Computational Aesthetics in Graphics, Visualization, and Imaging
Paul Rosin (Editor)

EXPRESSIVE 2015

# Rendering Artistic Light Patterns

L. Ji [†1], A. Gooch[3], L. Gammon[2] and B. Wyvill[1]

[1]Department of Computer Science,
[2]Department of Visual Arts,
University of Victoria
[3]Scientific Computing and Imaging, University of Utah

**Abstract**

*By combining knowledge from computer graphics and visual arts, we have built a projection installation based on a novel sketch-based shape pattern rendering method. Our novel rendering method is guided by an artist's drawing, and generates shape patterns resembling the input image, creating animation with an organic appearance. We have also applied the proposed method to render foliage shadow effects for virtual scenes. The major contribution of method is its ability to automatically render richly detailed, animated lighting patterns from an approximate lighting plan drawing. In our research team, professional artists and computer graphics researchers work together to develop our rendering method and the artistic rendering projects side by side. This interdisciplinary approach helps us to design and evolve our method for creating aesthetic work with computer graphics technologies.*

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture; Computer Applications [J.5]: Arts and Humanities—Fine Arts.

## 1. Introduction

In northern latitudes on the west coast, winter brings long nights and much rain. In response to this dreary environment, we have created a light art projection installation. One wintery night, our installation created bright and colorful light in a public space, and attracted viewers to stop by (Figure 1). In our research team, professional artists and computer graphics researchers work together to develop a novel rendering method that successfully supported this visual arts project. From the computer graphics perspective, we seek to devise a method that can automatically follow an artist's lighting design, and renders the depicted scene with detailed lighting conforming to the artist's composition.

The rendering method we propose for this goal contains two components: a stochastic optimization algorithm for creating shape patterns resembling an input image from given shape examples, and a simulation algorithm for generating organic-looking animation. The application of our method is not limited to the installation project; we also show its effectiveness through a digital rendering project that creates foliage shadow effects. Using a few paint strokes, an artist can easily design lighting for a virtual scene assumed to be placed under densely foliaged trees, without modeling the complicated tree structures. Our program then generates plausible light and shadow patterns from the input drawing. The program also applies a blur effect and produces animation to improve the visual appearance of the rendered scene, creating an impression of tree foliage shadows swaying in a gentle breeze.

We propose that rendering light guided by an artist's drawing is an intuitive way of rendering artistic lighting effects. Based on this idea, we propose a sketch-based rendering method, and examine this method in two different contexts of artistic creation. The contributions of our research consist of:

1. A novel sketch-based shape pattern rendering method that automatically generates shape patterns following an input image, and in addition creates coherent animation (Section 4);
2. A case study of a video projection installation in a public space supported by the proposed method. We exam-

† ilucky@cs.uvic.ca



DOI: 10.2312/exp.20151174

(a) The surrounding environment of our installation.

(b) The installation show in a misty winter night.

(c) The installation show attracted viewers to stop by.

Figure 1: Our "Light Movement" installation in University of Victoria with two data projectors and one photographic light (Figure 1(c)).

ine our implementation of the installation in detail, and identify how our sketch-based rendering method facilitates the art project (Section 3);

3. A method for creating foliage shadow effects in digital rendering. We show that our proposed method enables a straightforward and intuitive workflow for creating plausible, animated foliage shadow effects (Section 5).

The rendering method, the projection installation and the foliage shadows rendering project were developed side by side by our interdisciplinary research team. In this paper, we present the installation project as an introduction to our rendering method, then discuss the algorithmic design of the method. Next, we show how to use the proposed method in digital rendering for foliage shadow effects. We suggest future research and conclude our work in the last sections.

## 2. Related Work

Computer graphics researchers have envisioned the potential for freely composing lighting for a long time, with pioneer work dating back to Schoeneman et al. [SDS*93] and Poulin et al. [PRJ97]. Recently, researchers have tried to redefine surface lighting in a more user-friendly fashion, and have proposed systems to automatically compute parameters such as light source, material or surface mapping according to user input [PBMF07,RTD*10]. Kerr and Pellacini [KP09] noted that users are only capable of drawing coarse approximation of desired lighting, instead of directly composing every lighting detail. Therefore, sketch-based interfaces are preferred in lighting design, as demonstrated in the 'envyLight' [Pel10] and 'Illumination Brush' [OMSI07] systems. Following the same strategy, we only require an artist to input a sketchy lighting plan drawing; our program then automatically produces plausible lighting patterns and creates animation. In comparison to their systems that create physical based lighting effects with environment maps, our rendering method is motivated from a visual arts perspective and not limited to photorealistic lighting effects.

The development of powerful lights, projectors and computers have given artists the ability to manipulate light patterns in large scales. In 2009, artist Michel de Broin and a group engineers worked together to install a sphere of 1,000 mirrors 50 meters above the ground of Paris, and used large light projectors to render an artificial starry night [dB09]. Artist Charles Sandison created his own computer program to render a flood of bright text patterns from multiple data projectors inside the Central Exhibition Hall 'Manege' in Moscow [San13]. Recently, an interactive installation project was installed in Abu Dhabi by artist Rafael Lozano-Hemmer, who used strong search light beams to visualize the heartbeats of its viewers in the night sky [LH15]. In our research, we also present a projection installation in public space, in which the animated light patterns are generated from our rendering program.

