Interactive Reaction-Diffusion on Surface Tiles

Kui-Yip Lo
The Hong Kong University
of Science and Technology
csfelix@cse.ust.hk

Hongwei Li
The Hong Kong University
of Science and Technology
lihw@cse.ust.hk

Chi-Wing Fu
The Hong Kong University
of Science and Technology
cwfu@cse.ust.hk

Tien-Tsin Wong
The Chinese University of Hong Kong
ttwong@cse.cuhk.edu.hk

Abstract

This paper proposes to perform reaction-diffusion on surface tiles. The square tiles fit nicely and cost-effectively in GPU memory, whereas we also apply distortion minimization on tiles so as to precisely reduce the unbalanced scale and resolution problem of chemicals in the reaction-diffusion. The interconnection nature of tiles accounts for the surface topology, and thus allows the chemicals to flow naturally over surfaces of arbitrary genus. Furthermore, by taking advantage of the tile structure, we can efficiently perform localized reaction-diffusion, and adjust the pattern formulation in an interactive manner on the GPU. To demonstrate its performance, we develop an interactive system that allows texture designers to fine-tune and alter the reaction-diffusion process by directly painting chemicals onto the object surface. Finally, we also develop several non-trivial applications of reaction-diffusion, including the geometry-dependent reaction-diffusion and deformation-aware reaction-diffusion.

1. Introduction

Reaction-diffusion is a mathematical model proposed by Turing [21] to describe the formation of natural patterns. Graphics researchers [7, 9, 22, 23] realized it using computers in 1990's.

A rectangular map is a more convenient domain for performing the grid-structured reaction-diffusion. However, natural patterns are normally formed on irregularly-shaped animal surfaces rather than on a flattened plane. One can map the whole object surface to a single texture map, but if the mapping is not area-preserving, this leads to scale and resolution problems during the simulation of reaction-diffusion. Unless the diffusion rate is anisotropically weighted accordingly, the resultant reaction-diffusion pattern will be overly enlarged or compressed when mapped onto the objects. Existing methods perform reaction-diffusion directly on triangular meshes. Each triangle has to be laid on the texture in an area-preserved fashion. The problem is worsen when the object is not genus-zero in topology as the object may have to be divided. The connectivity at the boundaries of separated parts has to be properly handled as chemicals could flow *across* the boundaries during the simulation.

In this paper, we propose to perform reaction-diffusion on *surface tiles* in order to solve the topology issue. Each tile is in a square shape that fits nicely in the GPU memory. They can be connected in arbitrary topology, see the surface tiles on a genus-three model in Figure 1. Therefore, the tedious topology issue can be solved naturally. The connectivity among tiles can be easily handled by looking up the indirect addresses when retrieving the neighboring chemical values. The distortion-minimized tiles are obtained by parameterizing the object mesh using existing quad-mesh parameterization methods [3, 5, 8, 11, 16, 20].

Another well-known problem of reaction-diffusion is heavy computation cost. This hinders its application to texture design. Without interactive response, texture designers are hard to fine-tune the patterns in order to mimic the real-world counterparts. To make the design of reaction-diffusion textures practical, we developed an interactive system that allows the designers to fine-tune and alter the reaction-diffusion process, hence the pattern, by directly "painting" chemicals (parameters) onto the object surfaces. With the GPU-friendly tile structure, we can achieve high response rate with tailor-made shaders. As reaction-diffusion pattern in equilibrium state is constant, partial modification by designers can be interactively achieved by performing reaction-diffusion *locally* on the affected tiles. This property further speeds up the system response.

The organization of this paper is as follows: Section 2 describes the related work in reaction-diffusion. Section 3 presents our surface tile structure for modeling reaction-diffusion, whereas Section 4 explores the use of this tile structure to develop localized reaction-diffusion on object surfaces. Finally, Section 5 presents various interactive reaction-diffusion examples and some non-trivial applications of the system. Section 6 draws the final conclusion.

2. Related Work

Reaction-diffusion was first proposed by Turing [21] in his paper "The Chemical Basis of Morphogenesis." He described a system of two or more chemical substances diffusing and reacting together through a tissue to address the phenomena of morphogenesis. Although the initial state of the chemicals is homogeneous, the chemicals may finally reach a dynamic equilibrium state, where an organic pattern or structure can be developed as a result. The model of diffuse and react was formulated as a set of partial differential equations governing how the concentration of chemicals evolves over space and time. These "Turing Patterns" were also proved to be able to produce a wide variety of patterns existing in the nature, including the zebra stripes, leopard spots, and web-like patterns on giraffes, etc., referring to the development of several different reaction-diffusion models such as [2, 13, 14]. An overview of reaction-diffusion can be found in [6].

Graphics research pioneers, including Turk [22], Witkin and Kass [23], started to put reaction-diffusion into practice using computers in the early 90's. Later, Fowler et al. [9] applied reaction-diffusion to simulate pigmentation on sea shell models. Fleischer et al. [7] employed the reaction-diffusion to produce cell attributes, and hence, efficiently determined the bump and thorn patterns on various geometry surfaces. Chambers and Rockwood [4] generalized the reaction-diffusion system to be three-dimensional so as to generate solid textures. Stam and Fiume [19] employed reaction-diffusion to model fire and other gaseous phenomena such as wispy smoke and steam, while Phan and Grimm [15] proposed an interactive interface for sketching reaction-diffusion textures using machine learning.

In recent years, there are two major research directions related to reaction-diffusion. One of them is to employ reaction-diffusion as a visualization tool. Kindlmann et al. [12] employed anisotropic reaction-diffusion to illustrate the structure in volumetric diffusion data. Sanderson et al. [17] further applied reaction-diffusion to generate texture patterns with variable shape, size, and orientation based on the uncertainty in the data domain. The other research direction focuses on applying the GPU to accelerate the reaction-diffusion computation. This includes the work of Harris et al. [10], who employed the coupled map lattice (CML) to build a framework to perform various simulations including reaction-diffusion on graphics hardware, and the work of Sanderson et al. [1], who developed a GPU implementation of an extended reaction-diffusion model that allows users to explore the pattern generation. While these are indeed exciting research work in recent years, it is worth to note that a mapping from 3D surface to a 2D map is still required in order to make reaction-diffusion computable on the GPU.

