Fractal Image Editing with PhotoFrac

Tim McGraw
Computer Graphics Technology,
Purdue University,
West Lafayette, IN, USA
---
tmcgraw@purdue.edu

Esteban Garcia Bravo
Computer Graphics Technology,
Purdue University,
West Lafayette, IN, USA
---
garcia0@purdue.edu

Jo McGraw
Independent Artist,
West Lafayette, IN, USA
---
jomcgraw@gmail.com

Lisa Parker
Independent Artist,
West Lafayette, IN, USA
---
lisap1005@gmail.com

ABSTRACT

In this paper, we describe the development and use of PhotoFrac, an application that allows artists and designers to turn digital images into fractal patterns interactively. Fractal equations are a rich source of procedural texture and detail, but controlling the patterns and incorporating traditional media has been difficult. Additionally, the iterative nature of fractal calculations makes implementation of interactive techniques on mobile devices and web apps challenging. We overcome these problems by using an image coordinate based orbit trapping technique that permits a user-selected image to be embedded into the fractal. Performance challenges are addressed by exploiting the processing power of graphic processing unit (GPU) and precomputing some intermediate results for use on mobile devices. This paper presents results and qualitative analyses of the tool by four artists (the authors) who used the PhotoFrac application to create new artworks from original digital images. The final results demonstrate a fusion of traditional media with algorithmic art.

KEYWORDS

Fractals; Photography; Digital Art.

1 | INTRODUCTION

Benoit Mandelbrot (1989) described fractal art as inhabiting the blurred intersection between discovery, invention and creativity. By incorporating the compositional elements from other media into the resulting images we aim to develop an application which pushes that intersection further into the 'creativity' territory. Although fractals were not visualized until 1975, complex fractal-like patterns have been always an inspiration for artists and designers who became aware of the complex geometries they observed in natural systems, such as plants or terrain. Artists have explored the simultaneous complexity and beauty of floral patterns as decorative elements for tiles, plates or vases across all civilizations. Examples of this are Celtic Knots, the tile mosaics from the Ottoman Empire and Persian rugs. These intricate patterns convey a sense of order through their organized complexity, and often are used to represent higher spiritual concepts in a visual way. It is not by chance that "Psychedelic Art" has embraced kaleidoscopic, Mandelbrot-like imagery to represent higher states of consciousness.

Creating fractal-like structures is difficult due to the number of compositional elements that need to be aligned exactly. Obtaining a desired image may require a time-consuming search through many parameter values and spatial locations. PhotoFrac presents an alternative technique – combining fractal mathematical structures with the expressive power of existing digital imagery to enable artists to compose new works in real time.

Fractal patterns are characterized by self-similar features over many spatial scales. The mathematical techniques which have been used to generate these patterns, include escape-time iterative maps, strange attractors and iterated function systems. In the case of some iterated function systems, such as the Sierpinski gasket or Koch snowflake, the fractal structure is obvious given the details of their construction. For example, the construction of the Koch snowflake begins with an equilateral triangle. A new triangle one-third the size of the original triangle is added to the middle of each edge. This process of appending smaller triangles to each edge is repeated indefinitely. Zooming into the resulting structure reveals a familiar jagged appearance at all scales. However, for many fractal patterns it is much more difficult to get an intuitive feel for how the construction technique gives rise to the final image.

The Mandelbrot set (Mandelbrot, 1983) and Julia sets arise from a simple recursive formula applied to complex numbers. Unlike the Koch snowflake the resulting patterns are much less predictable, and can vary depending on location. The shape of the fractal is defined by the set of points in the complex plane which diverge to infinity (or "escape") after repeated iteration of the formula. In practice, however, points which exceed a fixed distance threshold during iteration are assumed to diverge. There are several ways to use these mathematical constructs in computer graphics applications. A simple approach is to apply a colormap or palette to some quantity, such as the iteration counter, to create an image. Another general approach called "orbit trapping" (Carlson, 1999) requires keeping track of how close the sequence of generated points (the "orbit") comes to some geometric shape. That distance can then be mapped to a color. Points, lines, circles and crosses have been popular shapes to use as geometric orbit traps. We use a variation on this approach called "bitmap orbit traps" which permits the user to apply colors from a selected image to the generated fractal pattern. This permits the user to incorporate other artistic works such as photographs or scanned paintings into the procedurally generated patterns. Using images as input to the creation process simplifies user interaction on mobile devices where editing code and navigating complex user interfaces is not convenient.

In the following sections, we describe related work, present the technical details of implementing a mobile app and web app, discuss the images generated for an art show, and describe prospects for future work.

2 | RELATED WORK

The full body of contemporary fractal art is too extensive to describe here, but we will highlight some notable works and artists. In the 1980s William Latham and Stephen Todd collaborated on several evolutionary art systems (Todd and Latham, 1994) based on mathematical methods like genetic algorithms, shape grammars and fractals. The resulting creations have been described as "artificial life." Conceptual artist Carlos Ginzburg considered fractals as a philosophical concept. His "Homo Fractalus" (2001) explored the ideas of chaos and self-similarity across the spatial scales of microscopic structures, individual humans, and culture. Artist and researcher Kerry Mitchell (1999), in his "fractal art manifesto", emphasizes the roles of composition, color and an expressive visual language in fractal art. He argued that the fractal artist is the most critical part of the creative process – not the algorithm or the computer. The theoretical implications of fractal geometry have been explored in applications ranging from image compression (Fisher, 1994) to modelling of natural (Oppenheimer, 1986) and synthetic structures (McGraw, 2015). Taylor (2006) found evidence that exposure to fractal artwork could lead to lower physiological signs of stress.