Computer graphics research of shape pattern arrangement usually distribute elemental shapes according to a given texture [HLT*09, MWT11]. Iterative and optimization algorithms are commonly used in adjusting shape patterns and making them resemble an example [AKA13,LBW*14]. Although we do not consider texture as the primary goal in our research, we also approach the pattern arrangement problem with a stochastic optimization algorithm. Given a segmented input image, classification methods can be used to fit a collection of simple shapes onto the image, and render the input

image into an abstract, paper cut-out style [SRHC08]. We set a similar goal in our research of using a set of elemental shapes to represent the input drawing. In addition, we create animation for the shape patterns instead of rendering static abstract images.

To create a coherent, organic looking animation for the shape pattern, we apply an harmonic motion simulation with random excitations. As suggested by Chuang et al. [CGZ*05] and Habel et al. [HKW09], a human observer can only judge a highly complicated dynamics system by its overall movement frequency and amplitude. Based on this observation, we only require artists to specify frequency and amplitude features of the generated animation, and use random values for other animation parameters. We choose the excitation (driven force) for the harmonic motion to be a random signal that conforms to the $\frac{1}{f^{\beta}}$ statistical model, discussed in Peitgen et al. [PS88], Chapter 1.2.3. This is a statistical model commonly observed in our daily life, from traffic patterns to ripples on lakes.

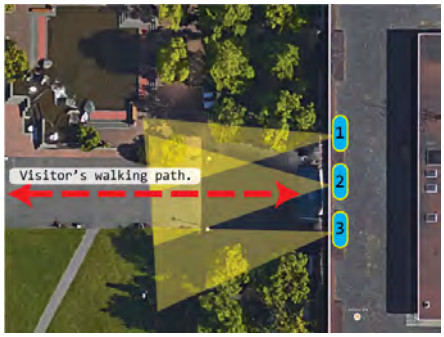
We also demonstrate rendering foliage shadow effects for virtual scenes using our method. In our approach, the foliage shadows are rendered as shape patterns, not cast from tree structures. Photorealistic methods of modeling and animating trees and their shadows are usually inspired by biological rules [PHL*09, PSK*12] or mathematical models [PLH*90, LPC*11]. Recently, researchers also demonstrated creating plant structures based on photo or video input [BNB13, LDS*11]. Since we focus on rendering the shadows from foliage rather than the photorealistic trees, our method only requires models of tree leaves instead of complete tree structures.

## 3. Light Art Projection Installation

We present our light art projection installation as an introductory example of the proposed rendering method. As an installation in public space, we have to adapt our design to available spaces and efficiently implement the installation in a short time. In this process, the method we developed for rendering light patterns from an input sketch plays an important role.

The environmental visual context motivates the design of our installation. As shown in Figure 1(a), on a typical winter day on our campus, the grassland exhibits a half yellow, half green tone due to the lack of sunshine. Soaked in rainwater, brown clusters of soil and plant stems scatter on the ground. At a distance in the background, the monochromatic sky falls behind the black forest and the concrete buildings. This dark and humid environment creates stress in many people [Ros12]. In contrast to this environment, we propose the creation of bright and colorful light patterns using data projectors in a public space. In the long and depressing winter nights, we hope our projection installation will deliver a visual presentation of crystal-like light shapes and a sense of cheerfulness.

(a) Initial planned installation site.




(b) Initial implementation plan.

Figure 2: Our initial planned site for the installation has a largely different visual context from the site finally approved. The implementation plan and the light pattern design must be revised. Satellite map from Google Maps [Goo15].

In our initial proposal, we planned to mount three data projectors on top of a building's roof, and render the space around a fountain of our campus (Figure 2). Although we gained support for the proposal of cheerful light patterns, the originally planned site was not available. We made several alternative proposals, and finally gained permission to set up the installation in another space that had a sculpture in its center [Sni76]. The projectors had to be mounted onto high ladders, instead of the rooftop; and our installation need to be shown on a specific day one week after the permission was granted. We discovered that administrative negotiation usually takes a major portion of effort in implementing art projects in public spaces. A well known example in contemporary arts is the "Running Fence" installation in 1972, which took the artist more than four years to acquire permission from the city administration [Chr76].

In our case, the overall design of rendering colorful light patterns with data projectors was unchanged from the beginning; but we had a very tight schedule to adapt this design for a new site. In less than a week, we needed to make new implementation plans according to the approved space (Figure 3(a)), and produce videos accordingly for both projectors at the new mounting position. In this situation, our rendering


(a) Implementation plan for the projection installation, with the lighting plan of the second projector.

(b) Lighting plan drawing of the second projector.

(c) A frame from the video rendered using our method. The model of the elemental shape is shown on top right.

Figure 3: The final implementation plan of the "Light Movement" installation. The lighting plan drawing (b) and the elemental shape (c, upper-right) are designed based the visual context of the installation site. These inputs are taken by our rendering program to produce the video for the installation. The enlarged lighting plan is flipped because the video is rendered from the viewpoint of the projectors. Satellite map from Google Maps.

program enabled us to complete the installation on time: we drew a new lighting plan image (Figure 3(b)), modeled the sculpture standing in the center of the space (Figure 3(c), upper-right), and rendered animation videos in a few minutes. In comparison, manually fitting lots of small visual elements into a desired pattern usually takes more time and effort. In order to produce a coherent light pattern video by hand, an artist has to place numerous shapes in video editing software, and laboriously adjust the trajectory of each shape throughout the animation. We developed our rendering program before the busy week of setting up the installation. The program saved much time for us in making the projection videos, and enabled us to fine-tuning the installation setup, such as renting suitable tall ladders and dimming nearby road lamps to enhance projection contrast.

