3D Software Interpretation of Reas's Processes

Tomasz Sebastian Rybiarczyk
University of Illinois at Chicago
Dept. of Computer Science
Chicago, IL, United States
trybia2@uic.edu

ABSTRACT
The emergence has long been a fascinating process exploited in arts and sciences. Casey Edwin Barker Reas heavily builds upon this phenomenon in his creations, where the compositions of various forms and behaviors give rise to images of highly distinct and mesmerizing shapes. This work describes an attempt to extend his work to the third dimension. Presented are the design decisions and trade-offs that had to be taken to best approximate the original work in 3D. The implementation of the work is also detailed to provide insights into presenting the process of emergence in interactive framerates. Selected original works are compared against the results as a mean of evaluation.

ACM Classification Keywords
I.3.3. Computer Graphics: Picture/Image Generation

Author Keywords
Computer Graphics; Design; Performance

INTRODUCTION

The notion of emergence has a long history. It was as early as the 19th century when John Stewart Mill claimed that a combined effect of several causes cannot be reduced to its component causes [3]. Steven Johnson in his book, "Emergence", gives the defines the emergence as follows:

> In the simple terms (emergent systems) solve problems by drawing on masses of relatively stupid elements, rather than a single, intelligent "executive branch". They are bottom-up systems, not top-down [3].

Many people, inspired by the idea of emergence began producing work of art incorporating it. The phenomena found its outlets also in computer-generated art. Casey Reas is another artist who grasped the concepts of emergence, defined a number of processes and developed software to produce 2D images which are "software interpretation of Process descriptions" [6].

Despite the fascinating effects that come to life from various software interpretations authored by Reas, there was no attempt to interpret his descriptions in a three dimensional universe. This paper documents the first such attempt. Early experiments were meant to approximate the original work as close as possible. Additional goal was to allow the viewers to observe the emergence process from any perspective in interactive framerates. This early work produced quite promising results.

The paper provides brief introduction to th original work of Casey Reas. Next, it documents the issues we considered when extending the original work to 3D. The implementation considerations in sufficient detail are also provided. Lastly, we present our results and compare them to images of Reas's original work (Figure 2).

ORIGINAL WORK

Casey Reas's vast portfolio contains many distinct compositions. Each one of them represents software visualization of various phenomena and draws inspiration from many sources. In this paper we focus on the series of *Processes*, composed of many *Elements*, which in turn encompass *Forms* and *Behaviors*. The definitions are provided in the following subsections.

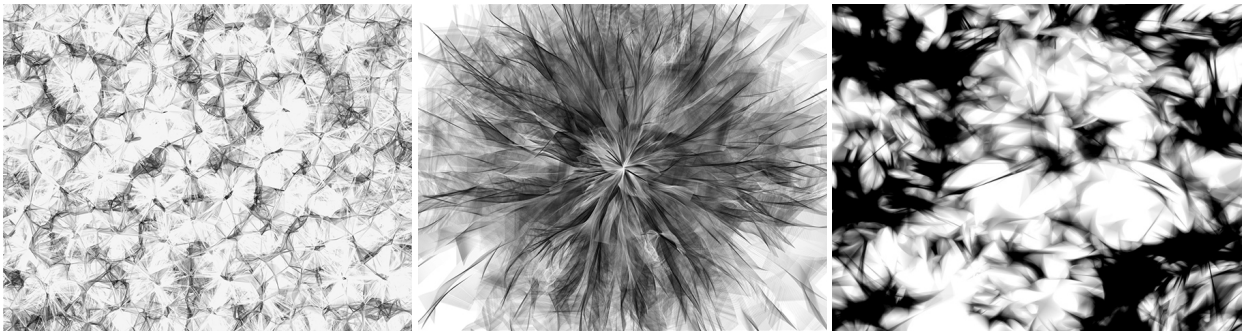
Process Definitions

Reas describes the *Element* as a machine composed of a *Form* and one or more *Behaviors*. The most common Forms in his work are circles and lines. Behaviors determine how the Element reacts certain events such as collisions (and overlap) with other Elements, collisions with universe boundaries, advancement paths. The Process is an environment for Elements and describes how to visualize the interactions between the Elements and other events involving the Elements in a universe. The description of each Process is given in English and its interpretation is left for the programmer.

Examples

For completeness, several Processes are described below. These descriptions are taken directly from Reas's *Process Compendium* text [6]. Their original software interpretations are illustrated in Figure 1.

- Process 4 - A rectangular surface filled with varying sizes of Element 1. Draw a line from the centers of Elements that are touching. Set the value of the shortest possible



(a) Process 4 (b) Process 10 (c) Process 18

Figure 1: Original software interpretation of selected Processes

line to black and the longest to white, with varying grays representing values in between.