Image editing apps on mobile devices have become popular and powerful (Marcolina, 2011). These apps fall under the Photography category in the Google Play app store. Since there are currently more than 40,000 apps in this category it is difficult to assess all of them. Surveying the most popular and top rated apps we have found several distinct classes: image

filtering apps which perform cropping and color manipulation effects (e.g. *Instagram*, Google's *Snapseed*, *Repix*), style transfer apps which perform noninteractive computations in the cloud (*Prisma*), image collage creation (*Layout*, *Photo Grid*), and the category most closely related to PhotoFrac – kaleidoscope generators (*Mirror Lab*, *Camera Kaleidoscope*). Other related apps are fractal generating tools, such as *Frax* and *Fractoid*. These apps maintain interactivity by progressively rendering fractals from coarse to fine resolution, and some offer the ability to render large images in the cloud. We did not find any, however, which offered bitmap orbit trapping or a truly interactive full-resolution editing experience. By precomputing results we maintain the responsiveness of the application which network latency would make impossible when offloading computation to cloud-based servers.

The bitmap orbit trap technique is a general approach that can be applied to any escape-time fractal, but in this project, we use the Julia and Mandelbrot sets. It is difficult to trace the origins of the bitmap orbit traps. They seem to have originated in the computer graphics demo scene (Scheib, 2002) and have not been documented in academic publications.

The Julia set is defined by the recurrence formula $z_{n+1} \rightarrow z_n^2 + c$, where c and z are complex numbers. Since complex numbers, $z = x + iy$, have a real and imaginary part, we can consider there to be a one-to-one correspondence between points in the complex plane, z, and points in the 2D image plane, (x,y), making it trivial to transform back and forth between the two spaces. For each pixel on the screen the initial value z_0 is computed from the coordinate of the pixel (x_0, y_0). The number c that defines the Julia set is a constant. The Mandelbrot set is defined by a similar formula, $z_{n+1} \rightarrow z_n^2 + z_0$.

The sequence of points $(z_0, z_1, z_2...)$ defines the so-called 'orbit'. The Julia set and Mandelbrot set are defined as the set of points for which the sequence does not diverge to infinity. Practically, however, we do not form an infinite sequence and do not compare with infinity. Good results can be obtained by checking to see if the complex modulus $\sqrt{x_n^2 + y_n^2}$ exceeds 10 while generating a sequence of 500 points. A bitmap image trap computes the distance from each z_n and saves the point with minimum distance to some geometric shape, such as a line or circle. In bitmap orbit trapping the trapped point, $z_t = x_t + i\ y_t$, is used as an image coordinate. For the output image, I, the pixel $I(x_0, y_0)$ is set to the color of the pixel $J(x_t, y_t)$, where J is a user-selected image.

The bitmap orbit trap permits an interpretation of fractals that is new to many users. The repeating structures and patterns in the fractal define a distortion field that stretches, compresses and repeats an image. This contrasts with the more common approach of visualizing fractals by mapping from numerical values to colors in a palette.

3 | PHOTOFRAC

To reach a broad audience of users we decided to release the application as a mobile app and web app. The mobile app enables users to explore the fractal and generate images using convenient touch gestures without worrying about the security issues of downloading, installing and maintaining software on a desktop PC.

Our fractal image generation system, PhotoFrac,

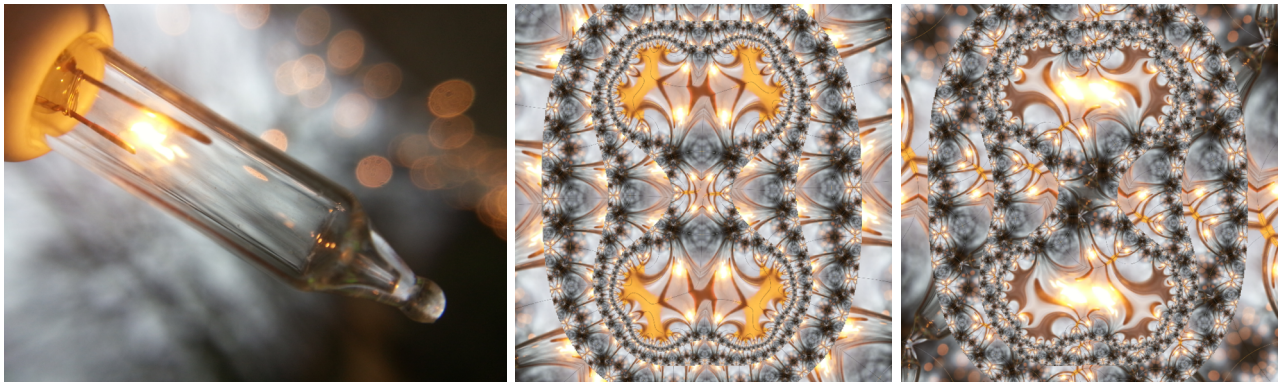

Figure 1 | Source photograph (left), photo embedded into a Julia set (center), another embedding with a different transformation matrix, M (right).

generalizes the bitmap orbit trap concept described in the previous section in several ways. We incorporated additional transformations to permit the image to be moved and resized relative to the orbit trap coordinates. Image filters such as edge enhancement and color grading were implemented to simplify the user workflow on mobile devices where processing an image using multiple apps can be inconvenient.