Our rendering method is capable of transferring an approximate lighting plan drawing into a detailed, animated video. To manually make a video that conforms to a lighting plan, an artist can use any material, from real-life photographic images and videos to abstract shapes. Currently, a computer program cannot match the creativity of a human artist, and we simplify our rendering task by restricting the content of the rendered video to be animated *shape patterns*. The shape patterns are generated by duplicating several *elemental shapes* over the video frame. In the presented installation, we chose the elemental shape to be a three-dimensional model of the sculpture in the center of the space. The elemental shape is shown in the upper right corner of Figure 3(c); and a rendered frame is shown in the rest of this figure. The elemental shape was duplicated many times and rendered in various colors and angles to match the lighting plane design in Figure 3(b). With this elemental shape, we effectively created light patterns that closely relate to the visual context of the installation site. Should the installation be shown in another space, the elemental shape for the light patterns would also need to be redesigned. We also assume the input lighting plan drawing to be a grey-level image, and only require the rendered video to resemble the input drawing in its overall appearance. We discuss the algorithmic design of our rendering method in the next section.

One night, the bright green and blue colors from the data projectors lit up the grassland, created a translucent, mystical and cheerful atmosphere. Animated by our method, those sculpture-shaped light patterns swayed around like tree leaves in a gentle breeze. The upper part of the sculpture was lit up by a photographic light, to introduce a warm flood of light into this winter night. With a dark lower part, the sculpture seemed to be rising from the surrounding light shapes. About half an hour into the installation show, the mist began to accumulate in the night air, and added glowing halos to the sculpture and its surroundings (Figure 1).

Many viewers were attracted to this place on our campus, curiously watched how these moving lights rendered the familiar surrounding into an exotic, delightful space. In conversation with the viewers, they remarked to us that the colorful lights really stood out in the dark night. An artist walking by commented that the movement of the light shapes appeared like living creatures, while the individual light shapes looked quite abstract. The installation was disassembled later in the same night.

## 4. Methodology

The goal of our rendering method is to automatically create shape patterns that match an input image, and to generate coherent animation for these shape patterns. We propose a two-part solution for this problem. First, we use stochas-

(a) Example guide image **I**.

(b) Candidate shape pattern $\mathbf{I}'$ with triangle, square and circle.

(c) Difference image **R** and the evaluation cells
(enlarged center part of the full image).

Figure 4: Optimization Heuristics. Blue in the difference image (c) denotes pixels covered only by the guide image but not the candidate shape pattern, while red stands for pixels covered only by the candidate shape pattern but not the guide image. The dark brown squares represent the evaluation lattice. The yellow squares represent instance cells, which have their centers aligned with the geometric centers of the shape instances.

tic optimization to generate shape patterns that resemble the input drawing. Secondly, we create animation of the shape patterns by simulating harmonic motion, with a statistical model for the excitation.

### 4.1. Creating Shape Patterns from an Artist's Sketch

Using a few bold paint-strokes to draw a lighting plan is an intuitive way for an artist to express the lighting composition (Figure 3(b)). Our method automatically transfers such a drawing into detailed shape patterns by duplicating a few elemental shapes with various colors and geometric transforms. The input lighting plan drawing is called the *guide image*. The basic strategy of our method is stochastic optimization: we generate a random shape pattern as the starting candidate pattern, adjust the candidate pattern and evaluate whether it better resembles the guide image, and repeat this process until a satisfying shape pattern is reached.

An artist can choose any set of three dimensional models as the elementary shapes, and specify a range of colors for them. In Figure 3, the elemental shape is a model of the sculpture; and in Figure 4, the elemental shapes are triangle, square and circle. In Section 5, we use models of tree leaves as the elemental shapes to render foliage shadow effects. In our implementation, we use software instancing to duplicate elemental shapes over the image plane, and render a shape pattern with orthogonal projection. These shape instances can have different scale and rotation transforms, but their geometry centers are constrained in a plane, and can only have planar offsets. The candidate shape patterns are rendered with vertex color set to white against a black background (Figure 4(b)). Colors for the shape patterns will be applied after the optimization. We compare the rendering of the candidate shape pattern against the guide image, and set the goal function of the optimization $z$ to be the accumulated pixel value difference between them:

$$z(\mathbf{I}') = \left(\sum |V(\mathbf{I}_{x,y}) - V(\mathbf{I}'_{x,y})|\right)/N.$$

In the above equation, **I** stands for the guide image, and $\mathbf{I}'$ represents a candidate shape pattern image in the optimization. $V$ denotes the value of a pixel. $N$ represents the total number of pixels in the rendered image, and the goal function is normalized in $[0,\ 1]$. In this section, we assume that the guide image has exactly the same pixel resolution as the rendering resolution of the shape patterns. If **I** and $\mathbf{I}'$ have different resolutions, our program will re-sample the guide image to the resolution of the shape pattern image by trilinear interpolation.

In each iteration of the optimization, we start with a candidate shape pattern as the *reference solution*. From the reference solution, we generate multiple new candidates by adding or deleting shape instances and adjusting the transforms of existing shape instances. We then evaluate the goal function $z$ on the newly created candidate, and compare it with the $z$ value of the reference solution. If any new candidate results in a better (smaller) $z$ value, we pick the best candidate shape pattern in the current iteration and set it as the reference solution for the next iteration. Otherwise, we keep the reference solution from the previous iteration, and discard all newly generated candidate shape patterns (the it-