3 Reaction-Diffusion on Surface Tiles

Fundamentally, Turing's reaction-diffusion model considers two chemicals diffusing and reacting through a tissue to account for the phenomena of morphogenesis. The concentration of the two chemicals are given by a and b, and are continuously regulated over the space and time according to the following partial differential equations:

$$\begin{aligned} \frac{\partial a}{\partial t} &= F(a,b) + \alpha_a \nabla^2 a \\ \frac{\partial b}{\partial t} &= G(a,b) + \alpha_b \nabla^2 b \,, \end{aligned}$$

where F and G are functions controlling the production (or the reaction) of a and b, respectively, α_a and α_b are the diffusion rate of a and b, respectively (assuming isotropic diffusion), and $\nabla^2 a$ and $\nabla^2 b$ are the Laplacians of a and b over the spatial domain, respectively. Even though the initial concentrations of a and b could be the same over the whole tissue, equilibrium state may finally be reached and a stable organic pattern could be produced at the end.

To practically compute this reaction-diffusion model, Turing further defined a two-dimensional discrete grid structure, as the spatial domain, to quantitatively store the concentration of the chemicals (per square grid cell). Furthermore, by rewriting the partial differential equations in discrete form, we can numerically compute the PDE over the grid cells, and thus can iteratively compute reaction-diffusion directly using computers. Other than using 2D grid structure, Turk [22] proposed to directly compute reaction-diffusion on 3D object surfaces by constructing highly uniform meshes on the object surfaces. Using this design, reaction-diffusion can be extended to work on surfaces of arbitrary shapes, and organic texture patterns can be iteratively generated on the mesh structure.

With the advances in programmable graphics hardware, graphics researchers started to employ the GPU to speed up the computation of reaction-diffusion [1, 10]. However, in order to perform computation on the GPU, the data has to be packed in a rectangular structure, so that the data can be stored (read and write through the shaders) and computed in the GPU textures. Hence, Turk's mesh-based computation model cannot be directly adopted on the GPU.

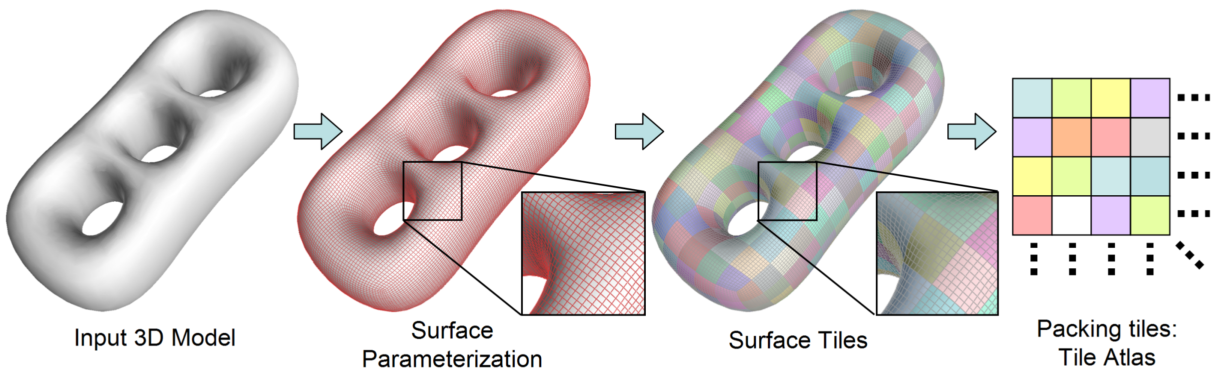

Figure 1. Tile-based reaction-diffusion on HOLES3 model: 1) Laying tiles on its surface parameterization, 2) Packing tiles into a 2D tile atlas, and 3) Computing reaction-diffusion on the tile atlas.

To efficiently compute reaction-diffusion on surface of 3D objects, while taking the advantage of the GPU to accelerate the computation, we introduce a tile-based computation model for performing reaction-diffusion on 3D objects. First, we employ surface parameterization models and arrange surface tiles on parameterized surfaces. Surface parameterization re-parameterizes object surfaces using quad structure. With this new parameterization, we can efficiently apply texture mapping onto the surface as well as perform a wide range of surface modeling tasks. Several ingenious methods [3, 5, 8, 11, 16, 20] were recently proposed to create surface parameterization.

Surface Tiles Based upon these methods, the very first step in our approach is to take a surface parameterization as the input, and lay a set of interconnected surface tiles over the parameterized surface. Note that *each surface tile is a square tile containing $n \times n$ texels on the object surface*. By this means, we gain the following advantages over previous reaction-diffusion models:

- First, we can nicely pack the square tiles into a rectangular domain, known as the *tile atlas*, and store them in the texture memory. Thus, we can take advantage of the GPU to compute reaction-diffusion on this tile-atlas domain, see Figure 1 for the illustration.
- At the same time, since the interconnected tile structure allows us to cover object surfaces of arbitrary genus, this tile-based computation model can be nicely performed on object surface of arbitrary genus.
- Besides, tiles can naturally divide the surface into disjoint areas, where we can selectively perform localized reaction-diffusion for speed-up. This is especially useful and practical when texture designers modify only part of the texture in equilibrium state, see Section 4.

In our particular implementation, we use 3D models from the following two surface parameterization methods: Periodic global parameterization [16] and PolyCube-Maps [20]. Nevertheless, any low-distortion and quad-based parameterization can be adopted in our application.

Reaction-Diffusion on Surface Tiles Before computing reaction-diffusion on surface tiles, we first have to prepare a *texel connectivity map* to store the connectivity between neighboring texels in the tile-atlas domain. This is an essential step in our surface-tile model because our model has to handle object surfaces of arbitrary genus. In details, this texel connectivity map is defined in the tile-atlas domain, and each texel in this map holds the texture coordinates (in the tile-atlas domain) of its four direct texel neighbors on the object surface. Hence, to perform reaction-diffusion on the GPU, we can apply indirect addressing to locate the neighbors of any texel on the tile atlas, see Section 5 for the details of the implementation.

Figure 2 depicts an example reaction-diffusion sequence on a genus-three model: the HOLES. Starting from a homogeneous chemical concentration over the entire object surface, we can iteratively compute reaction-diffusion *in parallel over all texels* on the object surface, and gradually reach the equilibrium state in around three to four seconds.

