Procedural Modelling with Reaction Diffusion and Growth of Thin Shells

Charles Gingras[*]
McGill University

Paul G. Kry[†]
McGill University

ABSTRACT

We investigate a procedural shape modeling approach based on reaction-diffusion equations and physically based growth of thin shells. This inspiration of this work comes from the morphological development of living tissues, such as plants leaves. There are numerous choices that can be made in assembling a computer simulation of these growth system. We explore two main approaches, one where a reaction-diffusion simulation is first run with the results used to identify regions of growth, and the other where we simulate shell growth concurrently with a reaction-diffusion simulation in the manifold. We demonstrate that a variety of interesting shapes can be grown in this manner, and provide some intuition to the challenging problem of associating changes in parameter settings with the final shape.

Index Terms: Computing methodologies—Physical simulation; Computing methodologies—Shape modeling

1 INTRODUCTION

The shapes of leaves and flower petals can arise from growth of a thin sheet. Using a physically based simulation of growth, the final shape is the static elastic equilibrium of a thin shell, where out of plane bending is produced by both localized growth, as well as differing amounts of growth on top and bottom surfaces. Recent research demonstrates that almost any shape can be produced [16], but does not attempt to describe the process that creates the growth patterns.

In this work we explore a procedural growth technique that takes inspiration from nature, where growth is controlled by biochemical signals that diffuse through the tissues. We initially started by exploring solutions to the reaction diffusion equations solved on a regular grid, but ultimately chose to solve the equations on a triangular mesh. Growing on a mesh requires the use of the cotangent Laplacian, which is more complicated that the regular grid Laplacian. However, the use of meshes ultimately simplifies the implementation by matching the mesh topology of the thin shell simulation. While the original rest-shape meshes need not be flat, we focus on initially flat meshes, and likewise we focus on the simpler case of localized growth rather than differential growth.

Our full system for procedural modeling creates a variety of complicated shapes based on different parameters provided to the the reaction diffusion problem, and ultimately, we produce a tool for exploration of this forward problem. Figure 1 shows a preview of the final shapes we are able to produce.

2 RELATED WORK

Biological growth is a complex process involving biochemical signals, gene expression and protein synthesis that when combined initiate and fuel physical reactions within cells to allow them to divide, leading to tissue expansion. Much work has been done to simulate this process using various approximations.

[*]charles.gingras@mail.mcgill.ca
[†]e-mail: kry@cs.mcgill.ca

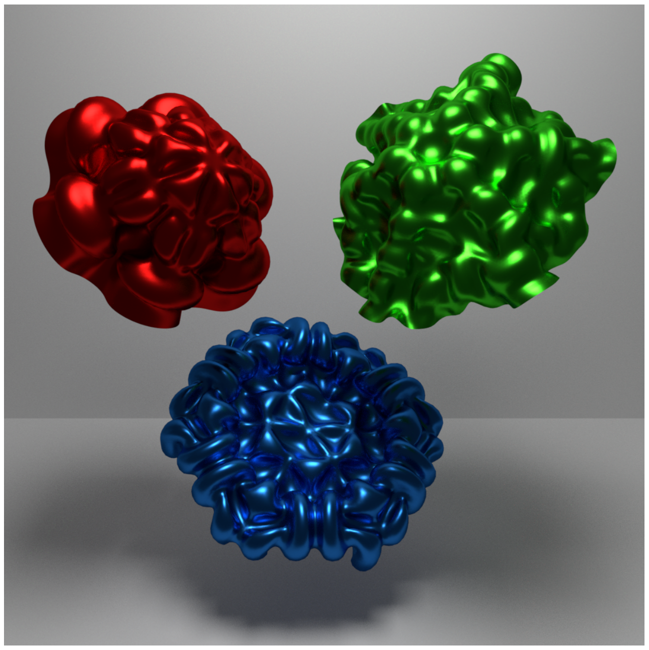


Figure 1: Three examples of procedurally modelled shapes produced through a combination of reaction diffusion and thin shell growth.

Hutton et al. [3] have developed software called “Ready” for exploring continuous and discrete cellular automata, including reaction-diffusion systems, on both grids and arbitrary meshes. This makes an interesting connection between Conway’s game of life and the 3D and continuous extensions which can lead to the formation of complex, biologically plausible shapes.

Early work in computer graphics recognized that the shapes of plants can be seen as the product of a algorithm. Prusinkiewicz and Lindenmayer proposed L-systems, a grammar to model plant shapes, and demonstrated that such a method while simple is extremely powerful for generating the shapes of twigs, trees, flowers, and other plant structures [8].

