONCLUSIONS

*"We wish to pursue the truth no matter where it leads – but to find the truth, we need imagination and skepticism both."*
— Carl Sagan

We have presented a parsimonious yet elegant simulation model, trying to mimic the fundamental workings of growing organisms, where the complexity of a few realistic biological patterns emerged from the combination of reaction, diffusion, chemical saturation and the application of simple rules. Our fundamental emphasis is on the *mechanochemical* processes, mainly chemical reaction inside cells, diffusion through their membranes, simplified division and damped collision.

We tried to minimize the complexity of both the feature set and the underlying implementation. The limited features, even more restricted by biological plausibility, made possible the systematic exploration of a large group of experiments, either by varying a few parameters at once or focusing in a few distinct setups. The results of this approach were presented in Chapters 5, 6 and 7. A summary of this research has also been published as a technical conference paper at the Graphics Interface 2017 (MALHEIROS; WALTER, 2017).

## 8.1 Contributions

Here are the main contributions of our work:

1. We have proposed a simple yet expressive simulation model, which, by implementing rule-based biological mechanisms at a mesoscale cellular level, can provide the building blocks to several distinct pattern designs.
2. We have made freely available a reference implementation for our model, for which we can highlight real-time visualization, ease of experimentation, use of efficient algorithms and high-quality output formats.
3. We have built a simple and reusable simulation framework, that is ready to be incorporated into other applications.
4. We have discussed the problem of pattern enlargement and proposed a reinforcement mechanism that can maintain its features during uniform growth.
5. We have shown that a broad range of traditional cellular automata can be reproduced by

our model, including more complex mechanisms like the Laplacian Growth.

6. We have described a set of reinforcement patterns, being either generated by discrete rules or by a single cubic equation, which gave patterns similar to cows and Dalmatian dogs.
7. We have analyzed the effect of adding chemical saturation to the standard reaction-diffusion process and shown that it works as another reinforcement mechanism, keeping the pattern during growth.
8. We have been able to synthesize patterns using chemical saturation that closely matched the pigmentation patterns of several real species, like giraffe, poison frog, nudibranch, moray eel and leopard.

## 8.2 Future work

We understand that there is still much to be explored based on our model. In this section, we give a brief outline of the future research possibilities that were hinted during our work.

- **Explore the effect of cell polarity:** we plan to further evaluate the interrelation between growth and definition of cell polarity, as a mechanism for the emergence of oriented patterns. For example, simulation of the striped pattern of the common cuttlefish (*Sepia officinalis*) or the intricate alternating stripes of the juvenile emperor angelfish (*Pomacanthus imperator*).
- **Reproduction of full-body patterns:** we think that is now possible to accurately reproduce full-body coating patterns for some species, like the zebra. For that, a precise reproduction of the growth rates of distinct parts of a growing animal must be constructed, from embryo to adulthood.
- **Other biological mechanisms:** a few processes were implemented and then dropped because of limited expressibility or performance concerns: cells with different radius, cell movement, cell-to-cell adhesion (WALTER; FOURNIER; REIMERS, 1998) and forced diffusion. Others were not considered out of added complexity, like more complex cell signaling and non-empty extracellular space. These are still possibilities to be explored later.
- **Explore the Gray-Scott equations:** although we have only implemented the classic Turing equations, it is straightforward to add support for other RD systems. In particular, the Gray-Scott equations (PEARSON, 1993) provide very complex pattern dynamics, where a few parameter combinations seem to exhibit a behavior similar to what we have

got with chemical saturation[1].

- **Other RD equations:** we have yet to analyze other representative RD systems, like the activator-inhibitor and substrate-depletion from (MEINHARDT, 2009) or the linear model from (MIYAZAWA; OKAMOTO; KONDO, 2010). We also think that oscillatory RD systems could be useful to generate moving waves of reagents, like the Brusselator (PENA; PEREZ-GARCIA, 2001) or the Oregonator (RINZEL; TROY, 1982) models, which could be useful to build synchronized growing fronts of cells. Perhaps the advection equation could also be incorporated into an action rule, so another type of chemical signal propagation would be available, like in (KIM; LIN, 2007).
- **Further explore reinforcement patterns:** we think there is a lot more to explore regarding the mechanism described in Section 7.2. For example, other types of equations could be evaluated, and a complete map of the parameter space for such patterns could be made.
- **Flexible structures and transport mechanisms:** we have already implemented a more complex type of cellular adhesion, envisaging flexible junctions between cells, but have since discarded it as a very specific feature. When pursuing more form-oriented features, such type of adhesion would be very important. And perhaps, together with explicit chemical transport mechanisms, circulatory and venation patterns could be built by an expansion of our model.
- **More rule predicates:** provide finer ways to detect cell compression, proximity with tissue borders and alignment with other cells, which could enhance simulation possibilities.
- **Enhanced growth control:** we think that the current division limit mechanism is useful but restrictive. In a few situations, a finer control would be needed, mainly to reduce pattern irregularities when performing uniform growth. This could be done by preventing newborn cells from dividing too early, for example.
- **Pattern catalog**: as a concerted effort to map interesting combinations of parameters, we have run several thousand automatically generated experiments, which gave rise to a large, yet incomplete, pattern catalog. An interesting future work would be to organize these results and explore other mechanisms of systematic exploration of experiments. We could even apply the concept of experiment mutation or genetic algorithms to drive such exploration.
- **Simulation over a 3D mesh:** although our focus in this work was restricted to 2D, we think that it is straightforward to extend the simulation to run over a 3D mesh. In fact,

[1]<http://mrob.com/pub/comp/xmorphia/>

we have already prototyped a simple feature that enables cells to be simulated over a texture chart, where cells at segment borders can be mirrored at other locations, providing chemical continuity. This feature, however, was not mature enough and therefore not included in the final implementation.