A transformation matrix, M, is applied when bitmap orbit trapping $[x_m, y_m] = M[x_t, y_t]$ to control the appearance of the photograph within the fractal as shown in Figure 1. Another matrix, N, controls the framing of the fractal itself. The final fractal image, I, is computed from the fractal orbit trap coordinates and the photograph, J, by computing $[x_n, y_n] = N[x_0, y_0]$ and then $I(x_n, y_n) = J(x_m, y_m)$. The image, J, is considered to infinitely repeat over the image plane, which enables negative image coordinates, and coordinates larger than the image size to be handled.

In a desktop computer system with a modern videocard the fractal generation process can be made very computationally efficient. Our initial prototype system was implemented in C++ and OpenGL. The photograph, J, was bound as an OpenGL texture map, and the orbit trap coordinates $[x_t, y_t]$ are computed in the fragment shader on the GPU. In this framework, the only data that needs to be transferred from the CPU to the GPU during image creation are the matrices M and N which are updated in response to user interaction. This permits very reactive and smooth response to user input. In the following subsections, we describe how we maintain low-latency user interaction on mobile platforms and the web.

3.1 MOBILE APP DEVELOPMENT

PhotoFrac was implemented in C#, Unity and the OpenGL shading language (GLSL) (Rost et al., 2009). The development team consisted of one programmer (the first author of this paper). Total development time was about four weeks. Using Unity permitted us to easily deploy the application to iOS and Android (Figure 2). The bulk of the image generation is handled by a shader written in the OpenGL shading language, as in the desktop prototype application. For each pixel on the screen an iterative computation of several hundred iterations is required. In preliminary experiments this resulted in poor performance (a few frames per second) on all but the highest-end phone hardware. A decision was made to precompute high resolution fractal results for a few selected fractals. The results are orbit trap image coordinates stored in a 32-bit color texture. This permits each image coordinate to be encoded as a pair of 8-bit values. Storing the image coordinates natively as floating-point coordinates was not an option since most mobile hardware does not support textures in this format. So, 16-bit floating point coordinates were computed in the shader from each pair of 8-bit values. Precomputed coordinates are transformed by the matrix, N, but the limited resolution of the precomputed texture image effectively restricts the range of scales and translations that can be applied using the matrix, M, since we precompute the coordinates only within a fixed rectangle and at a given resolution.

The transformation matrix, M, can represent rotation, translation and uniform scale. This permits the user to interactively control the mapping from precomputed trap coordinates to the image coordinates. Scale is controlled with a pinch gesture, translation by

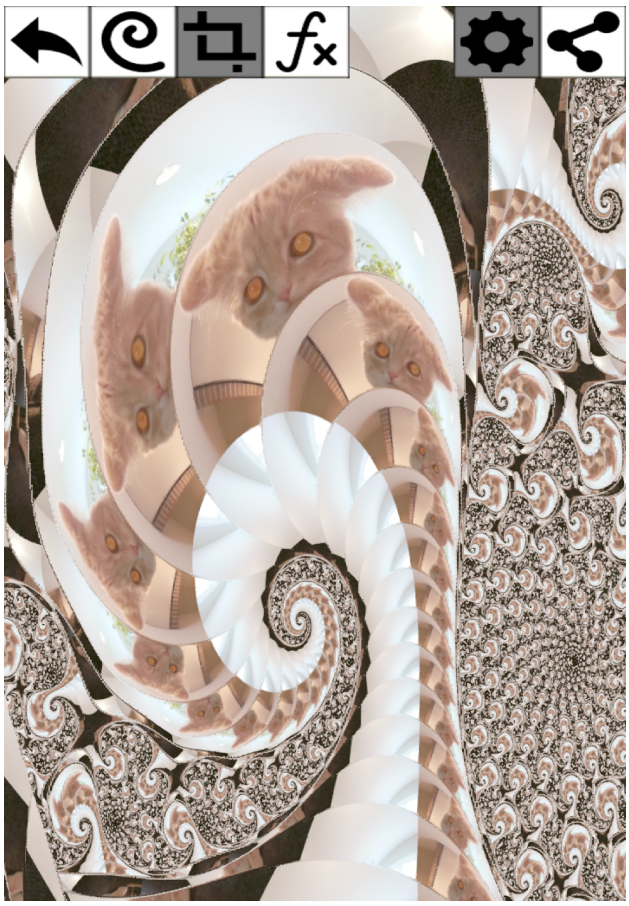

Figure 2 | Screenshot of PhotoFrac for Android.

dragging, and rotation with a two-finger rotation gesture. Scale controls the number of repetitions of the image within the fractal. When the image is scaled smaller it will repeat more times, but to reduce the appearance of sudden discontinuities across the image boundary we use a repeat and mirrored repeat modes that cause the image to be either copied or reflected across its boundaries, leading to kaleidoscopic effects. Other image manipulation features incorporated into the shader are color grading, and edge detection.

The orbit trap coordinates are precomputed at a resolution of 2048 x 2048. When the fractal is zoomed in such that a texel in the precomputed image is bigger than a pixel on the screen interpolation is required to maintain visual continuity. This results in visual artifacts in the bitmap orbit trap mapping which are not fractal in nature, but have nevertheless been used by some users to great effect.

