5-2020

An Exploration of Procedural Methods for Motion Design

David Hamilton

An Exploration of Procedural Methods for Motion Design

By

David W. Hamilton

An Undergraduate Thesis Submitted in Partial Fulfillment
of the Requirements for the
University Honors Scholars Program
Honors College
East Tennessee State University

David Hamilton	Date
Marty Fitzgerald, Thesis Mentor	Date
James Livingston, Reader	Date
Gregory Marlow, Reader	Date

Abstract

This research aims to apply procedural techniques, similar to those employed in node-based modeling or compositing software packages, to the motion design workflow in Adobe After Effects. The purpose is to increase efficiency in the motion design process between pre-production and production stages. Systems Theory is referenced as a basis for breaking down problems in terms of inputs and outputs. All of this takes place within the context of a larger motion design project.

Introduction

The motion design process has been compared to the process of building a custom home. If an orderly process is not followed, the result will be haphazard. BIEN, a motion graphics studio in Los Angeles, illustrates this point with the following scene:

> A young couple walks into a general contractor's office to discuss their custom home project and check out some in-progress photos. After reviewing a few images, they say:
>
> "Hey, we love that bathroom, but can we put it on the right side of the hall instead? And I know we said we liked the granite countertops, but after seeing a few samples this weekend, we now prefer concrete." (BIEN, 2019)

This is a huge structural change; one that will require rewiring, repiping, and much more. It will cost the homeowners a significant sum, which could have been avoided during the earlier planning stages of the project. When it comes to efficient processes, the order of operations matters immensely.

Background

For motion design, the process begins with just an idea. As with any creative process, every step along the way is another building block, getting closer to envisioning the final product before production begins. A treatment is developed, including moodboards with reference ideas, along with thumbnail storyboards that go along with a voiceover script, if applicable (BIEN, 2019). The next step is typically to develop styleframes to visually communicate moments of what the final output should look like. Finally, it's time to begin production, and implement what has been planned. Animatics are often made as a rough, timed previsualization of the motion in a scene.

From there, assets are built, designs finalized, and animation takes place, resulting in the first full draft of the piece.

Animation workflows have come a long way over the decades, from traditional "Disney" 2D methods, to the optimized 3D pipelines today. Advances are made when creatives make more effective use of tools. In the end, it's always the final output that matters; the more parts of the process that can be automated or optimized to avoid repetitive work, the more quickly and creatively fine-tuned the animation can be completed. This way of thinking has been crucial in developing the 3D pipeline for character animation, for example. In the past, a character would need to be drawn by human hands for each and every frame of animation. This process is tedious and repetitive, and would be even more time-consuming in 3D. Imagine if a character had to be modeled from scratch every time the animator needed to put them in a new pose. Instead, it is much more efficient to have characters modeled and rigged before being passed off to animators. With this additional work upfront, animators can begin with an asset that allows them to focus on the animation--the poses and the motion that gives the character life.

A similar procedural line of thinking can be applied to the 2D motion graphics pipeline. The term, "procedural," in computer graphics, is typically used in a 3D context. For example, procedural modeling refers to the construction of models, textures, etc. purely from sets of rules (Ganster, 2011). At some level, everything is boiled down into algorithms, and procedural-friendly software retains the sequence of inputs and outputs that are generating the final result. The user can then change any attribute or transformation along the chain, and the rest of the "procedure" will be followed to completion by the software. This allows for more creative iterations with less repetitive work on the part of the user.

Research Objective

For the purposes of this research, we will be examining a small segment of the motion design process and exploring the use of procedural methods in order to increase efficiency. Our exploration will take place within the parameters of a larger motion design project.

This particular project is an informative piece briefly explaining how human color vision works. The project has already undergone its treatment stage; that is, we have a voiceover script accompanied by rough thumbnail storyboards. We also have mood boards as reference for style. Our next step would be to make styleframes; our goal is to take these pre-production plans, and move towards production in the most efficient way--by incorporating procedural thinking. Finally, we will take a look at specific challenging moments in our storyboards, and begin problem solving for those scenes.

The piece begins by making the surprising statement that an image of a banana on the viewer's screen displays absolutely no yellow light. This premise is then broken down and examined, presenting the viewer with a visual/verbal summary of color vision. RGB screens are introduced, and it is demonstrated how they exploit color vision mechanisms.