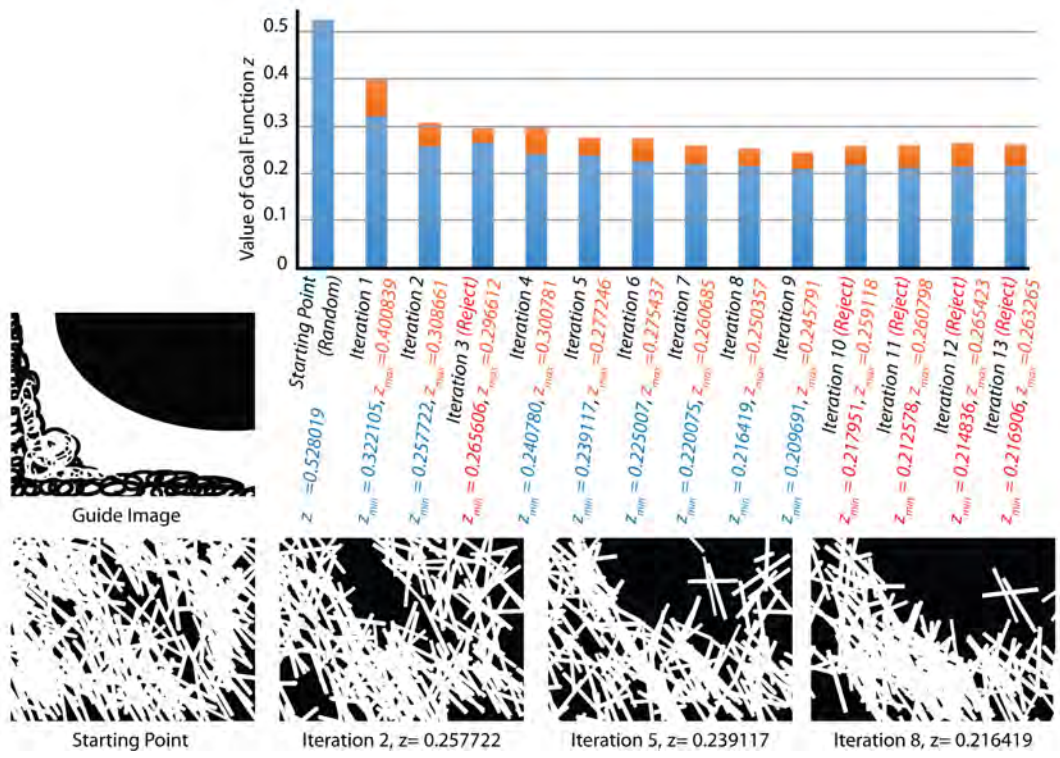



Figure 5: The stochastic optimization starts with a random shape pattern, and converges to a shape pattern that appears like the guide image. In this example, 12 new candidates are generated in each iteration; the lowest (blue) and highest (orange) goal function value of each iteration is illustrated in the top plot. Reference solutions (best shape pattern candidates) from some of the iterations are shown below.

erations marked with a red $z_{\min}$ value and a 'Reject' in the top plot of Figure 5). Our program visualizes the optimization process in real time by rendering the current reference solution onto the screen. The optimization keeps iterating until one of the following criteria is met:

1. The goal function $z$ of a shape pattern evaluates to a value below a given threshold $z_0$;
2. The maximum time set by the artist for the optimization has expired;
3. The artist decides to terminate the optimization because the current reference solution looks satisfying.

To efficiently evaluate the optimization, we propose to calculate the difference between a candidate shape pattern and the guide image *part by part* for optimization heuristics. We assume that if we can make the shape patterns look similar to the guide image in every part, then their overall appearance would also look similar. Following this idea, we first calculate a *difference image* $\mathbf{R}$ by subtracting a candidate shape pattern from the guide image:

$$\mathbf{R}_{x,y} = V(\mathbf{I}_{x,y}) - V(\mathbf{I}'_{x,y}).$$

The difference image $\mathbf{R}$ will have both positive and negative pixels, and is handled with a special implementation. We then create a set of *evaluation cells* $\mathbf{C}$ on $\mathbf{R}$ (Figure 4). Each evaluation cell $\mathbf{C}_i$ contains a square clip of the difference image. There are two kinds of evaluation cells:

1. A planar tiling of evaluation cells of the same size covers the entire image plane. They form the *evaluation lattice* on $\mathbf{R}$. The evaluation lattice is shown with dark brown lines in Figure 4(c). Cells in the evaluation lattice are created when the optimization starts.
2. *Instance Cells* are evaluation cells with their centers aligned with the geometric center of shape instances, which are represented by yellow squares in Figure 4(c). When we create a shape instance by duplicating an elemental shape and place the instance onto the shape pattern, we create an instance cell with it. When a shape instance is deleted, its corresponding instance cell is also removed.

Our program then calculates an *accumulated difference index* $g_i$ within the region of each evaluation cell $\mathbf{C}_i$:

$$g_i(\mathbf{C}_i) = \frac{\sum \mathbf{R}_{x,y}}{N_i} = \frac{\sum (\mathbf{I}_{x,y} - \mathbf{I}'_{x,y})}{N_i},\ (x,y) \in \mathbf{C}_i.$$

In the above equation, $g_i$ represents the difference of the shape pattern and the guide image in the local region of $\mathbf{C}_i$. $N_i$ denotes the total number of pixels in $\mathbf{C}_i$, so $g_i$ is normalized. If $g_i$ is positive, it means the shape pattern coverage needs to be increased in $\mathbf{C}_i$; and if $g_i$ is negative, it means the shape pattern covers too much area.