Distortion Minimization Furthermore, to minimize the distortion on the reaction-diffusion computation caused by the non-uniformity in the surface parameterization, we adapt the method developed by Witkin and Kass [23]; given $\mathbf{x}(u, v)$ as the mapping from local uv-parametric space to the object space, we first compute the Jacobian of $\mathbf{x}$, say J, locally over the object surface. Then, we can compute the metric tensor as a two-by-two matrix:

$$M = J^T J .$$

Finally, by locally adjusting the diffusion rate (diffusion matrix) accordingly, we can minimize the distortion caused by the surface parameterization. Figure 3 demonstrates the

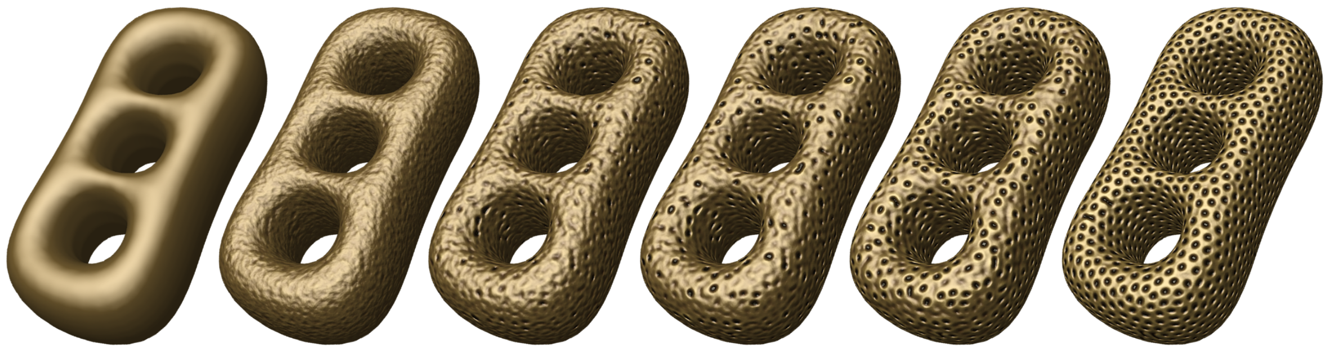

Figure 2. Computing reaction-diffusion globally over all tiles on the HOLES model (genus-three); It takes around 3-4 seconds to go from an initial homogeneous state to a final equilibrium state.

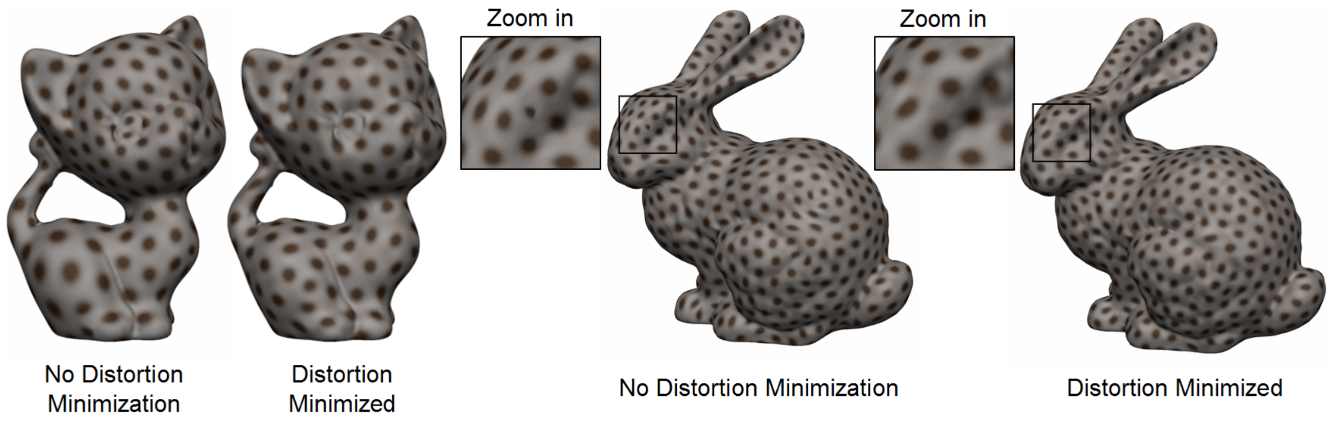

Figure 3. Results of distortion minimization: KITTEN (left) and BUNNY (right).

related results; by defining a map of diffusion rate in the tile-atlas domain and by locally adjusting the diffusion rate, we can normalize the pattern scale and compensate the surface distortion. In the figure, two surface parameterization models are employed in our experiment: the KITTEN model from Periodic global parameterization [16] and the BUNNY model from PolyCube-Map [20]. Note that in the surface parameterization of the KITTEN model, the surface texels on its body are larger than those on its head, while in the BUNNY model, the surface texels around its eyes are relatively smaller. Without distortion minimization, we can see from the figure that Turing's spot pattern has a wide range of sizes on the object surface due to the non-uniformity in the surface parameterization. With distortion minimization, the spot pattern becomes more uniform. Note that a similar method (surface metric) to account for the effect of surface distortion can be found in [18].

4 Localized Reaction-Diffusion

As mentioned in Section 3, the interconnected tile structure can also be served as a spatial data structure on the object surface to allow us to explore the locality in the reaction-diffusion computation.

Our observation with interactive control on reaction-diffusion is as follows. Once the reaction-diffusion process reaches an equilibrium state, the chemical concentration over the entire spatial domain becomes stable. If we modify the chemical concentration only on part of the object surface, for example, by distorting or painting chemicals on certain surface regions, or by deforming certain part of the object, most regions are not affected by the action, while the effect of the action will only be gradually propagated outward from the affected surface area. Hence, by having an interconnected tile structure over the object surface:

- First, we can precisely identify the surface regions affected by user's action as a set of surface tiles covering (or neighboring to) the related surface area.
- Hence, we can quickly determine an active set of tiles, known as the *active tile set*, so that we can compute reaction-diffusion locally on this tile set. As the effect of the user action propagates outward, we can progressively expand this tile set.