We tested PhotoFrac on a low-end hardware device to determine the worst-case scenario for performance. The test platform was a Samsung Galaxy S3 phone, which was a three-year-old model when we began software development. Since most of the computational work is precomputed, the bottleneck to performance is the fragment shader, which runs once for each pixel on the screen. The screen resolution of the test device was 720 x 1280. After logging render times during several interactive sessions, we found that the time per frame varied between 39 ms and 26 ms (25 - 38 frames per second), which is sufficient for interactive applications. Preliminary experiments without precomputation (computing all fractal interactions at runtime) resulted in 1 – 2 frames per second and frequent crashes. The Galaxy S3 and other devices of the same generation run version 4 of the Android operating system. Statistics available to registered app developers tell us that over 35% of the Android devices on which PhotoFrac is installed are running Android 4. Combined with the positive user ratings (3.79 out of 5) this suggests that most users find the performance to be acceptable.

PhotoFrac was deployed to the Google Play and iTunes app stores in the Summer of 2015 as a free download. As of Summer 2016 the Android version of the app has been downloaded 1342 times, and the iOS version 375 times.

3.2 WEB APP DEVELOPMENT

WebGL (Marrin, 2011) is a standard for high-performance graphics on the web. It is a subset of OpenGL ES, a graphics standard for embedded and mobile devices, which is itself a subset of the OpenGL standard. As such, WebGL lacks some advanced

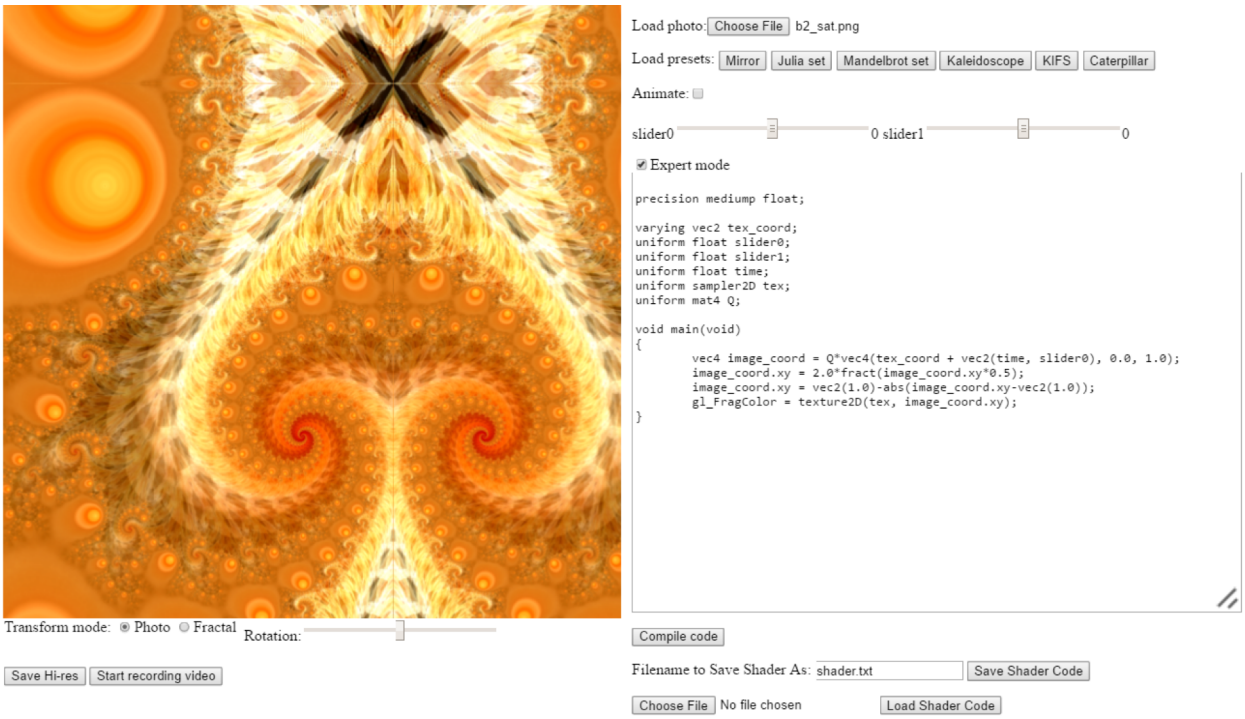

Figure 3 | Screenshot of PhotoFrac for the web.

features of OpenGL ES and OpenGL. The WebGL version of our app is intended to be run on desktop PCs with powerful videocards, so we are not constrained by computational power as we were on the mobile version. WebGL, however, can also run on several mobile web browsers, so users with powerful mobile devices can also use the web version. The supporting application framework is written in JavaScript, and the webpage it is embedded in is implemented in HTML and CSS. The web version of PhotoFrac can be found at http://www.skeezix6.com.

The web app opened a new opportunity for providing user-control of the image editing process: live-coding. Shader code in WebGL can be represented as human-readable text, and is compiled via a JavaScript function call. This permits us to provide a text window in the webpage in which the user may enter shader code, as shown in the right side of Figure 3. Modified shaders can be recompiled and used on-the-fly without reloading the webpage, permitting a much more flexible system than one which uses preselected equations and orbit traps. It is also a convenient way for the developers to iteratively refine the built-in fractal formulas.