More recently, Kennaway et al. [5], explored the use of polarity fields to direct and control growth. Their model uses a set of factors, some of which can propagate through the sheet while others cannot, this can be used to control different aspects of the growth. Groups of networks cooperate to control propagation of these factors, that is, the expression of polarity, gene expression, and growth constraint and regulation. Since polarity and growth rates are specified independently, they may be combined in many ways to generate a variety of results. In contrast to this, van Rees et al. [16] were able to determine rules for growth factors and directions to change an orthotropically growing bi-layer from a simple starting shape into a specified target shape. For instance, they obtain the shape of a snapdragon flower petal by growing a cylinder. The downside to this approach is that it does not allow for generation of shapes on the fly since the final shape needs to be pre-specified.

Biological growth is not solely observable in 3D: often enough, it occurs along a surface and can be represented as a 2D process.

Malheiros et al. [6] model cell growth in combination with reaction-diffusion simulation. They decouple reaction and diffusion so that diffusion happens at all times, while the reaction is specific to each cell. With the addition of anisotropy and gradients, they can generate a variety of biologically plausible patterns in 2D. In our work, we are inspired to reaction-diffusion to generate 3D shapes by allowing growth to bend the surface out of the plane.

Turing originally proposed reaction-diffusion as a way to model biological pattern formation stemming from known physical elements [14]. The model itself is quite simple mathematically, which leads many biologists to dismiss it on the basis that actual biological phenomena are quite complex. Despite this, it is believed that "[the] logic of pattern formation can be understood with simple models, and by adapting this logic to complex biological phenomena, it becomes easier to extract the essence of the underlying mechanisms" [1]. In fact, reaction-diffusion can be used successfully to recreate patterns resembling mitosis, coral, sea shells, and many patterns exhibited by various animals. Although work has been done on growth of plants and patterns to be used as textures, the use of reaction-diffusion as a way to represent the propagation of biochemicals in a tissue and their reactions with receptors to model growth in 3D space has not been explored yet.

Reaction-diffusion was also used to create textures for specific meshes by running it directly on the mesh. By running the reaction process iteratively on its own outputs, Turk [15] was able to create biologically plausible textures specific to the meshes they were generated for such as zebra stripes and leopard spots. Likewise, Witkin and Kass [17] produced displacement maps and textures using reaction diffusion.

Despite all this, reaction-diffusion remains largely unused for texture synthesis and other applications: tuning the parameters to obtain useful patterns can be quite lengthy and the whole process is not necessarily computationally cheap. Sanderson et al. [9] explored various methods to gain control over the patterns that are formed. Namely, they note that the free parameters involved in the process need not be fixed and can vary relative to the planar coordinates. Similarly, Wu et al. [18] and Fuselier et al. [2] explored different methods to numerically solve reaction-diffusion systems.

We use the thin shell simulator of Narain et al. [7], but ultimately modify the code so that plastic deformation may be specified by a growth function. In our simulation of reaction diffusion on triangular meshes, we use the standard approach for computing the mesh cotangent Laplacian [11]. For larger diffusion rates in the reaction diffusion equations, we note that it may benefit from implicit time integration [12], but we use explicit time integration and take care to use small enough steps to ensure stability. Given that we are ultimately simulating the RD problem within a manifold, we note that we could also employ methods proposed by Stam [13] for fluid simulation on smooth surfaces.

3 Method

Our method runs in two steps that run in a loop: a reaction-diffusion (RD) step followed by a growth step. At least one RD step is needed for growth to happen, however, the later RD steps may be skipped. Skipping the later RD steps makes the growth simulation depend solely on the initial distribution of the RD reagents. This typically leads to more predictable shape outcomes. When they are not skipped, the later RD steps occur after growth steps: hence, the shape of the mesh is not the same as the beginning mesh. Some triangles will have a different area and some have been added by subdivision. This in turn affects the RD process, meaning the growth algorithm also has an influence on the final outcome of the simulation. One benefit is that this increases the variety of shapes that can be obtained, but it also makes it harder to predict what shape a given set of RD parameters will produce.

The RD step is performed using CUDA parallel computing since most of the step is highly parallelizable, especially on meshes containing many vertices. This scales the running time down to a level that makes it practical to compute shapes in seconds to minutes.

Table 1: Range of parameters explored. Not all combinations produce interesting shapes. Typically, a much narrower range of values was observed to give nice results, but this narrowed range is dependent on the RD feed and kill parameters. Step size refers to the time step; CUDA steps refers to the number of RD steps between each growth step; refer to Section 3.2 for the scaling factors.

parameter	**low**	**high**
step size	0.1	1
diffusion rate a	0.1	1
diffusion rate b	0.05	0.5
CUDA steps N	25	100000
scaling factor γ	0.01	0.2
scaling factor λ	0.05	4.0
growth exponent n	0	3
growth exponent m	0	3
runtime (s)	10	120
feed value f	0.015	0.08
kill value k	0.046	0.065

3.1 The reaction-diffusion Step

There are several parts to the RD step that we describe here, from setting up the CUDA computations, computing the Laplacian coefficients, and the final computation of the concentration of quantities at the next RD step.