Tile Neighbors Given a set of n interconnected surface tiles, say $\mathcal{T}$, fully covering the surface parameterization:

$$\mathcal{T} = \{ \mathcal{T}_1 , \mathcal{T}_2 , \dots , \mathcal{T}_n \} ,$$

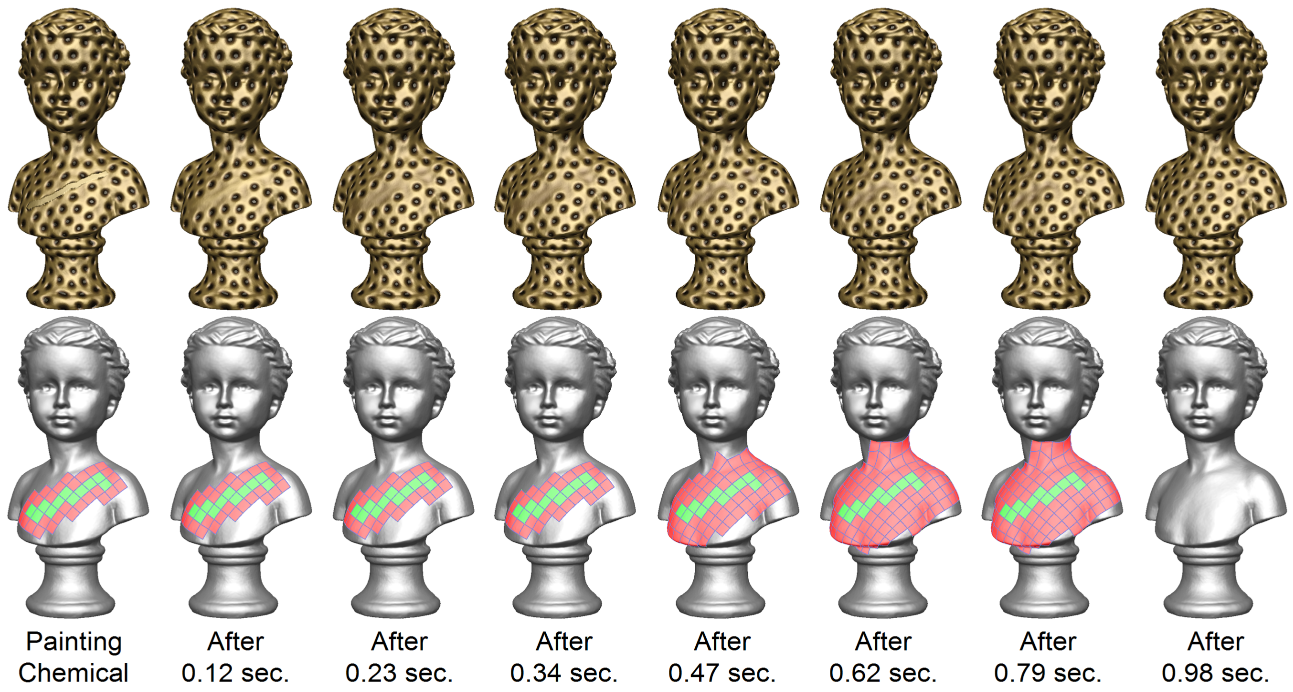

Figure 4. Distorting the equilibrium pattern on the BUSTE model; Reaction-diffusion patterns can be evolved at interactive speed by using localized reaction-diffusion computation on the active tile set.

where $\mathcal{T}_i$ is the ith surface tile in the set. We define the set of neighboring tiles of $\mathcal{T}_i$, say $\mathcal{N}(\mathcal{T}_i)$, as

$$\mathcal{N}(\mathcal{T}_i) \;=\; \{\, \mathcal{T}_j \mid \mathcal{T}_i \,\cap\, \mathcal{T}_j \,\neq\, \emptyset \text{ and } i \,\neq\, j \,\}\,.$$

Note that if two distinct tiles shares a vertex on the object surface, they are said to be neighbor of each other. Further than that, we can also define the neighboring function $\mathcal{N}$ on a set of tiles. Given $\mathcal{S} \subseteq \mathcal{T}$, we can define

$$\mathcal{N}(\mathcal{S}) \;=\; \{\, \mathcal{T}_j \mid \mathcal{T}_j \in \mathcal{S} \text{ or } \exists\, \mathcal{T}_i \in \mathcal{S} \text{ s.t. } \mathcal{T}_j \in \mathcal{N}(\mathcal{T}_i) \,\}\,.$$

Local computation The localized reaction-diffusion algorithm can be described as the following three steps: 1) initializing the active tile set, 2) expanding the active tile set, and 3) transiting from local computation using the active tile set to global computation over all tiles:

- After the user takes an action, say painting chemicals onto the object surface, we first identify a set of surface tiles on the object, say $\mathcal{A}_0$, that can completely cover the entire affected surface region.
- Then, we compute the initial active tile set:

  $$\mathcal{A}_1 \;=\; \mathcal{N}(\mathcal{A}_0)\,,$$

  and compute reaction-diffusion locally on tiles in $\mathcal{A}_1$.
- Furthermore, for any tile, say $\mathcal{T}_i \in \mathcal{T}$, by pre-simulation using the map of diffusion rate, we can pre-compute the number of iterations it could possibly take to spread the reaction-diffusion effect to its first neighbor set $\mathcal{N}(\mathcal{T}_i)$, to its second neighbor set $\mathcal{N}(\mathcal{N}(\mathcal{T}_i))$, and so on. Hence, we can determine the number of iterations that could possibly be taken for the reaction-diffusion effect to go beyond $\mathcal{A}_1$; by this means, we can iteratively expand the active tile set:

  $$\mathcal{A}_{i+1} \;=\; \mathcal{N}(\mathcal{A}_i)\,,\; i \;\geq\; 1\,.$$
- Finally, when the number of tiles inside the active tile set goes beyond a certain threshold, we can clear the active tile set, and revert to global computation mode over all tiles on the tile atlas.

Figure 4 presents an example of interactive reaction-diffusion; here we distort the chemicals on the surface of the BUSTE model (from Periodic global parameterization [16]) and watch it going back to equilibrium. The upper row in the figure shows an image sequence of evolving patterns rendered with bump mapping, while the bottom row reveals the corresponding active tile set; the tiles that are initially marked down by users' action ($\mathcal{A}_0$) are labelled in green, whereas the active tile set at any moment include the green tiles as well as the red tiles. Note also the time taken to reach each state (marked on the bottom of each column); in the last column, since the active tile set reaches the pre-defined threshold, we clear the active tile set and revert to global computation mode from there.