- **GPU implementation:** the use of efficiently parallelizable algorithms have been a major concern since the inception of this work, but a full implementation and its validation is still to be done. We estimate that with minor adjusts the simulation loop can run entirely on a GPU. By using spatial sorting as the underlying NNS algorithm, we think that the number of simulated cells is bound only by the available GPU memory.
- **Online web version:** we think a complete in-browser simulation is possible, using the nascent WebAssembly[2] standard. Coupled with a support website, it would be interesting to have users testing out ideas and sharing online their experiments. We prototyped a Javascript-only implementation of the reaction and diffusion on a fixed square lattice, using the P5.js framework[3], and the performance is already acceptable.
- **Artist-oriented interface:** as the current implementation is modular, it could be easily embedded into a more artistic-oriented application, making it possible the creation of patterns by an interactive arrangement of higher-level primitives, inspired on the combination of the building blocks described in Chapter 5 with growth strategies. In fact, we envision our current simulation loop as a virtual machine, whose bytecode is expressed by the simple commands of an experiment. Therefore the equivalent of a compiler could be devised to translate a higher-level representation (or even purely graphical diagrams) into an executable experiment.
- **Inverse problem:** at last, we think the inverse problem is tractable for some of the patterns shown, like cows or giraffes. That is, finding the exact parameters that generate a single target pattern. As we have discussed, most of the overall result is controlled by rules and global parameters like saturation limits or diffusion rates, whereas the local detail is mostly driven by the initial reagent concentrations. We speculate that a higher-level analysis can infer the matching category for a pattern, based on a pre-computed catalog, and then resort to optimization strategies to find an adequate prepattern.

[2]<http://webassembly.org/>
[3]<http://p5js.org/>

## 8.3 Final remarks

For us, it is important to report that most of our research was driven by an initial insight, about how much apparent complexity can be derived from simple iterative processes. And as a direct corollary, that we should first exhaust all simple explanations before moving to more complex ones.

During this effort, it became evident that some prior works were limited by their own choices of over formalism or too much distance from basic biologic analogies, even though trying to match organic patterns and forms. The more we read about related topics from Computer Graphics and Developmental Biology, the more we perceived that *ad hoc* solutions and unnecessary technical complexity took over the necessary search for simplicity, that is, for an underlying explanation for the natural diversity. And still, here and there were points that reinforced our expectations: the everlasting impact of Turing's RD equations, the speculations about the ubiquity of cellular automata from Wolfram, the adaptability and resilience of reaction-diffusion over arbitrary cell arrangements, the little-known behavior of anisotropy when affecting just one reagent, or the fact that the simplest hypothesis about pattern enlargement proved to be the right one.

In fact, when the first *in silico* results were generated from the coupling of reaction, diffusion and saturation, we received them with a mixture of amazement and perplexity: it did work as intended and wished for, but it was also very simple, almost obvious. And more importantly, it still was biologically plausible. Moreover, the patterns generated later, like the giraffe, the nudibranch, the moray eel or the leopard, signaled that, perhaps, our underlying model has a strong probability of being the correct explanation for many real patterns, functioning both as a simplification concept and as a prediction tool.

We envision our work as just one small step, an individual contribution to the intricate network that is Applied Computing. Computer Graphics may use our results for the automatic generation of realistic skin textures, or Experimental Biology may find underlying biochemical regulatory mechanisms that operate with similar effects as chemical saturation. But by now we have succeeded in our quest: we fulfilled our scientific intuition, answered some interesting questions, created pretty pictures and ended up with many more questions than when we started.

# REFERENCES

ABELSON, H. et al. Amorphous computing. **Communications of the ACM**, ACM, v. 43, n. 5, p. 74–82, 2000.

AGARWAL, P. The cell programming language. **Artificial Life**, MIT Press, v. 2, n. 1, p. 37–77, 1994.

AMAT, F. et al. Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data. **Nature methods**, Nature Publishing Group, 2014.

ANASTACIO, F.; PRUSINKIEWICZ, P.; SOUSA, M. C. Sketch-based parameterization of l-systems using illustration-inspired construction lines. In: **Proceedings of the Fifth Eurographics Conference on Sketch-Based Interfaces and Modeling**. Aire-la-Ville, Switzerland: Eurographics Association, 2008. p. 119–126.

BALL, P. **The self-made tapestry: pattern formation in nature**. Oxford, United Kingdom: Oxford University Press, 2001.

BALL, P. Forging patterns and making waves from biology to geology: a commentary on turing (1952) 'the chemical basis of morphogenesis'. **Phil. Trans. R. Soc. B**, The Royal Society, v. 370, n. 1666, p. 20140218, 2015.

BARD, J. A unity underlying the different zebra striping patterns. **Journal of Zoology**, Wiley Online Library, v. 183, n. 4, p. 527–539, 1977.

BARD, J. B. A model for generating aspects of zebra and other mammalian coat patterns. **Journal of Theoretical Biology**, v. 93, n. 2, p. 363 – 385, 1981. ISSN 0022-5193.

CAICEDO-CARVAJAL, C. E.; SHINBROT, T. In silico zebrafish pattern formation. **Developmental biology**, Elsevier, v. 315, n. 2, p. 397–403, 2008.

CARVALHO, L.; ANDRADE, M.; VELHO, L. Generating textures on surfaces with reaction-diffusion systems in the gpu. In: **Proceedings of NVIDIA GCDF - GPU Computing Developer Forum**. [S.l.: s.n.], 2012.

CHI, M.-T.; LIU, W.-C.; HSU, S.-H. Image stylization using anisotropic reaction diffusion. **The Visual Computer**, Springer, v. 32, n. 12, p. 1549–1561, 2016.

CHRISTLEY, S. et al. Integrative multicellular biological modeling: a case study of 3d epidermal development using gpu algorithms. **BMC systems biology**, BioMed Central Ltd, v. 4, n. 1, p. 107, 2010.

CICKOVSKI, T. M. et al. A framework for three-dimensional simulation of morphogenesis. **IEEE/ACM Transactions on Computational Biology and Bioinformatics**, IEEE, v. 2, n. 4, p. 273–288, 2005.

CLAVET, S.; BEAUDOIN, P.; POULIN, P. Particle-based viscoelastic fluid simulation. In: ACM. **Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation**. [S.l.], 2005. p. 219–228.