Methods

There is a moment towards the beginning of our storyboards that calls for a visual of a banana ripening. We would like to see a stylized animation that depicts brown spots appearing and spreading across the banana. There are many different ways we could technically end up with a similar output. This could be tackled using relatively traditional animation methods. An animator could study reference material, manually draw spots popping in frame by frame, and irregularly grow them by eye. However, if the animator does not get it quite right the first time, they would

likely have to restart the entire tedious process. Each iteration is as costly as the first, both in time and in labor.

An improvement on this method would be to use keyframable shape layers to create the brown spots. Once the general pattern of the spots has been created, each item can be keyframed and tweaked to pop in at different moments and at different growth rates. The animator can watch their work, and make tweaks without manually redrawing each frame. While this is a marked improvement, we can still do better.

When problem-solving procedurally, it is helpful to borrow from a philosophy known as Systems Theory. Systems theory is a study of the organization of phenomena, and can be applied to different scenarios in nature, or in business and economics (Sauter, 2000). The theory offers a method of analysis centered around inputs and outputs, which is applicable to a procedural workflow.

Banana Ripening Sim

Using this framework, we can break a problem down and define our system by asking questions about the inputs and outputs involved. To approach this chronologically, we must ask: First, what inputs do we need for our system to function? In our previous proposed methods for the banana animation, by contrast, the first step that had to be decided was how our spots would be distributed. It is not necessary for this input to be executed entirely and directly by one person--or by a person at all. Instead, we can utilize something automated to decide where a spot will appear and where it will not--some process to determine the arbitrary locations where spots should pop up. A noise generator is perfect for this problem. In this case, we have used turbulent noise applied to a solid layer.

Now that we have a starting input, it is a matter of determining what transformations to apply in order to achieve the desired output. To go from the grayscale noise to discreet spots, we can use a curves or levels effect to “crush” values above a certain threshold to white, and values below a certain threshold to black. As soon as we complete one step, we are faced with the next question: How can we get those spots to appear and grow organically? One way might be to keyframe the brightness back in the noise generator. Then, we will turn all this into brown spots on a yellow background. These can be solids which could be replaced by textures, either generated with effects or driven by other inputs that we can import. Putting our animation in context with color and texture helps to better evaluate our design, and make informed improvements.

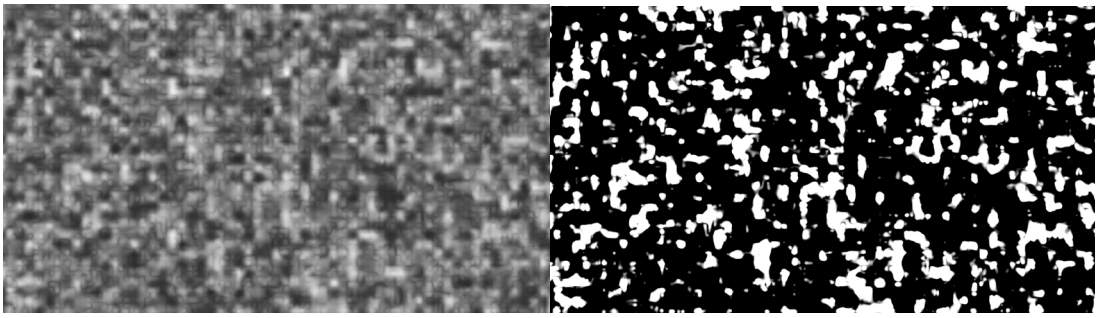

FIG 1: Raw generated noise on the left, values crushed on the right.

Comparing this to reference material, we find that our spots are too uniform. They all come in at around the same size at around the same time. Instead, they need to emerge in clusters which later expand and overlap. This can be achieved by inserting another noise effect, but this time at a larger scale. Again, we can crush the values and use this as a mask, multiplied with our existing spots. We can repeat this process in conjunction with a combination of other effects such as glows, blurs, masks, and roughen edges. Each iteration can be tested, viewed, and evaluated in near-final form. A lot of guesswork is eliminated, allowing the designer to experiment without fear of wasting

work.