4 | SHOW

Preliminary experiments incorporating photographic images into fractal patterns demonstrated that PhotoFrac was a versatile tool that allowed users to create unique images in their own style. Three of the authors (McGraw, McGraw and Parker) held an art show at the Athens of Indiana Arts Studio & Gallery. “Art Meets Math, Fascinating Fractals” opened on April 8th, 2016 in Crawfordsville Indiana. In the following subsections, the authors share their results and discuss their experiences in using the software to create works for the show.

4.1 USER STUDY: LISA PARKER

As an artist, my primary interests are in printmaking, but I also enjoy drawing, painting and sculpting. My subjects vary, and I move freely between representational and abstract imagery. The consistent thread throughout my work is a desire to create a visual metaphor. Prior to using PhotoFrac, I had little experience in creating digital art.

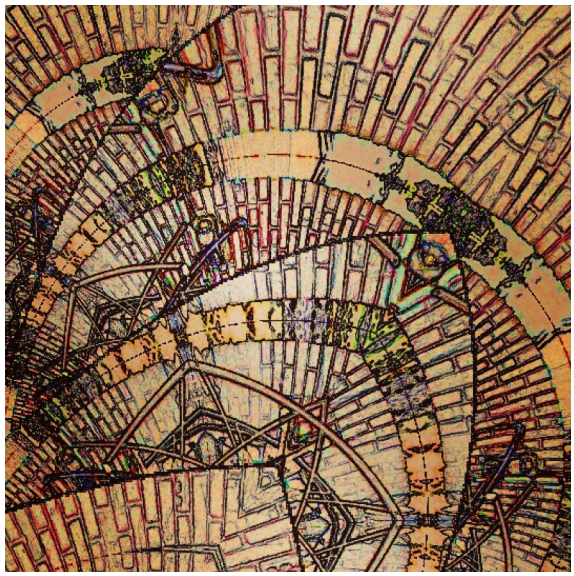

Figure 4 | “Architecture Dream”.

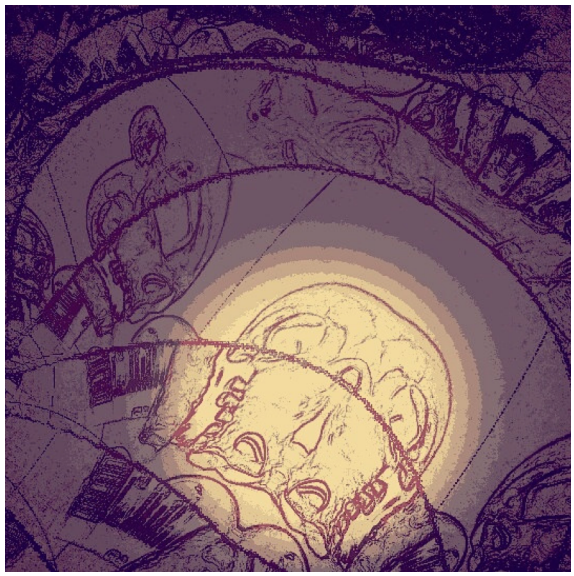

Figure 5 | “Big Brain Mask”.

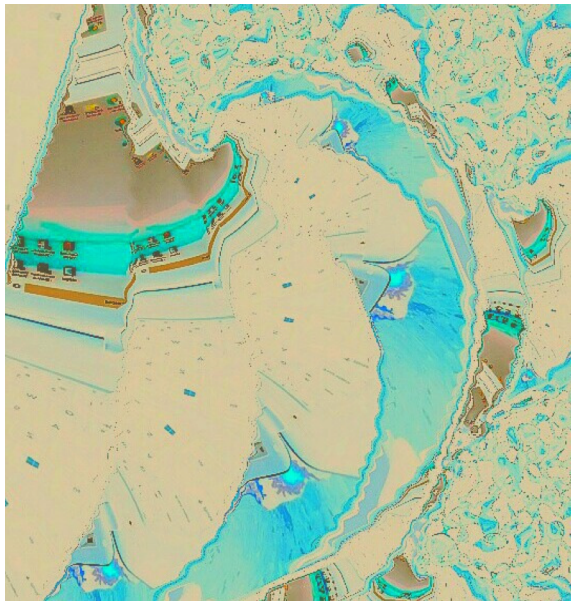

Figure 6 | “Tsunami”.

Many of my results make use of the image processing options (e.g. edge enhancement and contrast adjustment) which offer the ability to apply filters without leaving the app.

The math and technical aspects are hidden in the mobile app, so as a user I don't have to understand fractals or math to control the visual order created by applying these algorithms.

Moving within the fractal, I can choose to find balance, rhythm, texture, line, color, and shape. These elements combine and change according to the choice of fractal equation and location within the fractal.

"Architecture Dream" (Figure 4) demonstrates how PhotoFrac can amplify the elements of line and shape from my original photo. "Big Brain Mask" (Figure 5) is an example of new images being created and enhanced by combining shapes and lines as they move through the fractal. In "Tsunami" (Figure 6) rhythm and perspective are created by the diminishing size and repetition of shapes in an image.

4.2 USER STUDY: JO MCGRAW

I am primarily a painter and photographer of wildlife. As an artist living with a computer programmer, I appreciate that mathematics is a language that describes natural processes at work, even though I don't always fully understand the details. But the way fractals can stretch, duplicate, and distort an image give me a sense of the meaning of the equations.