- Process 10 - Position a circle at the center of a rectangular surface. Set the center of the circle as the origin for a large group of Element 1. When an Element moves beyond the edge of its circle, return to the origin. Draw a line from the centers of Elements that are touching. Set the value of the shortest possible line to black and the longest to white, with varying grays representing values in between.
- Process 18 - A rectangular surface filled with instances of Element 5, each with a different size and gray value. Draw a quadrilateral connecting the endpoints of each pair of Elements that are touching. Increase the opacity of the quadrilateral while the Elements are touching and decrease while they are not.

CROSSING THE DIMENSION BOUNDARY

As noted by Reas, it is up to an individual how one interprets the Process description. It becomes apparent that extending the ideas to the third dimension, there are even more occasions to employ personal judgment on design. This section explains some of the notable difficulties that had to be addressed in the process.

Representation in 3D

Due to the fact that the work is the first attempt in 3D interpretation of Reas' work, it was of great importance to retain the same interface for producing the images. As the original work involves drawing lines for the most part, the same primitive is used in this work. As seen in the previous section, some Elements consist of circles and the other consist of lines. In this approach, for obvious reasons, the use of circles was substituted with spheres. The use of lines remained in place for other Forms.

Interpretation of Process Text for 3D

An additional design decision that had to be taken was the interpretation of some of the Behaviors. For instance, Behavior 3 changes a direction of the Element. It is not obvious how the direction should be modified in this case. In the original work, as shown in the Compendium Lecture video [5], rotates the Elements clockwise at the same constant rate. In 3D however, this kind of behavior would not result in a desired effect. Our approach generates normal vector for each pair of colliding Elements. Cross product of the normal and the direction vectors is computed to produce the tangent. This vector then serves as an axis around which the direction vector is rotated. The angular velocity is kept constant for all Elements in a Process but it is assigned different value on per process basis.

Behavior 4 moves the Element away from an Element it overlaps. While this description seems straight forward, in practice it is not known at what rate should the Elements be pulled away from each other and how does this rate correspond Elements' velocity. It is also important to maintain the overlap state for a certain duration in order to produce meaningful results in context of visual representation. In our approach, each Element maintains a list of elements it collides with in current discrete step. The list capacity is limited to a certain number (currently 4) so that some amount of overlap conditions occur. The mean of the directions from current Element to colliding Elements is computed and used to translate current Element with speed proportional to the magnitude of its velocity. This interpretation was found to provide a nice balance between the number of overlap conditions and the efficiency constraints presented in the following subsection.

Outside the Behavior description realm there is also one other way that influences the emergence significantly: some Processes move the Elements to the center, once they collide the bounding surface. This is where our implementation violates the description in order to throttle the amount of lines drawn. Instead, the spawning position for Elements are moved further away from the origin with each generation step. The rate at which this happens is an arbitrary value that yields the best visual result.

Differences

The fact that the original work produces images on a two dimensional surface signifies that the results of each generation steps are accumulated in the framebuffer according to the blending equation being used. The technique revolves around keeping the framebuffer contents from frame to frame. As a consequence of this, the drawing routines are relatively

inexpensive operations since they must render only the results of the latest generation step. Our technique, however, must maintain all the geometry produced in the past steps to allow viewing from different perspectives. This implies drawing increasingly large number of simple primitives. In order to handle the task efficiently, it is necessary to allocated large buffers on the gpu and update them with latest results. Section 4 covers this approach in a greater detail. The immediate steps that can be taken to combat efficiency issues are to throttle the amount of lines being generated at each step.

Through experiment and observation, several factors that directly influence the number of lines being produced were identified. These factor include but are not limited to the number of Elements, the size of the bounding volume, the velocity of Elements and the initial positions of the Elements. Several of these issues have been addressed by the means of assigning different fine-tuned parameters for different types of Processes. Assigning arbitrary values to these parameters can have a negative impact on the quality of the visualization. The following subsection details a way to counter some of the negative impact.

Transparency

The great part of the aesthetic quality of the original work can be attributed to how the results of each generation step are combined with contents already present in the framebuffer. In this case, the results are simply drawn on top with varying opacity levels. Since the generation takes place on a two dimensional surface, there is no notion of depth in such context. In 3D, however, in order to best approximated the transparency effect, it is necessary to take into account the order in which the Elements are drawn. It is apparent that due to the overwhelming amount of lines being drawn at any time, all the traditional methods of organizing geometry in a 3D scene become ineffective due to the large overhead of rebuilding the space partitioning tree structures. As noted previously, one of the goals is to allow observing the emergence process from any perspective in interactive frame rates.