To guide the optimization with $g_i$, we set up a series of threshold values, and determine the appropriate adjustment to the shape pattern by comparing $g_i$ of a specific evaluation cell $\mathbf{C}_i$ to these threshold values. In the following list, $\lambda_{\text{cre}}$ and $\lambda_{\text{del}}$ are threshold values for creating and deleting shape instances. $\lambda_{\text{inc}}$ and $\lambda_{\text{rec}}$ denote threshold values for increasing or reducing a shape instance's rendering coverage. $\lambda_{\text{del}}$ and $\lambda_{\text{rec}}$ should have negative values, and the inequality $-1 < \lambda_{\text{del}} \leq \lambda_{\text{rec}} \leq 0 \leq \lambda_{\text{inc}} \leq \lambda_{\text{cre}} < 1$ should hold.

Given an evaluation cell $\mathbf{C}_i$ and its corresponding difference index $g_i$:

- if the evaluation cell $\mathbf{C}_i$ belongs to the evaluation lattice,
  - if $g_i > \lambda_{\text{cre}}$, we calculate a 'create' probability $p_{\text{cre}} = \frac{g_i - \lambda_{\text{cre}}}{1 - \lambda_{\text{cre}}}$. When generating a new candidate shape pattern, our program creates a new shape instance with the probability of $p_{\text{cre}}$. The initial geometric center of the newly created shape instance is a random point inside the evaluation cell $\mathbf{C}_i$. Other rendering transforms of the newly created shape instance, such as the rotation and scaling, are randomly initialized.
- if $C_i$ is an instance cell attached to a shape instance $i$,
  - if $g_i < \lambda_{\text{del}}$, we calculate a 'delete' probability $p_{\text{del}} = \frac{g_i - \lambda_{\text{del}}}{-1 - \lambda_{\text{del}}}$. When generating a new candidate shape pattern, our program deletes shape instance $i$ with the probability $p_{\text{del}}$.
  - if $g_i < \lambda_{\text{rec}}$, our program tries to decrease the coverage of shape instance $i$ on the shape pattern by reducing its scale parameter with a small random amount.
  - if $g_i > \lambda_{\text{inc}}$, our program tries to increase the coverage of shape instance $i$ by increasing its scale parameter with a small random amount.

Typically, an optimization iteration starts with creating the instance evaluation cells for the reference solution, continues with calculating the difference index $g_i$ for each cell, then generates multiple new candidate shape patterns by examining all difference indices and performing corresponding adjustments. Our program also adds a small amount of random perturbation on the rotation and translation transforms of every shape instance when creating a new candidate shape pattern. Although the evaluation cells and difference indices $g_i$ stay the same for the same reference solution, we can generate non-repetitive new candidates as long as we use independent random numbers through the process. For instance, we decide whether to create or delete a shape instance by drawing a random number from the uniform distribution of $[0,\ 1]$, and comparing it to $p_{\text{cre}}$ or $p_{\text{del}}$. Therefore, if the program discards all new candidates in an iteration because they all evaluate to larger $z$ than the reference solution, then it can keep using the evaluation cells and difference indices calculated in the previous iteration. As long as the program uses independent random numbers, it only needs to calculate the heuristics information once for one reference solution, until a better shape pattern is reached.

The random number generator we used throughout our program is the Mersenne Twister proposed by Saito and Matsumoto [SM08]. It is a pseudo-random number generator, which ensures our program to render exactly the same video as long as the seed number does not change.

### 4.2. Animation with Harmonic Motion

Harmonic motions are one of the most commonly observed physical movements. From a simple pendulum to complicated ocean waves, many phenomena can be modeled using harmonic motion [CK06]. Although our light patterns can be abstract shapes, we seek to render them with a coherent and organic animation. Therefore, we choose to use harmonic motion simulation to create the animation for the shape patterns.

To drive the harmonic motion simulation, we use the $\frac{1}{f^\beta}$ statistical model which resembles the frequency distribution of many natural phenomena [PS88]. A signal conforming to the $\frac{1}{f^\beta}$ model appears random in time domain, but exhibits a reciprocal curve similar to $\frac{1}{f^\beta}$ in its power spectrum, where $f$ denotes frequency and $\beta$ is a given constant. Smaller $\beta$ values result in more high-frequency turbulence in the signal, while larger values cause smoother oscillation (Figure 6). In our research, we are interested in this model particularly because it resembles the wind, and it creates the appearance of light shapes swaying in a breeze.

We propose a two stages method for creating the shape pattern animation. In the first stage, we generate a random signal that conforms to the $\frac{1}{f^\beta}$ statistical model. This is achieved by creating the absolute value part and the phase (argument) part separately in the frequency domain. The second stage runs a harmonic motion simulation with the signal created in the first stage as the excitation force.

*Stage 1.* Creating the $\frac{1}{f^\beta}$ excitation force.

Assuming that the animation contains $N$ frames, with $k$ frames per-second:

1. Set the first value of the frequency domain representation for the excitation to 0, since we need shape instances to

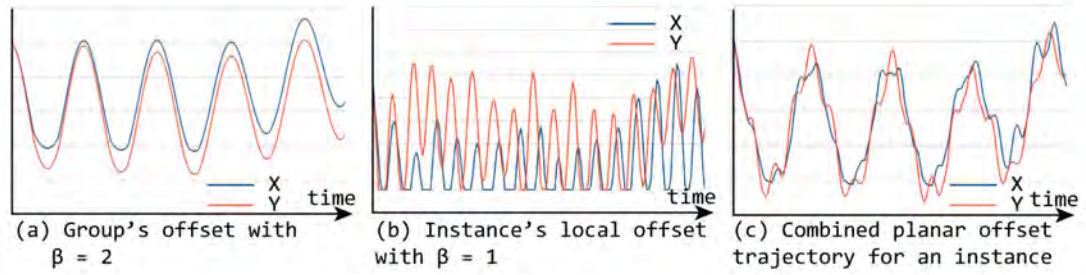



Figure 6: Harmonic motion simulation with the $\frac{1}{f^{\beta}}$ signal as the excitation. (a) The shape group trajectory, which is driven by a signal with $\beta = 2$ and moves in a planar space. (b) The local movement of a shape instance is driven by a signal with $\beta = 1$, resulting in much high frequency oscillation. The movement is also constrained in the shape's local movement range. (c) The combined shape instance trajectory over time. Each plot has individually scaled vertical axis, and in this example the local movement has a smaller range than the group movement. Time (horizontal) axes are of the same scale.

fluctuate around its original position and have no static offset.