3.1.1 Preparatory Calls

To perform the RD step, a number of preparatory calls are required to setup memory and data structures. The Gray-Scott RD model is based off of two reagents reacting together (we provide a full description along with the equations in Section 3.1.3). Thus, each mesh vertex must store the concentrations of these reagents ranging from zero to one. The other RD parameters, namely the feed rate, the kill rate, and the time step are stored along with the mesh data structure and are constant across the mesh. They are fixed at run time and we keep them constant across the duration of the simulation.

Concentrations stored at vertices need to be assembled into arrays for CUDA to use them. Likewise, the structure of the mesh is passed as two distinct arrays: one containing the per-vertex coordinates for each vertex and the other contains the per-face vertex indices. These are necessary to form the cotangent Laplacian.

3.1.2 Forming the Cotangent Laplacian

When the RD step begins, we must first evaluate the cotangent Laplacian of each vertex,

$$\nabla^2 a_i \approx \frac{1}{2A_i} \sum_{j \in N_i} \left(\cot \alpha_{ij} + \cot \beta_{ij}\right)(a_j - a_i), \qquad (1)$$

where a is a quantity stored at vertices, N_i is the neighborhood of vertex i, and α_{ij} and β_{ij} are the angles opposite to the edge connecting vertices i and j, as shown in Figure 2. The area A_i is approximated as a third of total area of the adjacent triangles, but ultimately we typically assume a constant area to improve stability in our simulations.

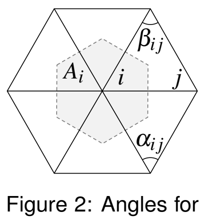


Figure 2: Angles for cotangent Laplacian

We compute edge lengths and then calculate the triangle areas using Kahan's approach to the Heron area formula [4, 10]. We can thus evaluate the cotangents for all edges by using the law of sines and the law of cosines. Lastly, we assemble a sparse matrix S where each $s_{ij} = \cot \alpha_{ij} + \cot \beta_{ij}$.

Note that all the prior calculations, except for the assembly of the matrix, are done in parallel. During the assembly, multiple cotangent values typically are added together to form a single entry in the sparse matrix. Although this does not inherently prohibit concurrency, it does complicate implementation. Looking further into this matter would be worthwhile in future research, especially if short RD steps are to be used. The recomputation of the Laplacian needs to be performed every time since the mesh changes as it grows.

3.1.3 Reaction-diffusion

Our implementation of RD is based on the Gray-Scott model, which formulates two coupled partial differential equations for quantities a and b,

$$\frac{\partial a}{\partial t} = D_a \nabla^2 a - ab^2 + f(1-a), \tag{2}$$

$$\frac{\partial b}{\partial t} = D_b \nabla^2 b + ab^2 - (k+f)b, \tag{3}$$

where D_a and D_b are the diffusion rates of a and b, and f and k are the feed and kill rates, respectively. Note here that a and b represent the concentrations of morphogenes (i.e., reagents) across the mesh that diffuse with the cotangent Laplacian defined above. In this model, reagent a is consumed upon coming into contact with reagent b to produce even more reagent b. The feed rate represents a constant amount of reagent a that is being added at each location (in our case, each vertex) to ensure the reaction can be sustained. The kill rate represents the exact opposite and serves the purpose of removing reagent b. Using forward Euler time stepping of these equations, we have a simple update rule for computing the new quantities a' and b' across our mesh at the next time step, that is,

$$a' = a + \Big(D_a \nabla^2 a - ab^2 + f(1-a)\Big)\Delta t, \tag{4}$$

$$b' = b + \Big(D_b \nabla^2 b + ab^2 - (k+f)b\Big)\Delta t. \tag{5}$$

This evaluation is separated in two parts. The first consists of summing the appropriate entries from the sparse matrix to obtain the Laplacian of each quantity at each vertex. The second evaluates a' and b' from Equations 4 and 5 for each vertex. Both steps are kept separate since CUDA does not guarantee synchronization between different blocks of threads. Keeping both parts in separate kernels enforces that all threads must complete work on the first part before any can work begins on the second.

This RD step can be executed multiple times in a row before the growth step begins. It typically needs to be tuned specifically to the RD parameters being used to somewhat match the speed at which the reaction spreads across the mesh. Too few repetitions on a slow spreading diffusion typically yields large growth in a constrained area of the mesh, which can eventually become unstable.

3.2 The Growth Step

The growth step is implemented with a thin shell simulation. A stretching force is applied to each face based on increasing an isotropic plastic rest strain on different triangles on the mesh, leading to growth. This isotropic plastic rest strain s is increased on each time step based on a function of the concentrations of a and b on the face. We compute the concentration of a quantity on the face by simply averaging the concentration at the three adjacent vertices.