Several alternative approaches in the domain of Order Independent Transparency (OIT) were considered. The A-Buffer approach by Carpenter [2] or any approach that maintains fragment lists is impractical due to high depth complexity of the scene we are trying to render. Similar issue affects depth peeling since it generally requires a large number of rendering passes, proportional to depth complexity [1]. McGuire and Bavoil describe an approach that require only two passes but the expectation is that the data we to be blended already exists in offscreen buffers [4]. We finally settled for an easy and straight forward approach of using a commutative blending equation, described by Sellers [7].

With simply adding the source and destination colors, we were able to achieve arguably convincing transparency effect. For effectiveness, it is necessary to choose colors with low per-channel intensity as to not cause the color clipping. We found it benefiting to express color in HSL. This way the lightness component corresponds to the opacity value in traditional blending. Also, adjusting lightness to an appropriately high value can greatly reduce the amount of lines that need to be blended together. This consequently allowed to reduce the number of Elements and increase performance. Additional design choice was to express varying gray levels (as required by most of the Process descriptions) with varying hue component.

IMPLEMENTATION

Our work was implemented using OpenGL graphics API. For efficiency purposes it heavily relies on hardware instancing an thus requires ARB_draw_instanced and ARB_instanced_arrays to be available. Any implementation and supporting version 3.3 Core of the the API satisfies this requirement. Depending on the Process, different sizes of per-instance attribute buffers are allocated. The sizes vary from 50 thousand to 5 million. Some implementations such as Intel Mesa may not support this many instances per call.

Collision detection is deferred to Bullet Physics. Depending on the type of the Form the Elements use, either sphere or cylinder collision shapes are employed.

CONCLUSIONS

In this work we have attempted to extend the original work of Casey Reas to the third dimension. Several Processes have been selected and implemented to approximate the original effect. Numerous design choices and reinterpretations had to be taken to achieve the desired effect. Our ultimate goal was to view the emergence process from any perspective and observe the growth in realtime. For comparison, our work was set aside the 2D Process visualization images. While some of our implementations exhibit greater differences to Reas work, we are fairly confident that additional fine-tuning of input parameters will yield much better results. Some differences are also due to the way the transparency is handled in our implementation. Employing a better blending model for our purposes will allow viewers to see much more detail that makes the original work aesthetically appealing.

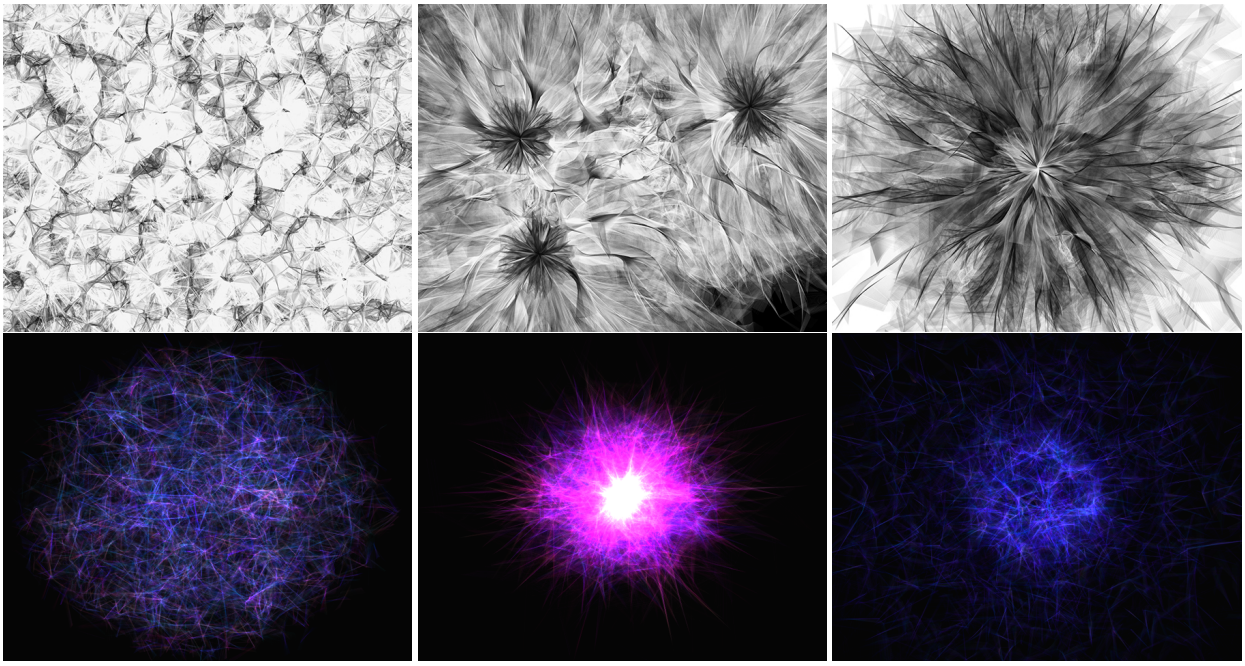
In spite of the number of issues we faced, we believe our work has a quality on its own. It opens the gates to many more possibilities. If all the issues we are currently experiencing are resolved, there exist bright perspectives to transform our implementation into a toolkit for visualizing emergence processes in 3D.