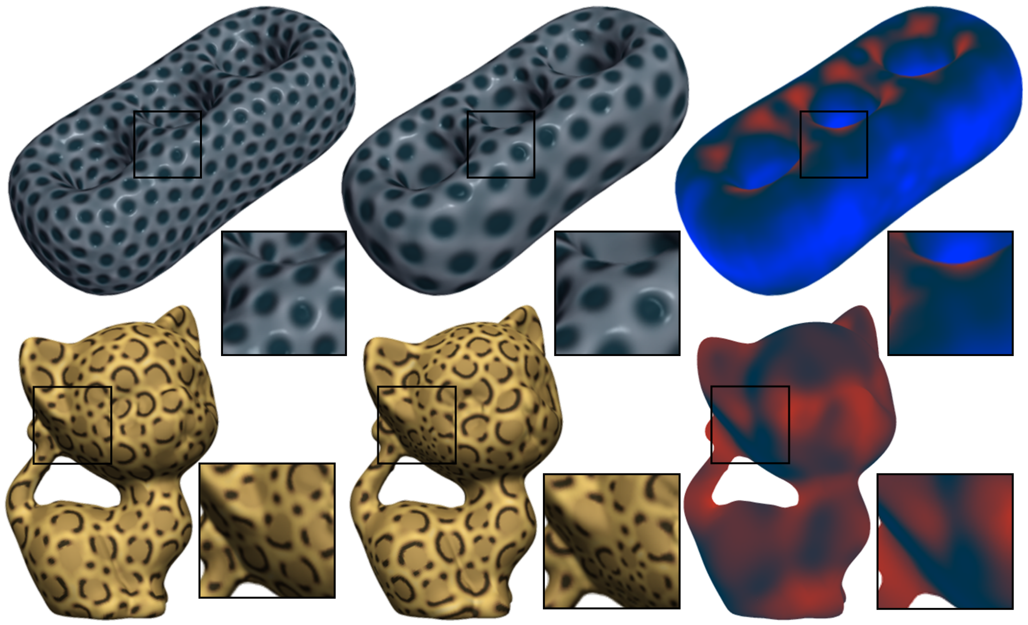

Figure 5. Geometry-Dependent Reaction-Diffusion: HOLES3 **model (top) and** KITTEN **model (bottom).**

5 Implementation and Results

5.1 Implementation details

In the implementation of the tile-based reaction-diffusion system, we employ the following hardware configuration in our experiment:

- An HP xw4400 workstation with Intel Core(TM)2 CPU 6400 at 2.13GHz and 1GB Memory
- Two NVidia Geforce 8800 GTX graphics boards (each with 768MB memory) running in SLI mode

For the software part, we employed OpenGL and Cg 2.0. Several texture maps defined on the tile-atlas domain are employed: two RGBA float textures for storing the concentration of chemicals (in case we use Turing's system with double buffering), one RGBA unsigned integer (GL_RGBA32UI_EXT) texture for the texel connectivity map, and one RGBA float texture for the map of diffusion rate. Note that we can compactly pack each uv-texture coordinate as a single 32-bit unsigned integer for efficient texture fetch. Hence, we can store four uv-texture coordinates (four neighbors) as an RGBA tuple per texel on the object surface. Using this scheme, we can efficiently unpack them using the fragment program code below:

```
// Unpack uv-coord. of texCoord0's four neighbors
uvec4 addr           = texRECT(conn_map, texCoord0);
uvec4 addr_left_right = addr >> 16;
uvec4 addr_up_down    = addr & 0x0000FFFFU;
```

Furthermore, we compute the reaction-diffusion in an offscreen manner using the OpenGL extension GL_EXT_framebuffer_object. The resolution of the framebuffer object equals the resolution of the tile atlas, and all tile-atlas-based textures mentioned above are attached to this framebuffer object for efficient data lookup. Hence, fragment programs can be used to perform the computation iteratively.

In case of global computation over the entire tile atlas, we render only a single rectangle to cover the entire viewport in the framebuffer object, so that we can trigger fragment programs to compute reaction-diffusion for all surface texels. In case of localized reaction-diffusion, within a single iteration cycle, we render only the tiles (in a tile by tile manner on the tile atlas) in the active tile set. Hence, we can minimize the number of surface texels to be triggered, and hence significantly reduce the amount of fragment processing in each iteration cycle.

5.2 Geometry-dependent Reaction-Diffusion

By adjusting the diffusion rate locally according to the metric tensor of the surface parameterization, we can regulate the pattern scale on 3D models. Further than that, we can also control the pattern scale locally on the 3D models by adjusting the pattern scale based on some kinds of local geometry property, for example, the mean curvature.

Figure 5 demonstrates the effect of using the mean curvature to adjust pattern scales on two different 3D models. The last column shows the mean curvature map; red and blue colors correspond to high and low curvature values, respectively. Models in the first column are rendered in homo-

Table 1. Performance Comparison of Global and Localized Reaction-Diffusion.

				Global Reaction-Diffusion		Localized Reaction-Diffusion						
						Phase 1: Using A1		Phase 2: Using A2		Phase 3: Using A3		
Model	Total number of tiles	Resolution of a tile	Resolution of tile atlas	# iterations per sec.	Time (sec.) to reach equilibrium	# iterations per sec.	# tiles in active tile set	# iterations per sec.	# tiles in active tile set	# iterations per sec.	# tiles in active tile set	Time (sec.) to reach equilibrium
Holes	400	16x16	320x320	6054	1.925	17344	31	15145	67	8174	121	1.525
		32x32	640x640	1755	2.549	10525	30	6999.8	68	4651	125	1.754
Bunny	1288	8x8	288x288	7149	2.741	15094	24	16909	50	8561	83	2.470
		16x16	576x576	2133	5.903	14086	24	12719	50	7645	83	4.194
Laurana	212	16x16	240x240	10064	1.877	20025	24	11351	51	8760	83	1.069
		32x32	480x480	3077	2.159	13942	24	9580	51	6749	84	1.301
Buste	768	16x16	448x448	3423	2.493	16041	21	14965	45	7935	79	1.108
		32x32	896x896	906	4.018	9027	24	6949	50	5296	86	1.343
Kitten	463	16x16	352x352	5292	4.964	18627	19	17242	43	8341	80	4.676
		32x32	704x704	8341	13.504	11831	19	8702	43	6031	80	8.571

geneous scale, whereas models in the middle column have their pattern scales adjusted by the mean curvature; here, patterns around high-curvature regions are adjusted to be smaller in size. This allows us to generate fine details on smaller geometric features, usually with high curvature.