To promote stability of the growing sheet, the plastic strain increment δs is scaled by the logarithm of the current number of iterations completed. This prevents the sheet from growing too quickly in a very narrow area at the beginning of the simulation. At later stages, the shell grows faster. Within this basic framework, the possible

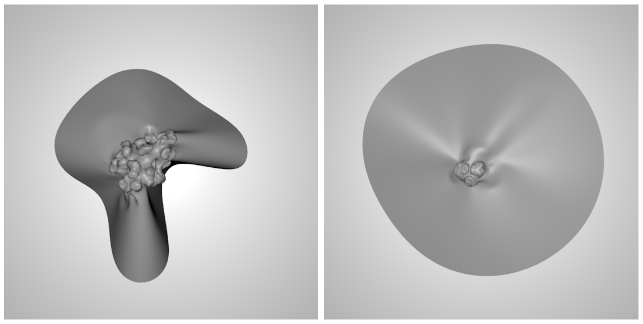


Figure 3: Examples of continuous coupled RD and growth, where parameter selection does not develop into stem structures before converging to zero. In these examples, an exponential growth function is used with f=0.0367, k=0.0649. The example on the left uses m=1, n=1, γ=0.04, and λ=1, while the example on the right uses m=1, n=2, γ=0.029, and λ=1.08.

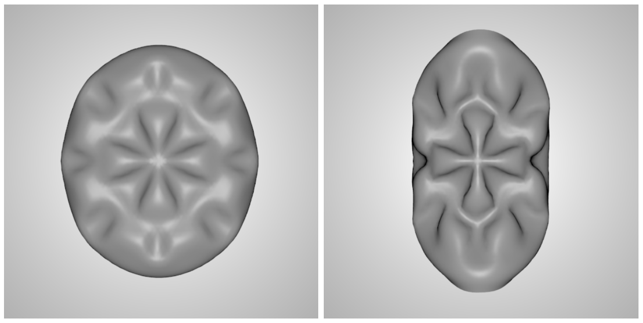


Figure 4: Left: 30 simulation steps. Right: 50 simulation steps. These examples use a polynomial growth function with m=0, n=1, γ=0.2, f=0.039 and k=0.058. Mesh refinement is disabled in this instance.

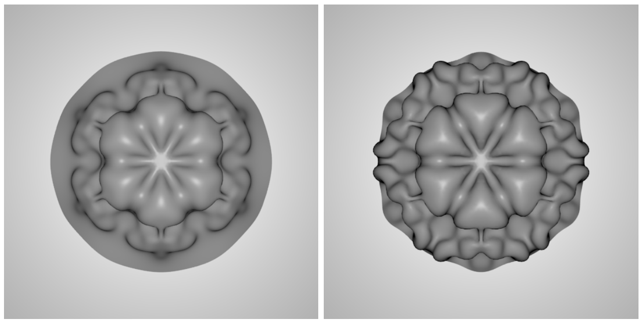


Figure 5: Left: 20 simulation steps. Right: 30 simulation steps. These examples use a polynomial growth function with m=0, n=1, γ=0.2, f=0.0545 and k=0.062.

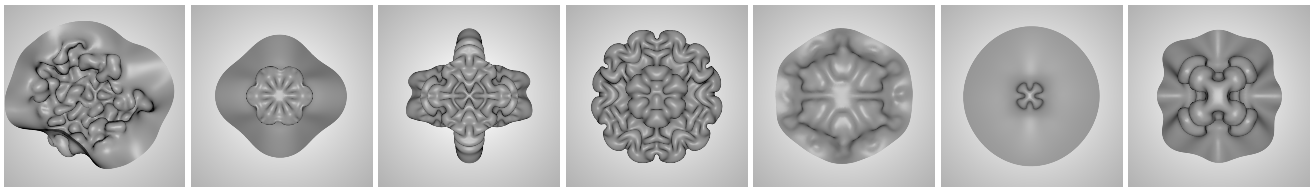
choices for producing growth are numerous. We explored two families of functions for the plastic strain increment, specifically,

$$\delta s_1(a,b) = -\exp\left(\frac{1}{\gamma}a^m b^n - 1\right)\lambda, \quad \text{and} \tag{6}$$

$$\delta s_2(a,b) = -\frac{1}{\gamma}a^m b^n, \tag{7}$$

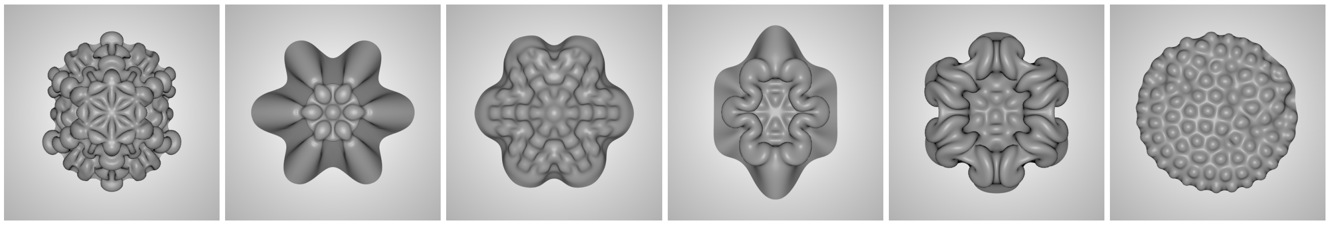
where we vary parameters γ and λ and the degrees m and n for the polynomial terms up to order 3 in either parameter (a or b).