COORE, D.; NAGPAL, R. Implementing reaction-diffusion on an amorphous computer. In: **Proceedings of 1998 MIT Student Workshop on High-Performance Computing in Science and Engineering**. [S.l.: s.n.], 1998. p. 189–192.

DEROSE, T.; KASS, M.; TRUONG, T. Subdivision surfaces in character animation. In: ACM. **Proceedings of the 25th annual conference on Computer graphics and interactive techniques**. [S.l.], 1998. p. 85–94.

DESBENOIT, B.; GALIN, E.; AKKOUCHE, S. Simulating and modeling lichen growth. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 2004. v. 23, n. 3, p. 341–350.

DEUSSEN, O. et al. Realistic modeling and rendering of plant ecosystems. In: ACM. **Proceedings of the 25th annual conference on Computer graphics and interactive techniques**. [S.l.], 1998. p. 275–286.

DURAND, B.; RÓKA, Z. The game of life: universality revisited. In: **Cellular automata**. [S.l.]: Springer, 1999. p. 51–74.

FLEISCHER, K. Investigations with a multicellular developmental model. In: **Artificial life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems**. [S.l.: s.n.], 1996. p. 229–230.

FLEISCHER, K.; BARR, A. H. A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis. In: CITESEER. **Santa Fe Institute Studies in the Sciences of Complexity - Proceedings Volume -**. [S.l.], 1994. v. 17, p. 389–389.

FLEISCHER, K. W. et al. Cellular texture generation. In: ACM. **Proceedings of the 22nd annual conference on Computer graphics and interactive techniques**. [S.l.], 1995. p. 239–248.

FOWLER, D. R.; MEINHARDT, H.; PRUSINKIEWICZ, P. Modeling seashells. **Comp. Graphics**, v. 26, n. 2, p. 379–387, 1992. ISSN 0097-8930.

FOWLER, D. R.; PRUSINKIEWICZ, P.; BATTJES, J. A collision-based model of spiral phyllotaxis. In: ACM. **ACM SIGGRAPH Computer Graphics**. [S.l.], 1992. v. 26, n. 2, p. 361–368.

GAFFNEY, E.; LEE, S. S. The sensitivity of turing self-organization to biological feedback delays: 2d models of fish pigmentation. **Mathematical Medicine and Biology**, IMA, 2013.

GARDNER, M. The fantastic combinations of John Conway's new solitaire game life. **Scientific American**, v. 223, n. 10, p. 120–123, oct. 1970.

GARVIE, M. R.; TRENCHEA, C. Identification of space-time distributed parameters in the gierer–meinhardt reaction-diffusion system. **SIAM Journal on Applied Mathematics**, SIAM, v. 74, n. 1, p. 147–166, 2014.

GIERER, A.; MEINHARDT, H. A theory of biological pattern formation. **Kybernetik**, Springer, v. 12, n. 1, p. 30–39, 1972.

GIURUMESCU, C. A. et al. Quantitative semi-automated analysis of morphogenesis with single-cell resolution in complex embryos. **Development**, Oxford University Press for The Company of Biologists Limited, v. 139, n. 22, p. 4271–4279, 2012.

GLEICK, J. Chaos: Making a new science. Viking, 1987.

GOEHRING, N. W.; GRILL, S. W. Cell polarity: mechanochemical patterning. **Trends in cell biology**, Elsevier, v. 23, n. 2, p. 72–80, 2013.

GOURRET, J.-P.; THALMANN, N. M.; THALMANN, D. Simulation of object and human skin formations in a grasping task. **ACM Siggraph Computer Graphics**, ACM, v. 23, n. 3, p. 21–30, 1989.

GRAVAN, C. P.; LAHOZ-BELTRA, R. Evolving morphogenetic fields in the zebra skin pattern based on turing's morphogen hypothesis. **International Journal of Applied Mathematics and Computer Science**, v. 14, p. 351–361, 2004.

GRISKEY, R. G. **Transport phenomena and unit operations: a combined approach**. Hoboken: John Wiley & Sons, 2005.

GULLAN, P. J. **Os insetos: um resumo de entomologia**. São Paulo: Roca, 2012.

HAMMER, Ø. Diffusion and direct signaling models are numerically equivalent. **Journal of theoretical biology**, Academic Press, v. 192, n. 1, p. 129–130, 1998.

HILDEBRAND, M.; GOSLOW, G. **Análise da Estrutura dos Vertebrados**. São Paulo: Editora Atheneu, 2006.

HISCOCK, T. W.; MEGASON, S. G. Orientation of turing-like patterns by morphogen gradients and tissue anisotropies. **Cell systems**, Elsevier, v. 1, n. 6, p. 408–416, 2015.

HUTTON, T. et al. **Ready, a cross-platform implementation of various reaction-diffusion systems.** 2017. Available from Internet: <https://github.com/GollyGang/ready>.

ITOH, T.; MIYATA, K.; SHIMADA, K. Generating organic textures with controlled anisotropy and directionality. **IEEE Comput. Graph. Appl.**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 23, n. 3, p. 38–45, may 2003. ISSN 0272-1716.

JIRASEK, C.; PRUSINKIEWICZ, P.; MOULIA, B. Integrating biomechanics into developmental plant models expressed using L-systems. **Plant biomechanics**, p. 615–624, 2000.

KAJIYA, J. T.; KAY, T. L. Rendering fur with three dimensional textures. In: ACM. **ACM Siggraph Computer Graphics**. [S.l.], 1989. v. 23, n. 3, p. 271–280.

KARR, J. R. et al. A whole-cell computational model predicts phenotype from genotype. **Cell**, Elsevier, v. 150, n. 2, p. 389–401, 2012.

KIM, T.; LIN, M. Stable advection-reaction-diffusion with arbitrary anisotropy. **Computer Animation and Virtual Worlds**, Wiley Online Library, v. 18, n. 4-5, p. 329–338, 2007.

KIM, T. et al. Fast simulation of Laplacian growth. **IEEE computer graphics and applications**, IEEE, v. 27, n. 2, 2007.