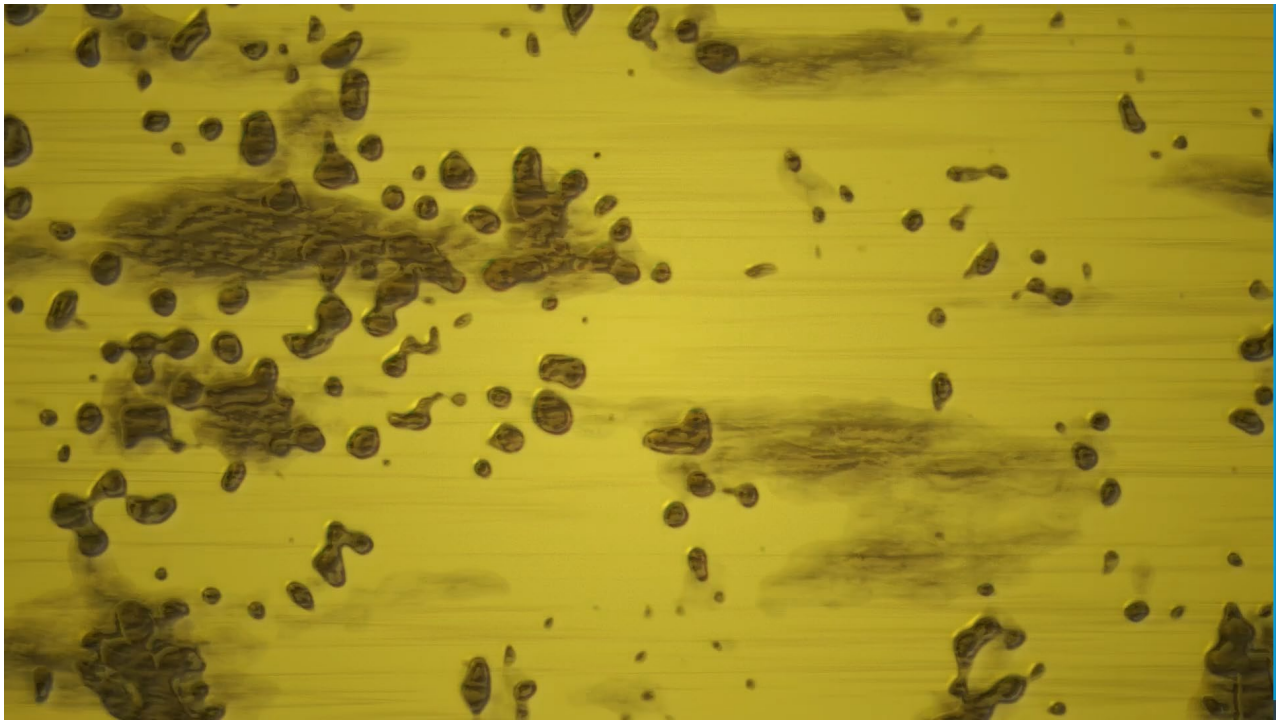

FIG 2: A final frame with texture and color applied

Eyeball Rig

There is a scene in our project which calls for an eyeball to rotate into frame, and dilate as the camera prepares to zoom in through the pupil. Again, there are a number of ways to approach such a shot, but in the interest of efficiency, we would like to build a "rig" procedurally that can be manipulated and keyframed by an animator.

When defining our system in the banana scene, we began with the necessary inputs, and worked forwards. Another way to approach the definition of a system is to work backwards, starting from the output. After all, what matters in the end is the final result. Questions asked as part of this method would include:

What essential outputs must the system produce to satisfy our needs?

What transformations are necessary to produce these outputs?

What inputs are necessary for these transformations to produce the desired outputs?

(Sauter, 2000)

First, defining our necessary output, we want an eyeball rig with controls for dilation and rotation. In a sense, to the end user, we have just defined our inputs as well: the controls. To get that result, we'll need a set of transformations that wrap a texture around a sphere, and for that texture to be altered based on a custom "dilation" controller input. Each part of the problem can be further defined as a smaller system of inputs and outputs.

FIG 3: Source noise for iris texture. Black solid at the top moves vertically in response to dilation slider in final comp.

In this case, the textures for both the white of the eye and the iris have been generated procedurally with a series of effects beginning with fractal noise. The "dilation" slider in our final comp has been mapped to two separate expressions back in the texture comps. First, the position of the black solid representing the pupil moves vertically in proportion to the user input on the slider. This translates in our final comp to the pupil growing larger or smaller. To add to the effect, the dilation slider also drives part of the fractal noise at the source of the iris texture. By offsetting one

of the subscale elements in relation to the slider value, we get the illusion of muscle contraction driving the pupil dilation. One this “rig” is set up and fine-tuned, the animator only needs to touch a few controls in order to keyframe the eye effectively.