We also considered other variations on the growth step. Adaptive remeshing is straightforward with the thin shell simulation code we use, but redistributing concentrations on remeshing is not obvious:



f=0.0367	f=0.0545	f=0.039	f=0.078	f=0.0545	f=0.029	f=0.0545
k=0.0649	k=0.062	k=0.058	k=0.061	k=0.062	k=0.062	k=0.062
m=0	m=0	m=0	m=0	m=0	m=0	m=1
n=1	n=1	n=1	n=1	n=1	n=1	n=1
γ=0.2	γ=0.15	γ=0.05	γ=0.02	γ=0.1, λ=0.5	γ=0.05, λ=1	γ=0.045, λ=1
poly	poly	poly	poly	exp	exp	exp

Figure 6: Examples of continuous RD growth. The left examples used a polynomial function of the RD parameters for growth whereas the right used an exponential function. Parameters are listed below each shape for the feed rate, kill rate, function parameters, and function used.



f=0.029	f=0.0545	f=0.039	f=0.078	f=0.078	f=0.025
k=0.057	k=0.062	k=0.058	k=0.061	k=0.061	k=0.06
m=0	m=1	m=1	m=0	m=0	m=0
n=1	n=0	n=0	n=1	n=1	n=1
γ=0.05	γ=0.05	γ=0.05	γ=0.05	γ=0.05	γ=0.05
N=4000	N=3000	N=3000	N=30000	N=40000	N=10000

Figure 7: Various forms created by first computing an RD pattern, and then growing the shell using a polynomial function. Parameters are listed below each shape for the feed rate, kill rate, polynomial function parameters, and RD steps.

remeshing is simply done by subdividing edges at their midpoint. New vertices are assigned reagent concentrations equal to the average of their 2 neighbors. In contrast, mesh refinement can be an interesting mechanism to explore. Note that we do not change the concentration of reagents after a growth step, even though the concentrations should be reduced if one assumes that the same material is now spread out over a larger area. Adjusting concentrations after growth might lead to an equilibrium shape after a sufficient number of growth steps, and is an interesting idea to explore in future work.

We are using a single thin shell rather than a bi-layer, thus, it would be interesting to consider biasing the bend angle to produce bending in a desired direction during growth. Changing how we alternate (or not) between RD and growth steps can have an important effect on shape. A high ratio of RD steps to growth steps will result in more uniform growth of the mesh whereas growth is more localized to where RD began in the opposite case. Finally, we note that any mesh can be used as the base mesh, with the cotangent Laplacian allowing for diffusion on any 3D triangular mesh.

4 Results

In all our results, for consistency, we grow shapes using a flat disc as the base mesh. We likewise use the same parameters for the ARC-Sim thin shell simulation across all simulations: we configure the material to have the properties of aluminum, and set the remeshing parameter to have refinement angle of 0.3, a refinement compression of 0.005, a refinement velocity of 0.5, size range of 2e-1 to 2e-2, and minimum aspect ratio 0.2. We typically disable collision detection and response in order to have faster compute times, though we did experiment with enabling collision response for a few examples that exhibited self intersecting growth.

Most simulations begin with all vertices having concentrations $a = 1$ and $b = 0$ except for the middle vertex set at $a = 0$ and $b = 1$. In some cases, the RD quickly converges to zero, in which cases a vertex on the edge of the disc is used rather than the center one. Results are defined as interesting in a somewhat arbitrary manner: typically a result is labelled as interesting if it possesses a distinct shape when compared to other results, if it is appealing to the eye, or if it exhibits a plant-like shape.

4.1 Continuous RD

Having the RD step occur in between each growth step has an outcome that is harder to predict since the mesh topology changes as the mesh grows: some triangles expand more than others, and an iterative refinement process is constantly subdividing triangles that grow past an arbitrary area size. In some cases, this can significantly impact the outcome. For instance, a mitosis like pattern can be obtained through RD, but if the mesh subdivides too quickly, the RD does not get the chance to spread out across the mesh. Subdivision is sometimes so quick as to cause the RD to vanish almost immediately, i.e., the concentration of reagent b goes to zero everywhere.