5.3 Localized Reaction-Diffusion

To demonstrate the efficiency of localized reaction-diffusion, we experiment localized reaction-diffusion and global reaction-diffusion (over the whole tile atlas) on five different 3D models, each with two different resolutions of surface parameterization. Table 1 presents the results; the 2nd to 4th columns describe the 3D model property, including the total number of surface tiles on it, the resolution (number of texels) of a tile, and the resolution of the tile atlas; the 5th to 6th columns describe the performance of global reaction-diffusion, including the number of iterations per second and the average time taken to reach an equilibrium state for a particular painting action; the rest of the columns describe the performance of localized reaction-diffusion: the last column shows the average time taken to reach the equilibrium state for the same action taken in the case of global computation, whereas the other six columns are divided into three parts. Each part corresponds to a particular phase corresponding to the expansion of the active tile set: from $\mathcal{A}_1$, to $\mathcal{A}_2$, and then to $\mathcal{A}_3$. Here, each part contains two columns, including the number of iterations per second we can achieve in that phase as well as the number of tiles in the active tile set.

From the table, we can see that localized reaction-diffusion can always outperform global reaction-diffusion because we can always perform more reaction-diffusion iterations using the same amount of computation time; hence, equilibrium states can be reached more promptly. In addition, we can also compare the relative performance between localized reaction-diffusion and global reaction-diffusion; here, the larger the size of the tile atlas, the more we can usually gain with localized reaction-diffusion, for example, note the row for the BUSTE model.

5.4 Interactive Painting Control

In our interactive reaction-diffusion system, we support the following modes of interactive painting control:

- *Locally distorting an existing reaction-diffusion pattern.* This action allows users to locally alter an equilibrium state and randomly reach some other equilibrium states, see the image sequence in Figure 4.
- *Painting chemicals by adding or removing chemicals permanently on a particular region on the object surface.* This action allows us to put in constraints in the spatial domain; in practice, these constraints can be letters or symbols hidden in the reaction-diffusion patterns, see the left two LAURANA renderings in Figure 6; two hidden letters `PG` are painted onto the strip pattern by removing some chemicals locally.
- *Controlling the pattern scale by painting.* Furthermore, the painting action can also be used to adjust the diffusion rate locally on object surface; hence, we can locally adjust the pattern scale, see the ring of smaller-scaled patterns deliberately put around the neck of LAURANA in Figure 6 (right).

5.5 Deformation-Aware Reaction-Diffusion

In addition, we can also interactively perform reaction-diffusion while locally deforming 3D models. The underlying mechanism is basically the same as that of interactive painting; when only a portion of the 3D model is being deformed, we determine the active tile set for the related surface region, and locally compute reaction-diffusion on the active tile set. Figure 7 presents an example sequence. Here, we first stretch the ear of KITTEN, and wait until the equilibrium. After around two to three seconds, we reverse the action and compress the ear. From the figure, we can see that the reaction-diffusion pattern on the ear can lively (and "organically") adapt to the deformation. Note that we

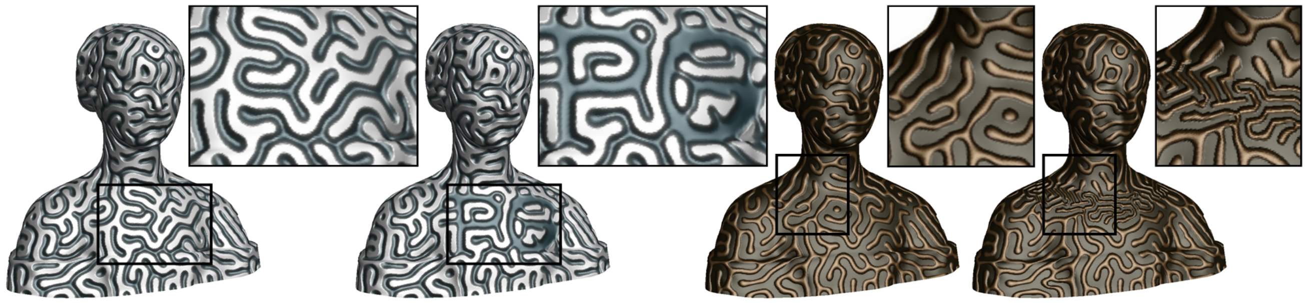

Figure 6. Interactive Painting Controls: Writing some hidden letters in the strip pattern (left) and locally controlling the pattern scale (right) on LAURANA model.

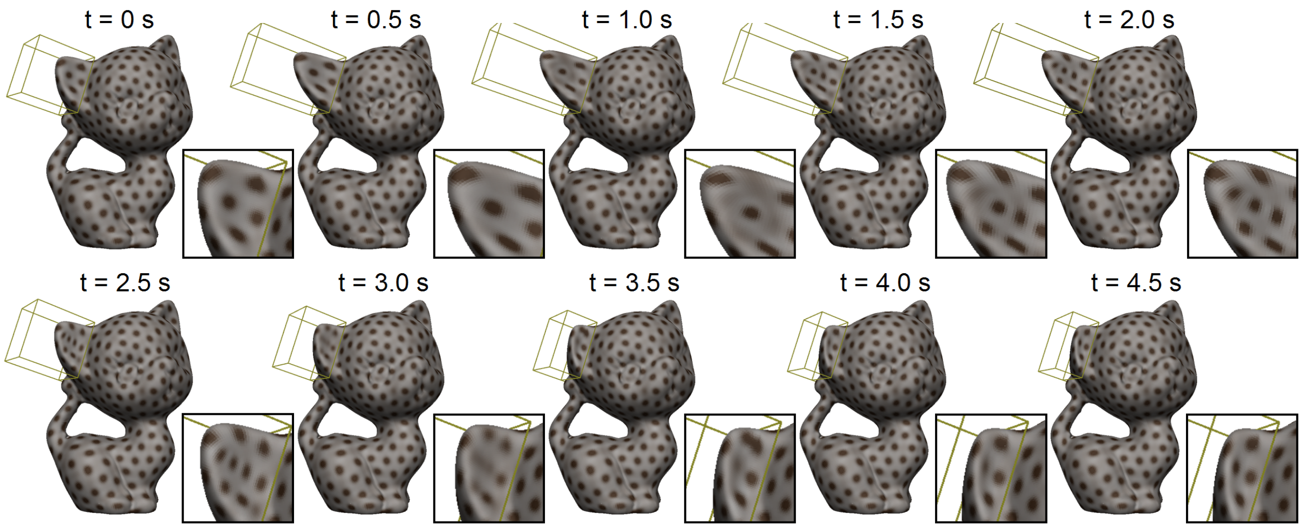

Figure 7. Interactive Deformation of KITTEN's ear: Reaction-Diffusion can be performed lively with the deformation action and equilibrium state can be reached within a few seconds.