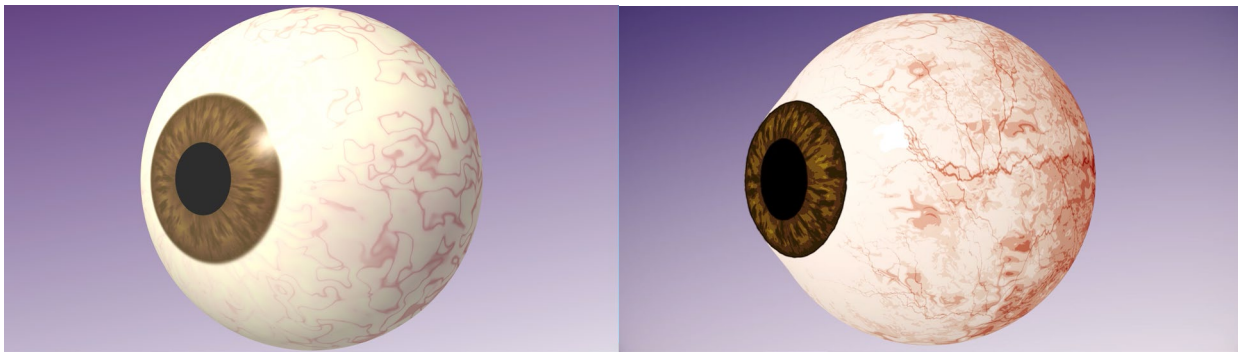

FIG 4: Even after a rig is created and keyframes applied, significant stylistic changes are possible without losing work

RGB Screen Filter

Another scene we will need involves an exaggerated close up of an RGB screen. In this case, the desired result behaves more like a filter than a rig, requiring a single user input: another composition, either video, still, or animated. The input to output chain is especially straightforward. First, the composition is pixelated, using the mosaic effect. Then, the red channel is isolated, and a procedurally generated mask lets through a set portion of each pixel. The same is done for the green and blue channels, offsetting the masks a bit. Using expressions, we can set up a control for how pixelated / close up the final output should be.

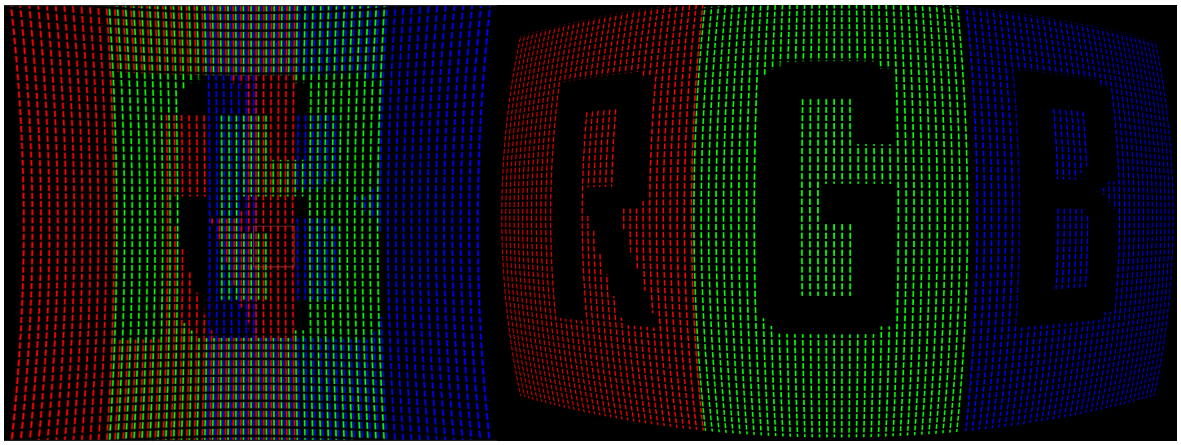

FIG 5: Sample outputs from the RGB Screen Filter.