Figure 3 shows examples where we were attempting to obtain a stem like structure that would grow out from the center and slowly grow limbs. Examples of coupled RD and growth at different time steps can be seen in Figures 4 and 5. The main growth pattern typically emerges after 10 to 25 simulation steps, after which growth still occurs but the pattern either does not change significantly or loses its visual appeal. Overall, Figure 6 shows a collection of interesting shapes that we observed with different parameter settings.

Figure 10, top, gives a good example of the difficulty in predicting the growth outcome with respect to the reaction diffusion pattern:

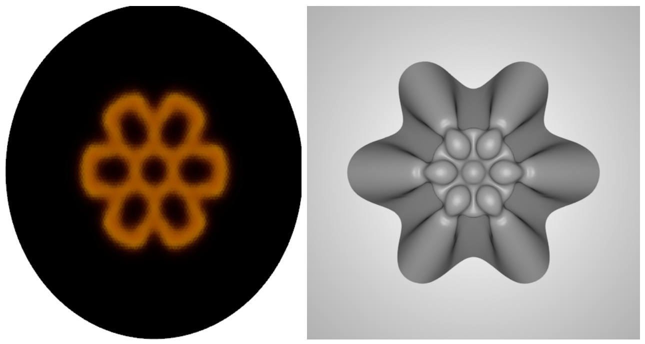


Figure 8: Left: a flower-like RD pattern that has "holes" in it. Right: the resulting mesh exhibits bead-like patterns. This example uses a polynomial growth function with m=1, n=0, γ=0.05, f=0.0545 and k=0.062.

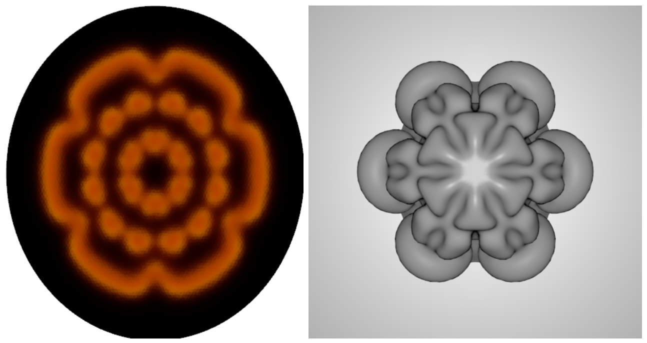


Figure 9: On the left, we have an RD pattern of concentric rings. On the right, we can see that each ring led to a "layer" in the output mesh. This example uses a polynomial growth function with m=0, n=1, γ=0.05, f=0.029 and k=0.057.

one would simply expect a concentric-like pattern to form, but something much more intricate was obtained. On the other hand, Figure 10 bottoms shows an outcome that has more in common with the reaction diffusion pattern that gave rise to it. One could still argue, however, that a pattern resembling coral would have been expected.

4.2 Running RD at the Start

When RD is only run before any growth step happens, the outcome is a lot more predictable in relation to the RD pattern since no RD occurs once the mesh begins to grow. Figure 8 shows and example where bead-like patterns form in the final shape and align with the "holes" of RD concentrations, and ultimately a petal-like pattern is formed around the center. Figure 9 shows another example where layers of reagent concentration leads to growth of ring like folds in the final shape. Figure 7 shows a collection of interesting shapes that we generated by first running RD and then growing shapes with different parameter settings.

Note that the examples above were generated using a 9062 vertex mesh of a disc. Using a smaller disc, i.e., a smaller mesh with with lower resolution different resolution yields a different result as shown in Figure 11 because the RD takes place over a smaller area. We also note that changes in mesh topology influence the the diffusion process and can lead to different results seeded with the same initial conditions.

The main reason for the difference in fact is that, due to the size difference, RD does not have as much time before reaching the outer region of the smaller mesh. When this occurs, the RD pattern typically deteriorates and does not reach a stable configuration. A bigger mesh, however, can have RD run on it much longer, without the RD reaching its outer region which typically leads to more appealing patterns, hence, more appealing growth shapes.

4.3 Limitations

One of the largest challenges is that parameter settings do not always lead to interesting results. Figure 12 shows a density map of where we observe interesting shape growth for different feed and kill values. For a given fixed feed and kill rate, we evaluate combinations of parameter values in Table 1 and note that some cases tend to yield good variation, while other settings produce similar results. Ultimately, the figure shows that some RD parameters allow for a greater variance in their results than others.

Differential growth is likely necessary to produce many of the shapes seen in nature [16]. While our simulation does not implement varying growth on top and bottom surfaces of the thin shell, we believe it would be straightforward to modify our simulation to accommodate this. Another important limitation with modeling shapes with the techniques we present here is the difficulty of controlling the outcome.

An easy first step towards automatically detecting undesired outcomes would be to detect cases where a parameter (a or b) reaches a near 0 value across the mesh in the first few steps of the simulation. For later stages of the simulation, a comparison between the input and output meshes to quantify changes (e.g., number of vertices added, displacement of vertices, and bend at edges) could be helpful to automatically discard a bigger proportion of undesired results.