KONDO, S.; MIURA, T. Reaction-Diffusion Model as a Framework for Understanding Biological Pattern Formation. **Science**, American Association for the Advancement of Science, v. 329, n. 5999, p. 1616–1620, sep. 2010. ISSN 1095-9203.

KONDO, S.; SHIROTA, H. Theoretical analysis of mechanisms that generate the pigmentation pattern of animals. In: ELSEVIER. **Seminars in cell & developmental biology**. [S.l.], 2009. v. 20, n. 1, p. 82–89.

KÜCKEN, M.; CHAMPOD, C. Merkel cells and the individuality of friction ridge skin. **Journal of theoretical biology**, Elsevier, v. 317, p. 229–237, 2013.

KURGANOV, A.; ROSENAU, P. On reaction processes with saturating diffusion. **Nonlinearity**, IOP Publishing, v. 19, n. 1, p. 171, 2005.

LINDENMAYER, A. Mathematical models for cellular interactions in development i. filaments with one-sided inputs. In: **Journal of Theoretical Biology**. [S.l.: s.n.], 1968. v. 18, p. 280 – 299.

LIU, R.; LIAW, S.; MAINI, P. Two-stage turing model for generating pigment patterns on the leopard and the jaguar. **Physical review E**, APS, v. 74, n. 1, p. 011914, 2006.

MAINI, P.; PAINTER, K.; CHAU, H. P. Spatial pattern formation in chemical and biological systems. **Journal of the Chemical Society, Faraday Transactions**, Royal Society of Chemistry, v. 93, n. 20, p. 3601–3610, 1997.

MALHEIROS, M.; WALTER, M. Pattern formation through minimalist biologically inspired cellular simulation. In: **Proceedings of Graphics Interface 2017**. [S.l.]: Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, 2017. (GI 2017), p. 148 – 155. ISBN 978-0-9947868-2-1. ISSN 0713-5424.

MALHEIROS, M. d. G.; WALTER, M. Simple and efficient approximate nearest neighbor search using spatial sorting. In: **Graphics, Patterns and Images (SIBGRAPI), 2015 28th SIBGRAPI Conference on**. [S.l.: s.n.], 2015. p. 180–187.

MALHEIROS, M. de G.; WALTER, M. A hybrid geometry and billboard-based model for trees. In: IEEE. **Games and Digital Entertainment (SBGAMES), 2011 Brazilian Symposium on**. [S.l.], 2011. p. 17–25.

MALHEIROS, M. de G.; WALTER, M. Spatial sorting: An efficient strategy for approximate nearest neighbor searching. **Computers & Graphics**, v. 57, p. 112 – 126, 2016. ISSN 0097-8493.

MANUKYAN, L. et al. A living mesoscopic cellular automaton made of skin scales. **Nature**, Nature Research, v. 544, n. 7649, p. 173–179, 2017.

MCGRAW, T. Generalized reaction-diffusion textures. **Computers & Graphics**, v. 32, n. 1, p. 82–92, 2008.

MEINHARDT, H. **Models of Biological Pattern Formation**. New York: Academic Press, 1982.

MEINHARDT, H. **The algorithmic beauty of sea shells**. [S.l.]: Springer Science & Business Media, 2009.

MEINHARDT, H.; KLINGLER, M. A model for pattern formation on the shells of molluscs. **Journal of Theoretical Biology**, Elsevier, v. 126, n. 1, p. 63–89, 1987.

MIURA, T.; MAINI, P. K. Periodic pattern formation in reaction–diffusion systems: An introduction for numerical simulation. **Anatomical science international**, Wiley Online Library, v. 79, n. 3, p. 112–123, 2004.

MIYAZAWA, S.; OKAMOTO, M.; KONDO, S. Blending of animal colour patterns by hybridization. **Nature communications**, Nature Publishing Group, v. 1, p. 66, 2010.

MURRAY, J. D. **Mathematical Biology. I An Introduction {Interdisciplinary Applied Mathematics V. 17}**. [S.l.]: Springer-Verlag, 2002.

MURRAY, J. D. **Mathematical Biology. II Spatial Models and Biomedical Applications {Interdisciplinary Applied Mathematics V. 18}**. [S.l.]: Springer-Verlag, 2003.

NEUMANN, J. von. First draft of a report on the EDVAC. 1945.

NEUMANN, J. von. The general and logical theory of automata. **Cerebral mechanisms in behavior**, New York: John Wiley& Sons, v. 1, n. 41, p. 1–2, 1951.

NULTSCH, W. **Botânica geral**. Porto Alegre: Artmed, 2000.

PAINTER, K. Models for pigment pattern formation in the skin of fishes. In: **Mathematical models for biological pattern formation**. [S.l.]: Springer, 2001. p. 59–81.

PAPERT, S. Teaching children to be mathematicians versus teaching about mathematics. **International journal of mathematical education in science and technology**, Taylor & Francis, v. 3, n. 3, p. 249–262, 1972.

PEARSON, J. E. Complex patterns in a simple system. **Science**, American Association for the Advancement of Science, v. 261, n. 5118, p. 189–192, 1993.

PEITGEN, H.-O.; RICHTER, P. H. **The Beauty of Fractals: Images of Complex Dynamical Systems**. [S.l.]: Springer-Verlag, 1986.

PENA, B.; PEREZ-GARCIA, C. Stability of turing patterns in the brusselator model. **Physical Review E**, APS, v. 64, n. 5, p. 056213, 2001.

PENNYBACKER, M.; NEWELL, A. C. Phyllotaxis, pushed pattern-forming fronts, and optimal packing. **Physical Review Letters**, APS, v. 110, n. 24, p. 248104, 2013.

PORTER, B.; MCCORMACK, J. Developmental modelling with sds. **Computers & Graphics**, Elsevier, v. 34, n. 4, p. 294–303, 2010.

PRUSINKIEWICZ, P.; HAMMEL, M. S.; MJOLSNESS, E. Animation of plant development. In: ACM. **Proceedings of the 20th annual conference on Computer graphics and interactive techniques**. [S.l.], 1993. p. 351–360.