As a user, I preferred using the mobile app. The touch interface gave me a better sense of control – with a pinch or pull of my fingers I could watch the figures in my photographs contort and change as they move through the shifting dimensions of the fractal. I tend to stay at the shallower levels of the fractal, where I can still see recognizable faces or objects, and use the transformations to explore distortions of their familiar features. I find that those changes can produce a surprisingly diverse emotional gauntlet. I often like to push an image through the fractal until a face collides with itself, or the fractal presents multiple images of the same figure, pulled and twisted in different directions.

In my work, I often edit my images in other apps before plugging them into PhotoFrac because I like to

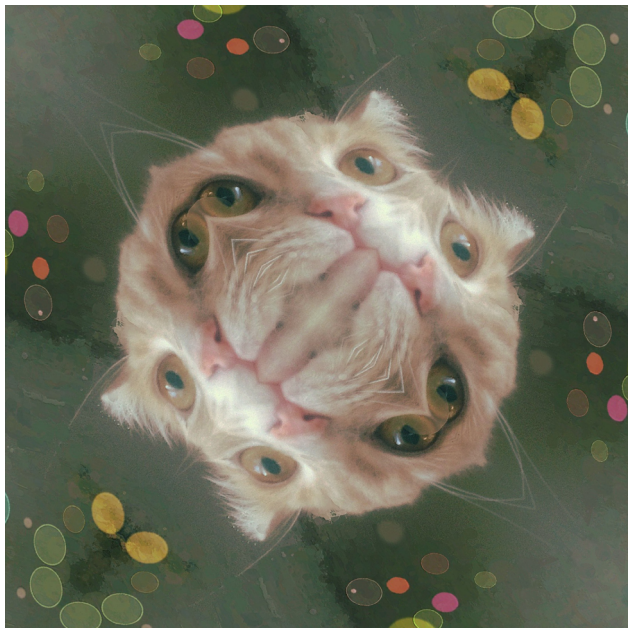

Figure 7 | "Eye ball".

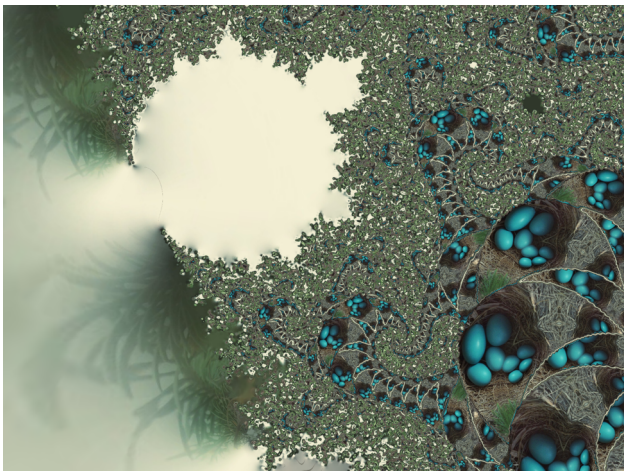

Figure 8 | "Egg Solitude".

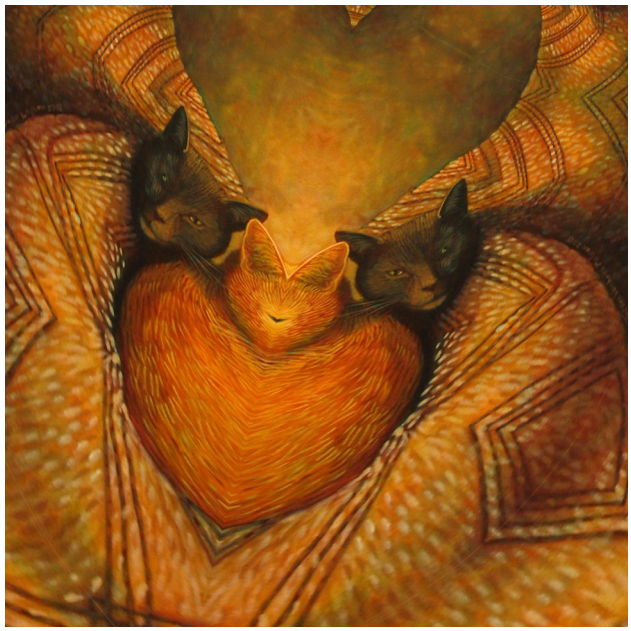

Figure 9 | "My Orange Heart".

soften and smooth the background so it blends continuously, or add texture or non-photorealistic effects to the image. Using multiple apps in my workflow gives me access to an expanded range of image filters and effects. In addition to photographs, I also use these apps to edit scans of my original paintings.

In Figure 7 four repeating images of the same cat face disappear into a single point in the fractal creating an "eye ball." To generate this image, the original photo was first edited in *Repix* (another photo editing app) to blur the background before editing it with PhotoFrac. This produces a more seamless blending of borders when the image is multiplied.

The edit of the robin's nest and eggs (Figure 8) shows how greatly magnified a small part of an image can become. In this case the pine needles in the background assume monstrous proportions, and the eggs multiplying in space look like they occupy underground tunnels.

An original acrylic painting with a variety of textures and two faces provided many potential combinations and collisions. The edit in Figure 9 generated several echoing heart shapes and minimal facial features on the cat in the middle.