have to recompute the Jacobian (and metric tensor) locally over the deformed area on the surface.

5.6 Other Applications

Water Flowing on Surface By customizing our reaction-diffusion model, we can simulate water-drop flowing over object surfaces. In details, we first add an RGB float texture storing object-space position per texel in the tile-atlas domain and attach it to the framebuffer object. To influence water diffusion by gravity, we can compute in the fragment program the directional vector, say $\vec{v}_{ij}$ between neighboring texels, and then apply $\vec{g} \cdot \vec{v}_{ij}$ to adjust the diffusion rate, where $\vec{g}$ is the gravity direction.

Furthermore, to simulate water absorption on object surface, we define a stickiness term to represent the absorption power of the paper/surface, so that we can produce traces of water flow on the surface. Finally, environment mapping (specular reflection) and bump mapping are used to shade the water drops, see Figure 8 for the simulation of a water drop flowing on the KITTEN model. By means of localized diffusion, this animation takes only three seconds.

Water Color Painting on Surface With additional customization, our reaction-diffusion system can be further applied to simulate water color painting on 3D models. First, to simulate the effect of water color painted on paper surface, the stickiness term we used here is no longer a constant over the entire object surface as in the case of water flow. Rather, it is a per-texel variable locally representing the absorption power of the surface. Furthermore, for efficiency in texture fetch, this per-texel stickiness term is stored in the w-component of the chemical-concentration texture. See Figure 9 for some example drawings by the water color painting tool that built on top of our reaction-diffusion system.

6 Conclusion and Future work

Reaction-diffusion can model a rich class of textures and phenomena. However, due to the high computational cost, interactive manipulation of reaction-diffusion textures has long been a difficult task. The proposed tile-based reaction-diffusion allows partial update to the reaction-diffusion textures by computing only those active tiles. Such cost-effective computation enables the high response rate of the system. We demonstrated the system efficiency by showing how texture designers can interactively paint the chemicals, deform the geometry, water color on surface, and then see how the texture patterns react. The computation of tile-based reaction-diffusion fits nicely on the GPU as its structure is rectilinear. Even the geometry is not genus-0, the interconnected tiles can cost-effectively represent the geometry without over-compression or over-stretching during the mapping.

One possible extension to the current framework is to support more general computation models to realistically simulate various types of natural phenomena on surfaces.

Limitations Since we work on surfaces of arbitrary topology, the input surface parameterization could contain any type of corner: 3-corner, 5-corner, or even 6-corner (note: k-corners are vertices in the parameterization shared by k surface texels). Here, certain irregularity could result on texels around these corners, note the circular spots on the HOLES3 model in Figure 5.

Acknowledgments We would like to express our gratitude to Ka-Ling Fok, for his initial experimental implementation of the reaction-diffusion system. This research was supported by the Research Grants Council of the Hong Kong Special Administrative Region, under RGC Earmarked Grants (Project No. HKUST612706 and CUHK417107).

References

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

[2] J. Bard. A model for generating aspects of zebra and other mammalian coat patterns. *Journal of Theoretical Biology*, 93(2):363–385, November 1981.

[3] I. Boier-Martin, H. Rushmeier, and J. Jin. Parameterization of triangle meshes over quadrilateral domains. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 193–203, 2004.

[4] P. Chambers and A. Rockwood. Visualization of solid reaction-diffusion systems. *IEEE Computer Graphics and Applications*, 15(5):7–11, 1995.

[5] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Spectral surface quadrangulation. *ACM Transactions on Graphics*, 25(3):1057–1066, 2006.

[6] I. Epstein and J. Pojman. *An Introduction to Nonlinear Chemical Dynamics*. Oxford Univ. Press, New York, 1998.

[7] K. W. Fleischer, D. H. Laidlaw, B. L. Currin, and A. H. Barr. Cellular texture generation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 239–248, 1995.

[8] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 157–186. Springer Verlag, 2005.

[9] D. R. Fowler, H. Meinhardt, and P. Prusinkiewicz. Modeling seashells. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 379–387, 1992.

[10] M. J. Harris, G. Coombe, T. Scheuermann, and A. Lastra. Physically-based visual simulation on graphics hardware. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 109–118. Eurographics Association, 2002.

[11] L. Kharevych, B. Springborn, and P. Schröder. Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics*, 25(2):412–438, 2006.

[12] G. Kindlmann, D. Weinstein, and D. Hart. Strategies for direct volume rendering of diffusion tensor fields. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):124–138, November-December 2000.

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

[14] J. Murray. *Mathematical Biology*. Springer-Verlag, New York, 1989.

[15] L. Phan and C. Grimm. Sketching Reaction-diffusion texture. In *Eurographics Sketch Based Interfaces and Modeling workshop*. Eurographics, September 2006.

[16] N. Ray, W. C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Transactions on Graphics*, 25(4):1460–1485, 2006.

[17] A. R. Sanderson, C. R. Johnson, and R. M. Kirby. Display of vector fields using a reaction-diffusion model. *Proceedings of the conference on IEEE Visualization 2004*, pages 115–122, 2004.

[18] J. Stam. Flows on surfaces of arbitrary topology. *ACM Transactions on Graphics*, 22(3):724–731, 2003.

[19] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 129–136, 1995.

[20] M. Tarini, K. Hormann, P. Cignoni, and C. Montani. PolyCube-Maps. *ACM Transactions on Graphics*, 23(3):853–860, 2004.