PRUSINKIEWICZ, P. Graphical applications of l-systems. In: **Proceedings of Graphics Interface**. [S.l.: s.n.], 1986. v. 86, n. 1986, p. 247–253.

PRUSINKIEWICZ, P. Modeling of spatial structure and development of plants: a review. **Scientia Horticulturae**, Elsevier, v. 74, n. 1, p. 113–149, 1998.

PRUSINKIEWICZ, P. et al. L-systems: from the theory to visual models of plants. In: CITESEER. **Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences**. [S.l.], 1996. v. 3, p. 1–32.

PRUSINKIEWICZ, P.; JAMES, M.; MĚCH, R. Synthetic topiary. In: ACM. **Proceedings of the 21st annual conference on Computer graphics and interactive techniques**. [S.l.], 1994. p. 351–358.

PRUSINKIEWICZ, P.; LINDENMAYER, A. **The algorithmic beauty of plants**. [S.l.]: Springer Science & Business Media, 2012.

PRUSINKIEWICZ, P.; LINDENMAYER, A.; HANAN, J. Developmental models of herbaceous plants for computer imagery purposes. In: **Computer Graphics (Proceedings of SIGGRAPH 88)**. [S.l.: s.n.], 1988. p. 141–150.

PRUSINKIEWICZ, P. et al. The use of positional information in the modeling of plants. In: ACM. **Proceedings of the 28th annual conference on Computer graphics and interactive techniques**. [S.l.], 2001. p. 289–300.

QUEIROZ, F. da S. **Síntese de Texturas: coloração contrastante de pelagem em mamíferos**. Dissertation (Master) — Universidade Federal de Pernambuco, Brasil, 2011.

RAFLER, S. Generalization of conway's "game of life" to a continuous domain – smoothlife. **arXiv preprint arXiv:1111.1567**, 2011.

RINZEL, J.; TROY, W. C. Bursting phenomena in a simplified oregonator flow system model. **The Journal of Chemical Physics**, AIP Publishing, v. 76, n. 4, p. 1775–1789, 1982.

RUNIONS, A. et al. Modeling and visualization of leaf venation patterns. In: ACM. **ACM Transactions on Graphics (TOG)**. [S.l.], 2005. v. 24, n. 3, p. 702–711.

RUSSELL, S.; NORVIG, P. Artificial intelligence: A modern approach. **Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs**, Citeseer, v. 25, p. 27, 1995.

SANDERSON, A. R. et al. Advanced reaction-diffusion models for texture synthesis. **Journal of Graphics, GPU, and Game Tools**, Taylor & Francis, v. 11, n. 3, p. 47–71, 2006.

SHOJI, H. et al. Origin of directionality in the fish stripe pattern. **Developmental dynamics**, Wiley Online Library, v. 226, n. 4, p. 627–633, 2003.

SHRAIMAN, B. I. Mechanical feedback as a possible regulator of tissue growth. **Proceedings of the National Academy of Sciences of the United States of America**, National Acad Sciences, v. 102, n. 9, p. 3318–3323, 2005.

SILBEY, R.; ALBERTY, R.; BAWENDI, M. **Physical Chemistry**. Hoboken: John Wiley & Sons, 2004.

STATHOPOULOS, A.; IBER, D. Studies of morphogens: keep calm and carry on. **Development**, The Company of Biologists Ltd, v. 140, n. 20, p. 4119–4124, 2013.

TANENBAUM, A. S. **Organização estruturada de computadores**. Rio de Janeiro: LTC, 2001.

TERAN, J. et al. Finite volume methods for the simulation of skeletal muscle. In: EUROGRAPHICS ASSOCIATION. **Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation**. [S.l.], 2003. p. 68–74.

TOFFOLI, T.; MARGOLUS, N. **Cellular Automata Machines: A New Environment for Modeling**. Cambridge, MA, USA: MIT Press, 1987. ISBN 0-262-20060-0.

TURING, A. M. On computable numbers, with an application to the Entscheidungsproblem. **J. of Math**, v. 58, n. 345-363, p. 5, 1936.

TURING, A. M. The chemical basis of morphogenesis. **Phil. Trans. Roy. Soc. London**, B, n. 237, p. 37–72, 1952.

TURK, G. Generating textures on arbitrary surfaces using reaction-diffusion. In: **SIGGRAPH '91 Proceedings**. [S.l.: s.n.], 1991. p. 289–298.

VERSTEEG, H. K.; MALALASEKERA, W. **An introduction to computational fluid dynamics: the finite volume method**. [S.l.]: Pearson Education, 2007.

VOLKENING, A.; SANDSTEDE, B. Modelling stripe formation in zebrafish: an agent-based approach. **Journal of the Royal Society Interface**, The Royal Society, v. 12, n. 112, p. 20150812, 2015.

WALTER, M. **Integration of Complex Shapes and Natural Patterns**. Thesis (PhD) — University of British Columbia, 1998.

WALTER, M.; FOURNIER, A.; MENEVAUX, D. Integrating shape and pattern in mammalian models. In: **SIGGRAPH 2001, Computer Graphics Proceedings**. [S.l.]: ACM Press / ACM SIGGRAPH, 2001. (Annual Conference Series), p. 317–326.

WALTER, M.; FOURNIER, A.; REIMERS, M. Clonal mosaic model for the synthesis of mammalian coat patterns. In: **Graphics Interface**. [S.l.: s.n.], 1998. p. 82–91.

WATANABE, M.; KONDO, S. Changing clothes easily: connexin41. 8 regulates skin pattern variation. **Pigment cell & melanoma research**, Wiley Online Library, v. 25, n. 3, p. 326–330, 2012.

WITKIN, A.; KASS, M. Reaction-diffusion textures. **ACM Siggraph Computer Graphics**, ACM, v. 25, n. 4, p. 299–308, 1991.

WOLFRAM, S. **A New Kind of Science**. [S.l.]: Wolfram Media Inc., 2002.

YOUNG, D. A. A local activator-inhibitor model of vertebrate skin patterns. **Mathematical Biosciences**, Elsevier, v. 72, n. 1, p. 51–58, 1984.