REFERENCES

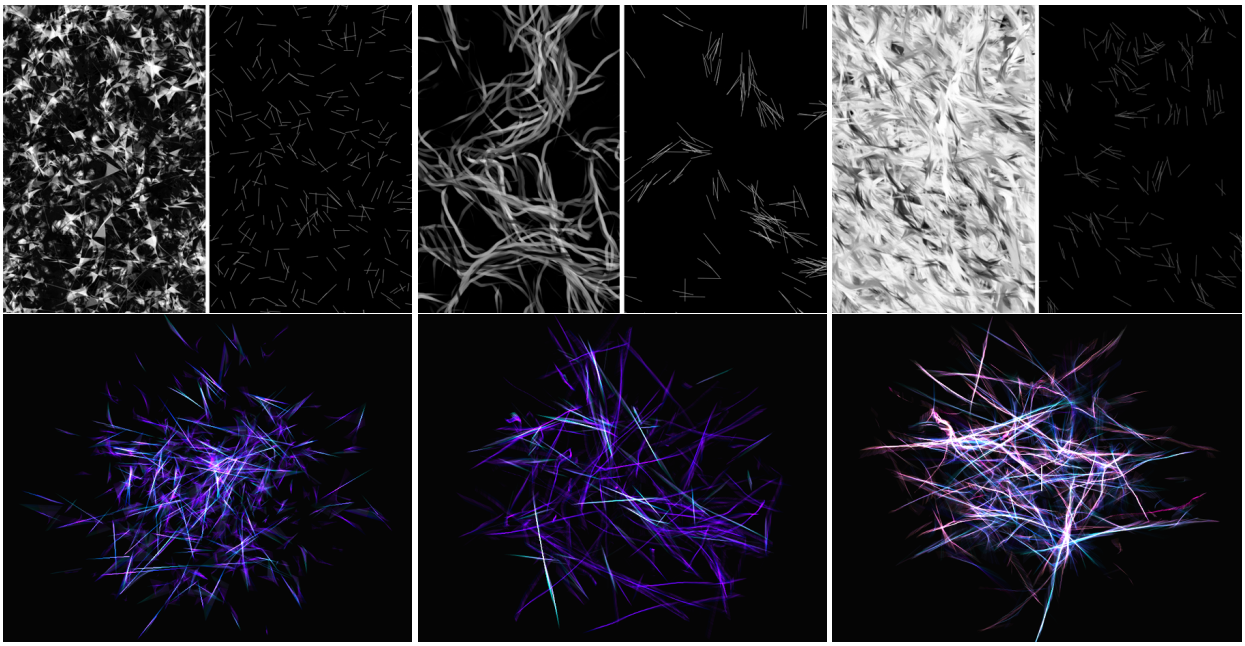
1. Louis Bavoil and Kevin Myers. 2008. *Order independent transparency with dual depth peeling*. Technical Report. Nvidia. Retrieved December 2, 2015 from http://developer.download.nvidia.com/SDK/10/opengl/src/dual_depth_peeling/doc/DualDepthPeeling.pdf.

2. Loren Carpenter. 1984. The A-buffer, an Antialiased Hidden Surface Method. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques (SIGGRAPH '84)*. ACM, 103–108.

3. Dan Collins. 2002. Breeding the Evolutionary: Interactive Emergence in Art and Education. In *4th Annual Digital Arts Symposium: Neural Network*. http://www.asu.edu/cfa/art/people/faculty/collins/emergence/emergence.htm



(a) Process 4 (b) Process 6 (c) Process 10



(d) Process 12 (e) Process 17 (f) Process 18

Figure 2: Comparison of the original (top images) and our (bottom images) software interpretations

4. Morgan McGuire and Louis Bavoil. 2013. Weighted Blended Order-Independent Transparency. *Journal of Computer Graphics Techniques (JCGT)* 2, 2 (18 December 2013), 122–141. http://jcgt.org/published/0002/02/09/
5. Case Reas. Compendium Lecture. Video. (????). Retrieved September 22, 2015 from http://reas.com/compendium_lecture/.
6. Casey Reas. 2010. *Process Compendium 2004-2010*. REAS Studio.
7. Graham Sellers. 2013. Order Independent Transparency. (20 August 2013). Retrieved Decemeber 2, 2015 from http://www.openglsuperbible.com/2013/08/20/is-order-independent-transparency-really-necessary/.