[21] A. Turing. The chemical basis of morphogenesis. *Royal Society of London Philosophical Transactions Series B*, 237:37–72, Aug. 1952.

[22] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 289–298, 1991.

[23] A. Witkin and M. Kass. Reaction-diffusion textures. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 299–308, 1991.

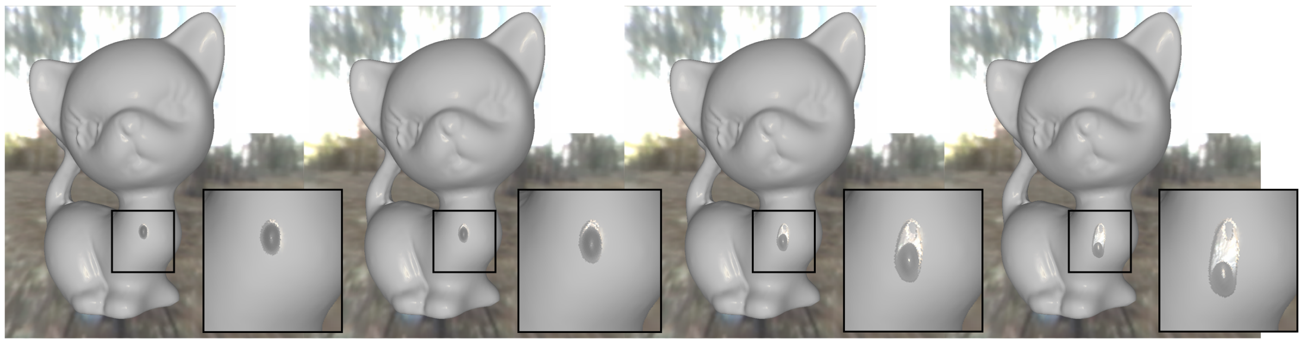

Figure 8. A water drop flowing down on KITTEN. The four snapshots (from left to right) are successively taken at a time interval of one second.

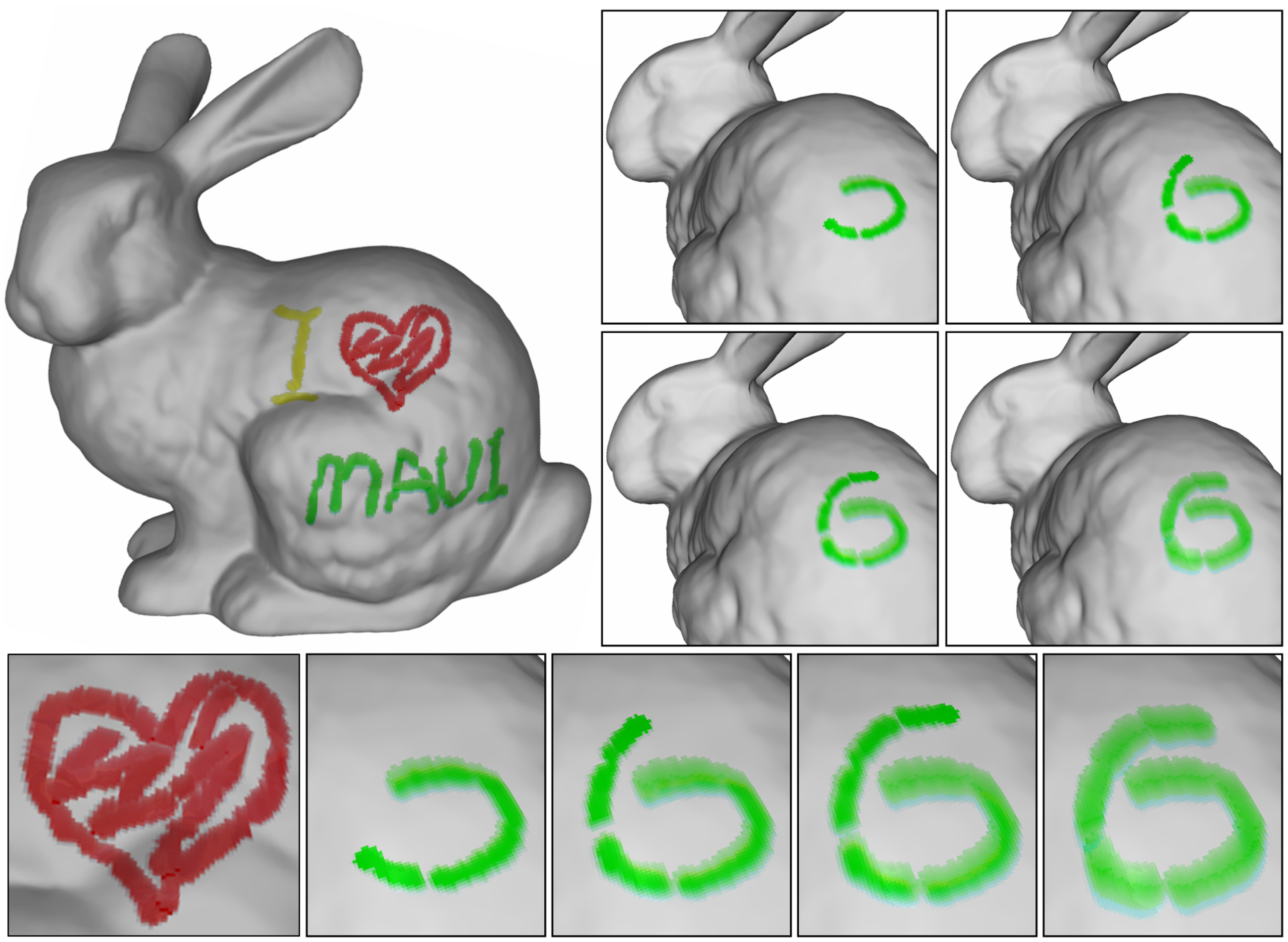

Figure 9. Water color painting on BUNNY. The four successive images on the right hand side are captured 0.5 seconds one after another. Here, the two chemicals in the reaction-diffusion system are used to represent water concentration and pigment concentration; because of the difference in their diffusion rates and also of the variable stickiness, we can mimic water flow and pigment flow, and produce the diffusion pattern around the brush stroke. Note that the more water the user specified in the brush, the more the amount of diffusion could be resulted in the simulation.