# APPENDIX A — EXPERIMENTS

In this appendix we present timing and other details for the simulation results presented in the main text, summarized in Table A.1. We also list the input files for a few remaining experiments, which were to long to include in earlier chapters. The test machine is powered by an Intel Core i7-4500U CPU running at 1.80 GHz.

Table A.1: Detailed information for figures.

| Fig. | Description | Cell count | Reagents | Iterations | Total time (s) | Cell arrangement | Output type |
|---|---|---|---|---|---|---|---|
| 5.1 | Diffusion test | 1000 | 1 | 225 | 0.627 | square grid | screenshot |
| 5.2 | Gradient test | 4000 | 2 | 961 | 1.586 | square grid | screenshot |
| 5.3 | Turing test | 5000 | 2 | 6400 | 4.503 | square grid | screenshot |
| 5.11 | Spots | 10000 | 4 | 6561 | 13.348 | square grid | screenshot |
| 5.12 | Transition | 10000 | 4 | 6561 | 12.960 | square grid | screenshot |
| 5.13 | Anisotropy | 5000 | 2 | 1600 | 1.620 | square grid | screenshot |
| 5.14 | Rays | 10000 | 4 | 10000 | 71.918 | hexagonal grid | screenshot |
| 5.15 | Rings | 10000 | 4 | 18000 | 125.158 | hexagonal grid | screenshot |
| 6.1 | Shell pattern | 10400 | 2 | 26000 | 86.066 | free | screenshot |
| 6.2 | Growing domain | 2500 | 2 | 30000 | 25.427 | free | screenshot |
| 6.3 | Center growth | 5001 | 2 | 10000 | 22.159 | free | screenshot |
| 6.4 | Spine growth | 4064 | 3 | 16000 | 28.392 | free | screenshot |
| 6.5 | Growth on borders | 6432 | 2 | 18000 | 29.709 | free | screenshot |
| 6.6 | Branching | 2058 | 2 | 10000 | 5.502 | free | screenshot |
| 6.7 | Uniform growth | 5248 | 2 | 15000 | 37.292 | free | screenshot |
| 6.8 | Frozen pattern | 5248 | 2 | 15000 | 36.159 | free | screenshot |
| 6.10 | Reinforcement | 5248 | 3 | 15000 | 38.599 | free | screenshot |
| 7.1 | Game of life | 625 | 2 | 160 | 0.765 | square grid | screenshot |
| 7.2 | Wireworld | 225 | 3 | 72 | 0.985 | square grid | screenshot |
| 7.3 | Laplacian growth | 10000 | 3 | 3200 | 2.919 | square grid | screenshot |
| 7.4 | Cow | 10000 | 1 | 2000 | 3.098 | square grid | screenshot |
| 7.7 | Dalmatian | 10000 | 1 | 2000 | 3.072 | square grid | screenshot |
| 7.8 | African wild dog | 10000 | 1 | 2000 | 3.094 | square grid | screenshot |
| 7.13 | Giraffe | 10000 | 2 | 38000 | 54.722 | square grid | screenshot |
| 7.15 | Nudibranch | 10000 | 2 | 3000 | 4.801 | square grid | screenshot |
| 7.16 | Poison dart frog | 10000 | 2 | 8000 | 8.135 | square grid | texture |
| 7.18 | Moray eel | 20000 | 3 | 5000 | 21.122 | square grid | texture |
| 7.21 | Leopard | 6228 | 2 | 14000 | 22.298 | free | texture |

We briefly discuss here two cases. The moray example has 20,000 cells, runs for 5,000 iterations and uses a square grid NNS, just taking 21 seconds to execute. The leopard example reaches 6,228 cells in 14,000 iterations. When run using a *k*-d tree, it takes 43 seconds, whereas with spatial sorting it takes 22 seconds. Spatial sorting is consistently faster than the *k*-d tree, and in this particular case, the average miss rate when detecting neighbors is 0.5%, using a 48 Moore neighborhood.

**Listing A.1** Experiment file for Figure 5.14, radial stripes.

```
define chemical V anisotropic
define chemical U
define chemical R
define chemical P

// -------- prepattern

use chemical U conc 4 dev 1 diff 0.024
use chemical V conc 4 dev 1 diff 0.00625
use chemical R conc 0       diff 0.05
use chemical P conc 0       diff 0
create hex_grid 100 100
set cell 5050 chemical P conc 1

// -------- rules

rule if P conc == 1 change R conc  4
rule if P conc == 0 change R conc -0.001
rule always polarize R conc
rule always react U V scale 0.004 turing alpha 16 beta 12

colormap select gradient
colormap slot 00 use color "white"
colormap slot 99 use color "420009"
stop at 10000
```

**Listing A.2** Experiment file for Figure 5.15, concentric stripes.

```
define chemical V
define chemical U anisotropic
define chemical R
define chemical P

// -------- prepattern

use chemical U conc 4 dev 1 diff 0.03
use chemical V conc 4 dev 1 diff 0.00625
use chemical R conc 0       diff 0.05
use chemical P conc 0       diff 0
create hex_grid 100 100
set cell 5050 chemical P conc 1

// -------- rules

rule if P conc == 1 change R conc  4
rule if P conc == 0 change R conc -0.001
rule always polarize R conc
rule from 3000 always react U V scale 0.006 turing alpha 16 beta 12

colormap select gradient
colormap slot 40 use color "0ee1da"
colormap slot 80 use color "80763e"
stop at 18000
```

**Listing A.3** Experiment file for Figure 6.1, shell growth.

```
define chemical V
define chemical U
define time_step 0.1

// -------- prepattern

use chemical U conc 4 dev 0.5 diff 0.08
use chemical V conc 4 dev 0.5 diff 0.01
use polarity -90
create sqr_grid 80 1

// -------- rules

rule if AGE == 200 divide direction 0
rule if AGE <  200 react U V scale 0.01 turing alpha 16 beta 12
rule if AGE == 200 change U diff -0.08
rule if AGE == 200 change V diff -0.01

colormap select gradient
colormap slot 00 use color "dd9861"
colormap slot 60 use color "3f0400"
colormap slot 80 use color "782e07"
stop at 26000
```