2. Calculate the frequency step $f_{\lambda} = \frac{k}{N}$. In the frequency domain representation, the $i_{\text{th}}$ value stands for the sinusoidal component of frequency $if_{\lambda}$.
3. Generate $N/2$ random values for the absolute value part of the frequency domain representation. The values should approximately conform to the $\frac{1}{f^{\beta}}$ distribution (with the first value set to 0). For the $i_{\text{th}}$ absolute value:
   a. Construct a normal distribution $N(\mu,\ \sigma^2)$ with the mean $\mu = \frac{1}{(if_{\lambda})^{\beta}}$ and variance $\sigma = m\mu$, where $m$ is a small deviation factor ($<< 1$) given as a constant parameter.
   b. Use a Gaussian random number generator on top of the uniform random number generator to create a random value that conforms to the above normal distribution. In our implementation, the Box-Muller method [BM58] is used.

   The absolute values need to be greater or equal than zero, so the resultant random values are set to zero if the Gaussian random number generator gives negative values.
4. Generate $\frac{N}{2}$ uniform random values in the range of $[-\pi,\ \pi]$ to be used as the phase part of the frequency domain representation.
5. Combine the phase part with the absolute value part to form complete complex numbers, and extend the sequence to size $N$ with complex conjugation. Then, the time domain excitation is calculated using the inverse Fourier transform.

Since this sequence of $N$ excitation samples needs to be real numbers in the time domain, its frequency domain representation must be conjugation symmetric [OWY83]. Therefore, our program only needs to generate $\frac{N}{2}$ complex numbers in the frequency domain, and the high frequency half of the frequency domain representation can be deduced with complex conjugation. The random excitation samples generated from the above algorithm are in the range of $[0,\ 1]$. They are multiplied with given excitation magnitude values before being used to drive the harmonic motion simulation. If an indefinite length of animation is required instead of a fixed length of $N$ frames, our program simply continues to drive the harmonic motion from the start of an excitation sequence when the sequence is exhausted. Repeating the entire excitation sequence in the time domain does not alter its frequency domain $\frac{1}{f^{\beta}}$ distribution. In our implementation, we set $N = 2000$ and $k = 30$.

*Stage 2.* Creating the movement trajectory for the shape pattern's animation:

Instead of letting every shape instance move at an independent random direction, we create more coherent movement by making them move in groups. In the animation, we seek to create an appearance of shapes moving around in clusters while fluctuating individually. We set the number of groups according to the number of shape instances being rendered, and determine the groups of shape instances with the fuzzy C-Mean clustering algorithm (FCM) [Bez81]. Aesthetically, the shape groups should overlap each other a little, thus a fuzzy clustering algorithm is chosen over a binary clustering algorithm such as K-Means.

We implement the Duhamel's integral method for the simulation of a single degree of freedom harmonic motion with an arbitrary excitation [CK06]. As our program focuses on visual representation instead of physical accuracy, satisfying results are achieved using one simulation time step per rendering frame. The following list explains our animation algorithm in detail.

1. Arrange the shape instances into several groups by applying the FCM algorithm to their geometry center points. The FCM algorithm calculates a probability matrix, representing the probability of each shape instance belonging to a particular group. Our program determines the group of a shape instance by drawing a uniform random

number in the range of $[0,\ 1]$, and compares it to the group entries in the matrix, choosing the first group that has a greater accumulated probability than the random number.

2. Create the trajectory of the groups by simulating two one-dimensional harmonic motions, driven by independent excitation sequences. The movement of the shape groups are also confined in the image plane, like the geometry centers of the shape instances. One of the two excitations driving the planar translation movements is multiplied with a large magnitude value, resembling the major direction of group movement. For each frame, the offsets of the two harmonic motions are combined into a two-dimensional coordinate, then rotated by a given angle to create the group movement in the direction specified by the artist.
3. Create the trajectory of every shape instance by simulating five one-dimensional harmonic motions with independent random excitation. They are used as the movements on the three rotational axes plus two planar translation axes of a shape instance, which has its geometric center constrained inside a plane but is free to rotate and move otherwise. Excitations used in this step are multiplied with small magnitude values.

In our implementation, the group movements are driven by $\frac{1}{f^\beta}$ excitations with $\beta = 2$. The high $\beta$ value creates smooth movements that appear similar to sinusoidal trajectories with slow moving centers. The movements of shape instances are driven by excitations with $\beta = 1$, which create large amounts of high frequency oscillations. When rendering an animation frame, the movement of a shape instance is added on top of its group movement to produce the final rendering transform for that shape instance (Figure 6). The movements of shape groups and instances are also limited within given ranges. Given as parameters to our program, the movement ranges ensure that at any time during the animation, the shape patterns in the rendered video will not deviate too much from the guide image.