4.3 USER STUDY: TIM MCGRAW

As a graphics programmer, and the sole software developer of PhotoFrac, my goal in working with the tool was to explore the extensible nature of the web version. Users with some programming experience are the target audience for the optional “expert mode” interface shown on the right-hand side of Figure 3. In this text window the GLSL shader code may be edited, and dynamically recompiled with a button click.

Recompilation of the shaders takes only a few milliseconds, and I found that editing the code became a natural part of the image processing workflow. The interface includes two general purpose sliders which are mapped to shader variables that can be used for any purpose. This is useful for exploring the parameter space of the algorithms to find optimal values. Those values can later be hard-coded in the shader. There is also a built-in variable representing

Figure 10 | “Urban Caterpillar”.

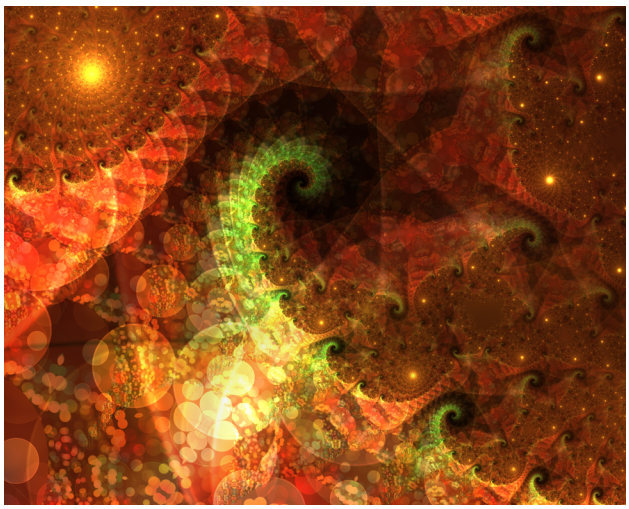

Figure 11| “Sprout”.

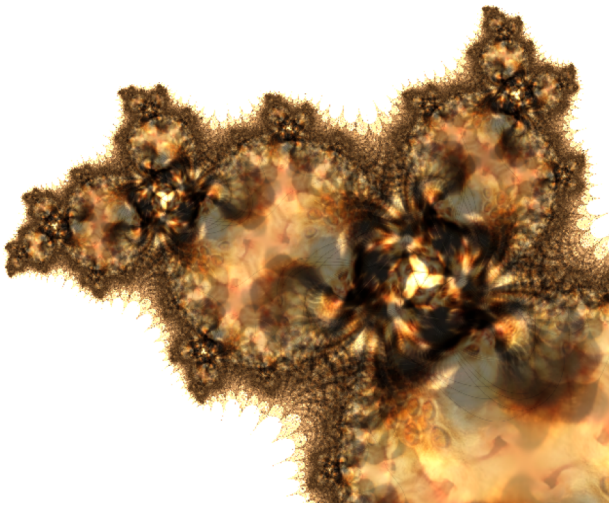

Figure 12 | “Stubble”.

time that can be used to generate short animation sequences.

A drawback of the "expert-mode" system, compared to the mobile version of PhotoFrac is that the user-generated procedures may not create usable images. In fact, errors in the code may prevent it from even compiling or running. Constraining the user to known coordinates and ranges of parameter values in the mobile app restricts the number of options available to the user, and eliminates errors due to programming language syntax and meaningless mathematical equations.

However, the flexibility and power of the web version comes at a price. When using the web version with a mouse and keyboard, one loses the tactile experience of transforming the images and fractals with touch gestures.

Many of my images are the result of experimenting with new orbit trapping schemes. Figures 10-12 demonstrate a scheme that does not compute a single image coordinate, but instead blends together image colors for each point in the orbit. This method is computationally more intensive than the prior methods, but still allows interactive frame rates.

This orbit trap allows some features of the original image to still be recognizable at low iteration counts, as in "Urban Caterpillar" (Figure 10). At higher iteration counts the original image can become obliterated and only the source colors remain.

4.4 USER STUDY: ESTEBAN GARCIA BRAVO

I am a visual artist working with digital media, including computer animation, fabrication and interactivity. When I first started manipulating images in PhotoFrac the results were reminiscent of some notable animation sequences: the distorted Donald Duck in "The Three Caballeros", and "Malice in Wonderland" by Vince Collins. Neither of these animations present logical narratives, but rather display a colorful organization of bold shapes. Watching these animations becomes an experience on its own, free from reasoning and reminiscent of psychedelic experiences. My main use of PhotoFrac was processing my own original cartoon drawings.

In Figure 13, a "Bug" character is recombined into a kaleidoscopic effect. In all my compositions, I focused

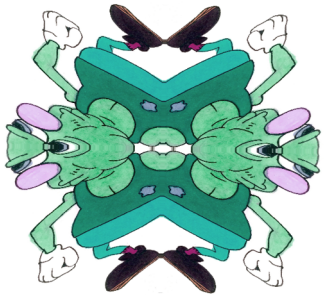

Figure 13 | "Bug".

Figure 14 | "Bird".

Figure 15 | "Sad Dog".

on the parts of the fractal that represented radial symmetry. This allowed me to be more in control of the fractal algorithm and not lose the consistency of shape and the boldness of the cartoon aesthetic.