**Listing A.4** Experiment file for Figure 6.2, growing domain.

```
define chemical V anisotropic
define chemical U
define domain 200 200

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.08
use chemical V conc 4 dev 2 diff 0.01
use polarity 0
create sqr_grid 1 25 dev 0.1

// -------- rules

rule always react U V scale 0.01 turing alpha 16 beta 12
rule if AGE == 300 divide direction 0
stop at 30000
```

**Listing A.5** Experiment file for Figure 6.3, center growth.

```
define chemical V //anisotropic
define chemical U //anisotropic
define time_step 0.2

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.08 // 0.04
use chemical V conc 4 dev 2 diff 0.01
create cell 0 0

// -------- rules

rule always react U V scale 0.01 turing alpha 16 beta 12
rule if AGE == 1 divide direction 137.5
stop at 10000
zoom level 1.423
```

**Listing A.6** Experiment file for Figure 6.4, growth from spine.

```
define chemical V //anisotropic
define chemical U
define chemical F
define division_limit 8

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.08
use chemical V conc 4 dev 2 diff 0.01
use chemical F conc 1       diff 0
create sqr_grid 40 1 fixed

// -------- rules

rule always react U V scale 0.01 turing alpha 16 beta 12
rule from 1 if AGE == 0 change F conc -1
rule if F conc == 1 and
rule probability 0.004 divide direction 90
rule if F conc == 1 and
rule probability 0.004 divide direction -90

//snap at 0 4000 8000 12000 16000 then exit
stop at 16000
zoom level 1.387
```

**Listing A.7** Experiment file for Figure 6.5, growth on borders.

```
define chemical V anisotropic
define chemical U

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.08
use chemical V conc 4 dev 2 diff 0.01
create cell 0 0

// -------- rules

rule always react U V scale 0.01 turing alpha 16 beta 12
rule if NEIGHBORS <= 3 and
rule probability 0.004 divide direction 0 dev 180 // 10 dev 0
stop at 18000
zoom level 0.9737
```

**Listing A.8** Experiment file for Figure 6.6, branching pattern.

```
define chemical V anisotropic
define chemical U

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.06
use chemical V conc 4 dev 2 diff 0.01
use polarity 90
create cell 0 0

// -------- rules

rule always react U V scale 0.005 turing alpha 16 beta 12
rule if NEIGHBORS <= 2 and
rule probability 0.01 divide direction   0 dev 0
rule if NEIGHBORS <= 2 and
rule probability 0.001 divide direction  20 dev 0
rule if NEIGHBORS <= 2 and
rule probability 0.001 divide direction -20 dev 0
stop at 10000
zoom level 0.6627
```

**Listing A.9** Experiment file for Figure 6.7, uniform growth.

```
define chemical V
define chemical U
define division_limit 6

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.15
use chemical V conc 4 dev 2 diff 0.01
create hex_circle 45

// -------- rules

rule always react U V scale 0.01 turing alpha 16 beta 12
rule from 2000 until 15000 probability 0.0001 divide direction 0 dev 180
snap at 2000 6000 9000 12000 15000 then exit
zoom level 1.303
```

**Listing A.10** Experiment file for Figure 6.8, growth with frozen concentrations.

```
define chemical V
define chemical U
define division_limit 6

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.15
use chemical V conc 4 dev 2 diff 0.01
create hex_circle 45

// -------- rules

rule until 1998 always react U V scale 0.01 turing alpha 16 beta 12
rule from 1999 until 1999 always change V diff -0.01 // stop diffusion
rule from 2000 until 15000 probability 0.0001 divide direction 0 dev 180

snap at 2000 6000 9000 12000 15000 then exit
zoom level 1.303
```

**Listing A.11** Experiment file for Figure 6.10, growth with reinforcement.

```
define chemical R
define chemical V
define chemical U
define division_limit 6

// -------- prepattern

use chemical U conc 4 dev 2 diff 0.15
use chemical V conc 4 dev 2 diff 0.01
use chemical R conc 0       diff 0.0004
create hex_circle 45

// -------- rules

rule always react U V scale 0.01 turing alpha 16 beta 12

// copy pattern from V to R, mapping it to the [0,2] interval
rule from 1999 until 1999 always map V conc 1.0 7.0 to PAT 0.0 2.0
rule from 1999 until 1999 always change R conc PAT

// reinforcement of R
rule from 1999 if R conc in 0.1 0.9 change R conc -0.0004
rule from 1999 if R conc in 1.1 1.9 change R conc +0.0004

rule from 2000 until 15000 probability 0.0001 divide direction 0 dev 180

snap at 2000 6000 9000 12000 15000 then exit
zoom level 1.303
```

**Listing A.12** Experiment file for Figure 7.2, Wireworld cellular automaton.

```
define chemical S
define chemical N
define chemical C

// -------- prepattern

use chemical S conc 0 diff 0
use chemical N conc 0 diff 0.01
use chemical C conc 0 diff 0
create sqr_grid 15 15
set cells 77 to 79 chemical S conc 3
set cell  61 chemical S conc 3
set cell  65 chemical S conc 3
set cell  46 chemical S conc 3
set cells 50 to 58 chemical S conc 3
set cell  31 chemical S conc 3
set cell  35 chemical S conc 3
set cell  17 chemical S conc 2
set cell  18 chemical S conc 1
set cell  19 chemical S conc 3

// -------- rules

// periodic counter
rule if C conc == 0 change C conc +1
rule if C conc == 1 change C conc +1
rule if C conc == 2 change C conc -2

// 0: mark head [1] in the neighborhood
rule if C conc == 0 and
rule if S conc == 1 change N conc +1

// 1: diffusion happens in the neighborhood

// 2: force neighborhood to zero
rule if C conc == 2 change N conc -1

// 2: empty [0] -> empty [0]

// 2: head [1] -> tail [2]
rule if C conc == 2 and
rule if S conc == 1 change S conc +1

// 2: tail [2] -> wire [3]
rule if C conc == 2 and
rule if S conc == 2 change S conc +1

// 2: wire [3] -> head [1] if exactly one or two neighbor cells are heads, otherwise keep as wire
rule if C conc == 2 and
rule if S conc == 3 and
rule if N conc in 0.01 0.02 change S conc -2

colormap select striped
colormap slot 00 use color "303030" // empty
colormap slot 25 use color "0080ff" // head
colormap slot 50 use color "ffffff" // tail
colormap slot 75 use color "ff8000" // wire
stop at 72
```