5 Conclusion and Future Work

We have presented a method that implements Reaction-Diffusion in a mesh to induce its growth in an attempt to mimic the propagation of biochemicals through living tissues. The model allows a variety of parameters to be tuned which in turn allow us to derive various shapes that have an organic appearance.

Overall, the most interesting avenue of future work would be to examine the more difficult inverse modelling problem. That is, to ask if there are simple explanations of shapes and shape variation, e.g., leaves, by physically based growth. There are likewise a number of other ways to extend this work, from straightforward low-level ideas to more complicated efforts.

In future work, we could change our Laplacian calculation to properly use the area of the triangles to make the RD independent of the mesh size, and stable numerical integration methods could be adapted to the RD equations. Furthermore, anisotropy could be included by adding constraints on the directions in which RD can propagate, and this would obviously provide changes to the results. This could be done using polarity fields [5] and would have to be taken into account during the RD step.

Not all combinations of feed and kill values will lead to RD, that is, the RD sometimes reaches an equilibrium by having a single of its reagents (a or b) reach a concentration of 1 throughout the mesh while the other is set to 0. This can be used to control which regions of the mesh grow and which do not by specifically setting their feed and kill values.

We would like to consider using 3D meshes as initial shapes, such as cubes, cylinders, prisms, cones, and other arbitrary shapes. We chose to use the cotangent Laplacian because it can be used on 3D meshes. As it is, the code does not need to be modified to use such meshes. Likewise, we also considered running RD on a regular grid in a texture space of arbitrary shapes and recognize that such a solution could be a fast alternative with a good solution for computing the Laplacian at texture seams.

Finally, feeding an output mesh back into the pipeline would be interesting. The growth speeds up as RD and growth steps go by. Using early stopping to prevent this and restart the procedure could

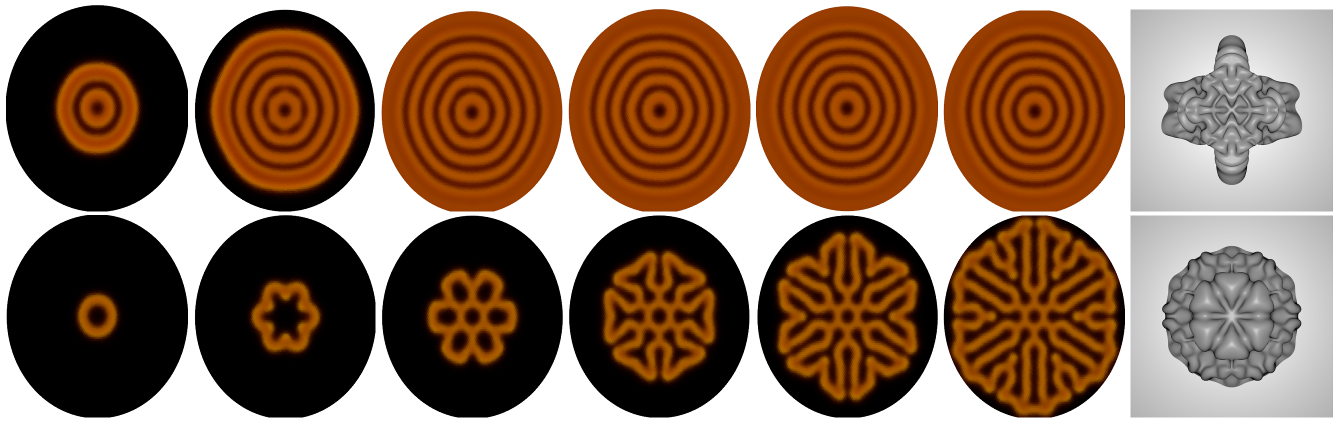


Figure 10: Evolution of the reaction diffusion from left to right in two examples, along with the final outcome at right. Both examples use a polynomial growth function, with the top example using m=0, n=1, γ=0.05, f=0.039 and k=0.058, while the bottom example uses m=0, n=1, γ=0.05, f=0.0545 and k=0.062.