Cone Signal Visualization

There is a recurring need in our storyboard for a visualization of the different signals that the cones might send to the brain when experiencing different colors of light. It was decided that a radar graph would represent the information well, displaying a different shape for every color. If this were to appear only once or twice in the piece, and it was known which colors exactly were being used, it might make sense to manually keyframe the shape path of the graph. However, we would like this visual to have a continual presence on the screen for a particular sequence. Thus, it will be helpful to rig it up so that the animator only needs to manipulate one color input, and the graph will follow along.

We use a set of expressions to separate each of the RGB values from the color input. Another expression is used to approximate the response level of each cone. These are then mapped to points on a corner pin effect, using trigonometry to place them appropriately along each of the three axes. The result is a highly responsive data visualization that can be easily updated with simple keyframes on a color control.

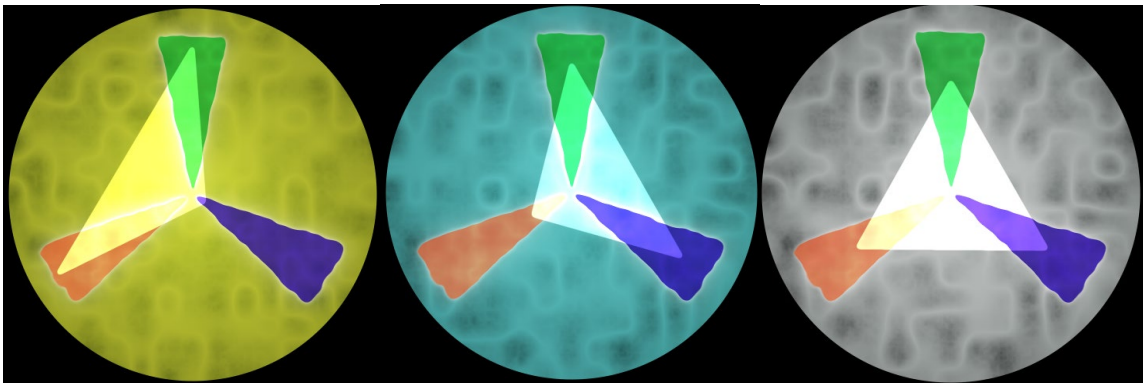

FIG 6: Cone signal visualizer responding to yellow, cyan, and neutral gray inputs respectively.

Benefits

Using procedural methods, we've constructed assets acting much like rigs or models that can be passed on to animators. The techniques used allow for tweaks in style to be made without losing valuable rigging and even animation information that is still applicable. These tools offer a higher level of flexibility for the animators to experiment and fine-tune, without sinking a lot of time with repetitive tasks on each iteration.

Normally when dealing with a creative process, it matters immensely that tasks are finalized and executed in a set order. However, when working procedurally, one can effectively design that process, setting up a system without physically executing each step personally. The brute force is being accomplished by the computer, which, unlike people, does not mind repetitive processes one bit. Working procedurally, one can lay out a process, then alter a piece of that process early in the chain, thereby affecting everything after that point, but without any manual redoing of work.

Future Applications

There are a number of practical applications for this kind of method. Setting up tools and assets procedurally allows work to be re-used and adapted for other projects, or even sold as templates or plugins. Alternatively, the techniques we have explored could also be applied in use cases in which the end user is not involved in animation at all. For example, a procedural setup with a single input could be mapped to respond to user input in an app or webpage. Using the same line of thinking required for procedural workflow, it would be theoretically possible to swap out any one piece of the input-output chain normally performed by a designer, with a specially trained neural network. This would free up more time, enabling even more iteration and experimentation by the designer. The possibilities are far-reaching, and this research has only just scratched the surface.

References

Abrams, Evan. [ECAbrams]. (2019, May 17). *From Storyboard to Motion* [Video]. YouTube. https://www.youtube.com/watch?v=84rA66PobII

BIEN. (2019, March 28). Everything You Need to Know About The Motion Design Production Process: BIEN: Motion Design Studio. Retrieved from http://thisisbien.com/motion-design-production-process/

Chen, S. (2017, June 1). Procedural Thinking: Understanding AI and ML in the next workforce revolution. Retrieved from https://medium.com/eliza-effect/procedural-thinking-how-procedurally-literate-are-you-241b50728eaa

Ganster, B. (2011). Procedural Modeling. Retrieved from https://cg.cs.uni-bonn.de/en/projects/procedural-modeling/

Sauter, V. L. (2000). Systems Theory. Retrieved from http://www.umsl.edu/~sauterv/analysis/intro/system.htm