**Listing A.13** Experiment file for Figure 7.3, Laplacian growth.

```
define chemical P limit 1
define chemical G
define chemical C

// -------- prepattern

use chemical P conc 0   diff 0.01
use chemical G conc 0.1 diff 0.1
use chemical C conc 0   diff 0

create sqr_grid 100 100
set cell 4949 chemical P conc 1
set cell 4950 chemical P conc 1
set cell 4951 chemical P conc 1

// -------- rules

// clock
rule if C conc == 0 change C conc +1
rule if C conc == 1 change C conc -1

// force all to zero
rule if C conc == 1 change P conc -1

// some non-active neighbors will be activated
rule if C conc == 1 and
rule probability G conc and
rule if P conc in 0.03 0.04 change P conc +2

// keep previously activated
rule if C conc == 1 and
rule if P conc > 0.9 change P conc +2

// change probability gradient
rule if P conc == 0 change G conc +0.0001
rule if P conc == 1 change G conc -0.01

stop at 3200
```

**Listing A.14** Experiment file for Figure 7.7, reinforcement pattern for Dalmatian dog.

```
define chemical P limit 1

// -------- prepattern

use seed 4
use chemical P conc 0.5 dev 0.5 diff 0.01
create sqr_grid 100 100

// -------- rules

rule if P conc > 0.8 change P conc +0.03
rule if P conc < 0.6 change P conc -0.01

colormap select gradient
colormap slot 00 use color "white"
colormap slot 99 use color "black"
stop at 2000
```

**Listing A.15** Experiment file for Figure 7.8, reinforcement pattern for African wild dog.

```
define chemical P limit 3

// -------- prepattern

use seed 5
use chemical P conc 0.5 dev 0.5 diff 0.01
create sqr_grid 100 100

// -------- rules

rule from 100 if P conc > 0.5 change P conc +0.01
rule from 100 if P conc < 0.5 change P conc -0.04

colormap select gradient
colormap slot 00 use color "f3b88e"
colormap slot 40 use color "black"
colormap slot 60 use color "black"
colormap slot 99 use color "white"
stop at 2000
```

**Listing A.16** Experiment file for Figure 7.16, saturated RD pattern for poison dart frog.

```
define chemical U limit 6.3
define chemical V anisotropic
define time_step 0.5

// -------- prepattern

use polarity 90
use chemical U conc 4 dev 2 diff 0.30
use chemical V conc 4 dev 2 diff 0.02
create sqr_grid 100 100
set cells 0 to 4999 chemical U conc 4 dev 2 diff 0.011 polarity none
set cells 0 to 4999 chemical V conc 4 dev 2 diff 0.003 polarity none

// -------- rules

rule always react U V scale 0.008 turing alpha 16 beta 12
colormap select gradient
colormap slot 20 use color "ffbf00"
colormap slot 25 use color "cyan"
colormap slot 45 use color "202020"
texture size 512 512
stop at 5000
```

**Listing A.17** Experiment file for Figure 7.18, saturated RD pattern for moray eel.

```
// moray eel (muraena melanotis)

define chemical U limit 6.5
define chemical V limit 6.5
define chemical P

define time_step 0.3

// -------- prepattern

use seed 1

use chemical U conc 4 dev 2 diff 0.05
use chemical V conc 4 dev 2 diff 0.01
use chemical P conc 0       diff 0.01

create sqr_grid 200 100

set cells     0 to  4999 chemical U diff 0.40
set cells  5000 to  5399 chemical U diff 0.20

set cells 19000 to 19199 chemical U diff 0.20
set cells 19200 to 19999 chemical U diff 0.40

set cell 10240 chemical P conc 8
set cell 10850 chemical P conc 8

// -------- rules

rule always react U V scale 0.03 turing alpha 16 beta 12

rule from 2500 until 2000 always change P diff -0.01 // stops gradient
rule if P conc > 0.01 change V conc +1.0 // cells affected by gradient

colormap select gradient
colormap slot 25 use color "30292d"
colormap slot 30 use color "e8e336"
colormap slot 40 use color "white"

stop at 5000

texture size 800 400
```

**Listing A.18** Experiment file for Figure 7.21, saturated RD pattern for leopard.

```
define chemical U limit 4.05
define chemical V

define division_limit 6
define domain packed
define time_step 0.1

// -------- prepattern

use chemical U conc 4 dev 0 diff 0.40
use chemical V conc 4 dev 0 diff 0.01

create hex_grid 50 58

set cell 205 chemical V conc 10
set cell 215 chemical V conc 10
set cell 225 chemical V conc 10
set cell 235 chemical V conc 10
set cell 245 chemical V conc 10

set cell 610 chemical V conc 10
set cell 620 chemical V conc 10
set cell 630 chemical V conc 10
set cell 640 chemical V conc 10

// other seed cells omitted for brevity:
// 1005, 1015, 1025, 1035, 1045
// 1410, 1420, 1430, 1440
// 1805, 1815, 1825, 1835, 1845
// 2210, 2220, 2230, 2240
// 2605, 2615, 2625, 2635, 2645

// -------- rules

rule always react U V scale 0.025 turing alpha 16 beta 12

rule from 4000 probability 0.0001 divide direction 0 dev 180

colormap select gradient
colormap slot 20 use color "251c12"
colormap slot 30 use color "98682d"
colormap slot 70 use color "fbc38d"
colormap slot 80 use color "f5e8a8"

stop at 14000
texture size 512 512
zoom level 1.503
```