The input image for Figure 14 was a drawing of a cartoon buzzard. During processing, I could control how the lines and colors formed new interesting shapes that did not necessarily represent the original drawing anymore, but made it look more interesting. By simplifying line and reducing the amount of color, more unexpected shapes emerged. The last test was to record the animations through the PhotoFrac website. Figure 15 shows a single frame of one of those animations.

The web version of PhotoFrac generates 15-second animations of the user manipulating the fractal. This feature is useful for animators wanting to produce abstract film shorts or create animated kaleidoscopic textures in real time.

5 | CONCLUSIONS

In this work, we have described PhotoFrac, a fractal photo editing tool. The challenges and solutions to creating an interactive user experience on mobile hardware and the web were described, and the experiences of three artists and the developer using the tool were documented.

The PhotoFrac mobile app enables the user to experiment with algorithmic art without working in terms of equations or iteration counts or other technical parameters. Colors, forms and textures are specified by selecting a source image. The transformation matrices which govern the result can be quickly specified with familiar touch-screen gestures. The more full-featured web version of PhotoFrac permits the underlying equations and orbit trap techniques to be dynamically modified, while sacrificing the portability and gesture-based interface.

Areas for future work include incorporating 3D fractals, such as the quaternion Julia sets, Mandelbulb and Mandelbox. A separate branch of the PhotoFrac project for experimenting with fractal music visualization is currently under development.

REFERENCES

Carlson, P. W. (1999). Two artistic orbit trap rendering methods for Newton M-set fractals. Computers & Graphics, 23(6), 925-931. http://dx.doi.org/10.1016/S0097-8493(99)00123-5

Fisher, Y. (1994). Fractal image compression. Fractals, 2(03),347-361. http://dx.doi.org/10.1142/S0218348X94000442

Ginzburg, C. (2001). The Neuronal Network of Social Culture, Homo Fractalus. Leonardo, 34(1), 7-7.

Mandelbrot, B. B. (1983). The fractal geometry of nature (Vol. 173). Macmillan.

Mandelbrot, B. B. (1989). Fractals and an art for the sake of science. Leonardo. Supplemental Issue, 21-24. http://dx.doi.org/10.1145/73877.73881

McGraw, T. (2015). Interactive Procedural Building Generation Using Kaleidoscopic Iterated Function Systems. In Advances in Visual Computing (pp. 102-111). Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-27857-5_10

Marcolina, D. (2011). iPhone Obsessed: Photo editing experiments with Apps. Pearson Education.

Marrin, C. (2011). WebGL specification. Khronos WebGL Working Group.

Mitchell, K. (1999). The fractal art manifesto. https://www.fractalus.com/info/manifesto.htm

Oppenheimer, P. E. (1986). Real time design and animation of fractal plants and trees. ACM SIGGRAPH Computer Graphics (Vol. 20, No. 4, pp. 55-64). ACM. http://dx.doi.org/10.1145/15922.15892

Rost, R. J., Licea-Kane, B., Ginsburg, D., Kessenich, J. M., Lichtenbelt, B., Malan, H., & Weiblen, M. (2009). OpenGL shading language. Pearson Education.

Scheib, V., Engell-Nielsen, T., Lehtinen, S., Haines, E., & Taylor, P. (2002). The demo scene. In ACM SIGGRAPH 2002 conference abstracts and applications (pp. 96-97). ACM. http://dx.doi.org/10.1145/1242073.1242125

Taylor, R. P. (2006). Reduction of physiological stress using fractal art and architecture. Leonardo, 39(3), 245-251. http://dx.doi.org/10.1162/leon.2006.39.3.245

Todd, S., & Latham, W. (1992). Evolutionary art and computers. Academic Press.

BIOGRAPHICAL INFORMATION

Tim McGraw is an Assistant Professor of Computer Graphics Technology at Purdue University. His areas of interest are procedural content generation, image processing and scientific visualization. He has been awarded 4 patents for visualization systems developed with Siemens Corporate Research. He has previous industry experience as a mechanical design engineer and as a game developer (Electronic Arts, Schell Games, Rainbow Studios). He received his Ph.D. in Computer and Information Science and Engineering from the University of Florida.

Esteban García Bravo holds a PhD in Computer Graphics from Purdue University, an MFA in Studio Arts from Purdue University and a BFA in Time Based and Electronic Media Art from Universidad de los Andes, Bogotá. His research on computer art history and digital media art practices has been featured in the annual meetings of international organizations such as SIGGRAPH, ISEA and Media Art Histories-MAH. Esteban is currently an assistant Professor in the department of Computer Graphics Technology at Purdue University, where he teaches digital imaging foundations and computational aesthetics.

Jo McGraw is an artist living in West Lafayette, Indiana. She works primarily with heavily varnished acrylic on wood. Her favorite subject is animal portraits, with a focus on each animal as a unique individual, especially cats and reptiles with emphasis on the rhythms and textures suggested by fur and scales. She also enjoys working in ceramics, digital art, and watercolor. She studied Art and English at West Virginia University.

Lisa Parker is an artist from Lafayette, Indiana. She received her Master's Degree in Printmaking from Purdue University. She works in multiple mediums, including printmaking, drawing, painting and sculpture. She has had work in many group shows, a solo show in 2014, and will have second solo show at Tippecanoe Arts Federation in August 2016. She describes her artwork as "visual metaphors about dreams, daydreams and the universe within each of us."