Our rendering program reads an image file as the guide image, and loads the elemental shapes from an Autodesk .fbx file [Aut14]. The artist using our rendering tool can design the guide image and elemental shapes with any external software tool. The program stores numerical parameters in an external XML file, such as the optimization heuristics thresholds ($\lambda_{\text{cre}}$, $\lambda_{\text{del}}$, etc.) and the intrinsic frequency and damping for simulating the harmonic motion. On a desktop computer with an Intel i7 processor and a Nvidia GTX 280 graphics card, the optimization process takes around 10 seconds per iteration, and one minute in total. The animation algorithm creates approximately 100 frames per-minute and writes them to the hard drive as image files. The exact performance depends on the complexity of the elemental shape, the number of shape instances and the rendering resolution. The entire process is visualized on the screen in real time, and the artist can pause or halt the program anytime during the optimization or animation stage.

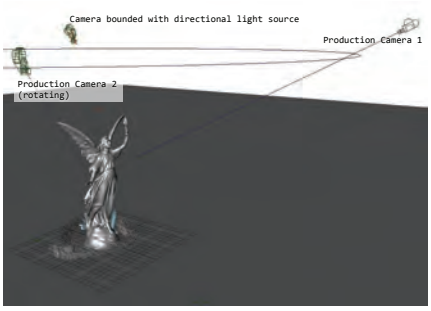
## 5. Rendering Foliage Shadows Under Sunshine

We introduce the proposed rendering method with a case study of a projection installation, but the application of our method is not limited to rendering abstract light patterns. The general idea behind our approach is to automatically fill in lighting details following an artist's sketch with a few given shape examples. To illustrate the effectiveness of the proposed method in a digital rendering context, we demonstrate rendering *foliage shadow effects* for virtual scenes with our method. In this example, we draw a lighting plan for a scene, and link the shape patterns rendered from our program as animated projective light masks in Autodesk Maya. We choose this example, because the light patterns on the ground under foliaged trees always consist of shadows from a large number of similar shaped tree leaves. Therefore, our rendering method can be applied, taking the tree leaves as elemental shapes, if we assume that we do not need to correlate the foliage shadows with a tree canopy in a photorealistic manner.

A bright sunny day brings vivid light and shadow patterns under trees. When a gentle breeze sweeps through the leaves, the complicated light patterns on the ground change their appearances in a visually pleasing movement. Because the densely foliaged trees have complicated structures, photorealistic methods for modeling and animating their shadows are usually sophisticated. With our approach, we can render foliage shadow effects without modeling the complicated tree structure. As a starting point, we create a scene in Maya with a directional light source, which represents the direction of sunshine (Figure 7(a)). To draw the lighting plan for this scene, we align a camera with the directional light source, and align the look-at direction of that camera with the light direction (Figure 7(b)). We render the scene from that camera, and draw the lighting plan on top of this image on a transparent layer (Figure 7(c)). In this example, the lighting plan drawing indicates a shadowy surrounding and a bright Lucy statue. The layer with the lighting plan drawing is extracted as the input guide image for our program. We used Adobe Photoshop [Ado08] for drawing the lighting plan image. The elemental shapes for this example will be a few models of maple leaves.

Instead of rendering the shapes as light patterns, we render black shapes against a white background. Colors for the shape pattern and the background can be set in the configuration file of our rendering program. The output animation frames are then linked as the projective light mask on the directional light source (Figure 7(d)). The leaf-shaped elemental shapes for shadows are set to perform much local rotational movements, which appears like tree leaves fluctuating on the tips of tree branches. The groups of shadow shapes in a same scene are set to move in similar direc-

(a) Scene with the Lucy statue.




(b) Light source and the attached camera.

(c) Artist draws the lighting plan on top of the scene.

(d) Final rendering.

Figure 7: The foliage shadow effects example. We align a camera with the light source (b), and draw the lighting plan on top of a rendering from this camera (c). Using the proposed rendering method, we create plausible foliage shadow effects, and keep the Lucy statue bright in shadowy surroundings (d) as indicated in the lighting plan.

tions, which resembles foliage clusters on tree brunches swaying gently in wind. Harmonic motion and the $\frac{1}{f^{\beta}}$ excitation are shown to be effective in modeling plant movements [HKW09, OTF*03]. Our final rendering conforms to the lighting plan drawing, at the same time exhibiting detailed tree foliage shadows and plausible animation.

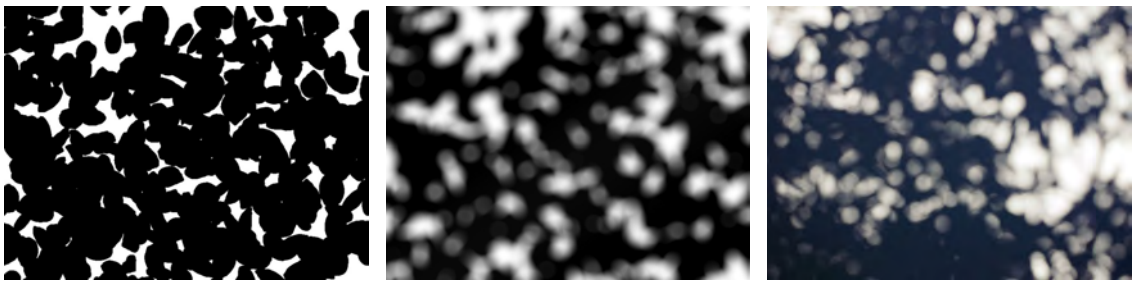
### 5.1. Soft Shadows under Sunshine

A remarkable visual effect of the shadows under foliaged trees are the circular shape of bright patterns regardless of the complex shapes of slits among the foliage (Figure 8(c)). This phenomenon happens because the slits inside the foliage act as pin-hole cameras and project the sun's images onto the ground at proper distances. In addition, foliage shadows usually look blurry because the sun is an area light source. In our research, we simulate these two important visual features with a blur filter.