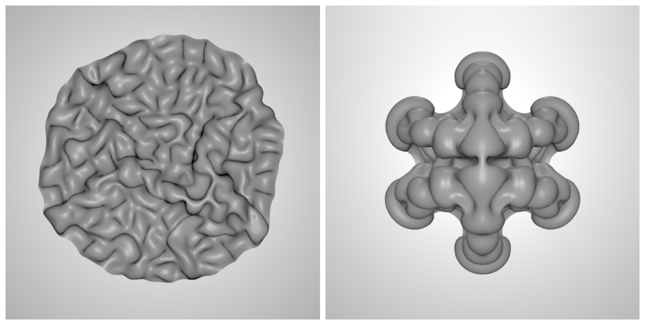


Figure 11: Coarser meshes (left) give a different outcome since the RD depends on the mesh topology. The RD does not inherently behave differently on coarser meshes, it simply reaches the boundaries of it sooner. This restricts how much the pattern can develop which subsequently affects the growth. Both examples use a polynomial growth function with m=0, n=1, γ=0.05, f=0.029 and k=0.057

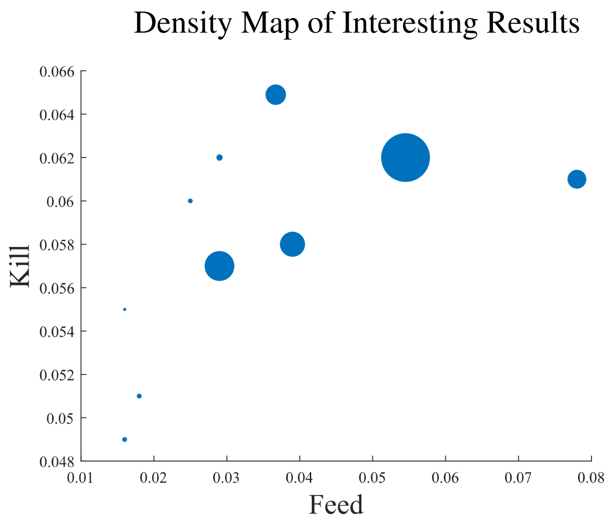


Figure 12: Larger collections of interesting shapes are possible by changing growth parameters at certain fixed feed and kill rate settings.

help control this and possibly add more variance to the results at later stages. Likewise this adds the possibility to restart RD from a different location in the mesh than the initial round which may or may not create interesting results.

References

[1] E. Coen and A. B. Rebocho. Resolving conflicts: modeling genetic control of plant morphogenesis. *Developmental cell*, 38(6):579–583, 2016.

[2] E. J. Fuselier and G. B. Wright. A high-order kernel method for diffusion and reaction-diffusion equations on surfaces. *Journal of Scientific Computing*, 56(3):535–565, 2013.

[3] T. Hutton, R. Munafo, A. Trevorrow, T. Rokicki, and D. Wills. Ready, a cross-platform implementation of various reaction-diffusion systems. *URL https://github.com/GollyGang/ready*, 2015.

[4] W. Kahan. Miscalculating area and angles of a needle-like triangle. lecture notes for introductory numerical analysis classes, 2012.

[5] R. Kennaway, E. Coen, A. Green, and A. Bangham. Generation of diverse biological forms through combinatorial interactions between tissue polarity and growth. *PLoS computational biology*, 7(6):e1002071, 2011.

[6] M. Malheiros and M. Walter. Pattern formation through minimalist biologically inspired cellular simulation. In *Proceedings of Graphics Interface 2017*, GI 2017, pp. 148 – 155. Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, 2017. doi: 10.20380/GI2017.19

[7] R. Narain, T. Pfaff, and J. F. O'Brien. Folding and crumpling adaptive sheets. *ACM Transactions on Graphics*, 32(4):51:1–8, July 2013. Proceedings of ACM SIGGRAPH 2013, Anaheim.

[8] P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.

[9] A. R. Sanderson, R. M. Kirby, C. R. Johnson, and L. Yang. Advanced reaction-diffusion models for texture synthesis. *Journal of Graphics Tools*, 11(3):47–71, 2006.

[10] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Applied computational geometry towards geometric engineering*, pp. 203–222. Springer, 1996.

[11] J. Solomon, K. Crane, and E. Vouga. Laplace-Beltrami: The swiss army knife of geometry processing. In *Symposium on Geometry Processing graduate school (Cardiff, UK, 2014)*, vol. 2, 2014.

[12] J. Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pp. 121–128. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999. doi: 10.1145/311535.311548

[13] J. Stam. Flows on surfaces of arbitrary topology. *ACM Trans. Graph.*,

22(3):724–731, July 2003. doi: 10.1145/882262.882338

[14] A. M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, 1952.

[15] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In *ACM SIGGRAPH Computer Graphics*, vol. 25, pp. 289–298. ACM, 1991.

[16] W. M. van Rees, E. Vouga, and L. Mahadevan. Growth patterns for shape-shifting elastic bilayers. *Proceedings of the National Academy of Sciences*, 114(44):11597–11602, 2017. doi: 10.1073/pnas.1709025114

[17] A. Witkin and M. Kass. Reaction-diffusion textures. *ACM Siggraph Computer Graphics*, 25(4):299–308, 1991.

[18] C. Wu, J. Deng, and F. Chen. Diffusion equations over arbitrary triangulated surfaces for filtering and texture applications. *IEEE transactions on visualization and computer graphics*, 14(3):666–679, 2008.