Viewed from Earth, the sun has a constant angular diameter of approximately $\theta = 0.5$ degree [SB10]. If the assumed distance from the shadow receiving plane to the shape patterns is $d$, then the shadows on the receiving plane should be blurred with a circular kernel of radius $d\tan\theta$. We calculate the blur as a convolution of the sharp rendering result with a filter image [SS98]. The filter image is initialized as all-black with a circular white spot of diameter $d\tan\theta$ at the center. The artist can pick another image as the filter, such as an astronomy photo of the sun [Max91]. The pin-hole camera effect naturally emerges when proper distance parameters are set (Figure 8(b)).

In our implementation, the blur convolution is executed in the frequency domain as a multiplication, with corresponding forward and inverse two-dimensional Fourier transforms. As the distance parameter does not change throughout the animation, the frequency domain filter image is precalculated. The Fourier transforms and texture multiplications are executed on GPU with Microsoft Direct 11 Compute Shader [Mic15], which provides a fast and constant rendering speed when the blur radius is large. Another example scene shows the Buddha statue rendered with foliage shadow effects in Figure 9.

Our method of rendering foliage shadows has two advantages: first, an artist is exempted from modeling the geometric structure for the caster of the light patterns, such as a complicated tree. Secondly, the rendering result is guaranteed to resemble the lighting plan drawing, because the light pattern is motivated from the drawing instead of being calculated from photorealistic light-object interactions. Artists who worked with us evaluated our approach as "straightforward, controllable and effective", and suggested that a wide variety of light pattern effects could be rendered with our method, such as underwater caustics lights and indoor light patterns from mirror balls. The common property of these scenes is that the lighting can be modeled as shape patterns



(a) Sharp rendering of a shape pattern with oval leaves.

(b) Shape pattern blurred with a radius of 20 pixels.

(c) Photograph of real leaf shadows on a white foam core.

Figure 8: An example of the blur filter in foliage shadow effects rendering, and comparison with real life photograph.

containing limited number of elemental shapes. Additional algorithmic components would be needed to adapt our rendering method into these different scenarios, just as the blur filter we created in the foliage shadows example.

## 6. Limitation and Future Work

A limitation of our rendering method is that the shape patterns only resemble the overall appearance of the guide image. If the given elemental shapes are large, the generated shape patterns cannot reproduce every detail in the input image. When the shape patterns are animated, the rendering result may further deviate from the guide image in some parts.

In our research, we have worked with artists of various disciplines, from fine arts to the visual effects industry. We found artists with different backgrounds have different expectations of rendering software tools. Artists from a fine art program prefer to concentrate on the overall visual appearance when creating images, and feel uncomfortable with a sophisticated numerical interface. Thus, we packed all the numerical parameters into an XML file that is not frequently changed, and let artists focus on drawing the lighting plan and modeling the elemental shapes. On the other hand, our artist colleagues from the visual effects industry point out that extensive control over the rendered image is necessary for CGI production. With professional experience in making computer games and animation films, they asked for a well designed numerical interface, which exposes details of the internal rendering mechanism. Therefore, we propose to develop a numerical interface on top of the current XML configuration file. More importantly, a series of formal user studies should be conducted with artists from different disciplines to better understand their various expectations of rendering software tools.

## 7. Conclusion

The major contribution of the proposed sketch-based rendering method is its easy and intuitive workflow, in which an artist is not required to laboriously make every detail of the rendered animation. Supplied with only a few paint-strokes and elemental shapes, our method automatically renders richly detailed shape patterns. This approach resembles a result motivated rendering methodology: instead of starting from a scene model and calculating light-object interaction to deduce the rendered image, we start with a target appearance of the depicted scene (the guide image), and create the lighting details with animation.

We have demonstrated the effectiveness of the proposed method in two distinctive rendering contexts. In our research team, professional artists and computer scientists work side by side to create a contemporary art installation and render foliage shadows effects. During this process, we constantly evolved the algorithmic design of our rendering program, resulting in the method described in this work. Our approach is different from typical computer graphics research, which solves a defined theoretical or practical problem then invites external artists to validate the solution. We hope both our rendering method and our research approach will inspire aesthetic creations with computer graphics technologies in the future.

## Acknowledgements

Many thanks to Professor. Paul Walde and Mr. Cliff Haman for their generous instruction and support for our projection installation. Thanks to our lab colleagues for their volunteered time and effort for putting up and disassemble the installation on the scheduled time. This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

## References

[Ado08] ADOBE SYSTEMS INCORPORATED: Photoshop, creative suite 4. Software, 2008. 9

[AKA13] ALMERAJ Z., KAPLAN C. S., ASENTE P.: Patch-based geometric texture synthesis. In *Proceedings of the Symposium on Computational Aesthetics* (New York, NY,

(a) Scene with the Buddha statue. (b) Scene rendered from the light source. (c) The lighting plan drawing.

(d) Light patterns with two layers. (e) Scene rendered with the foliage shadow effect from the light source. (f) Scene rendered with the foliage shadow effect from another viewpoint.

Figure 9: Foliage shadow effect rendered in a scene with the Buddha statue. In this example, the elemental shape is the bamboo leaf, and the shadow effect consists of two layers of shape patterns with different degrees of blur (d). The lighting plan (c) resembles a shadowy surrounding, a partially shadowed rooftop and a bright Buddha statue. In the final rendering (e,f), this lighting design is preserved.

USA, 2013), CAE '13, ACM, pp. 15–19. URL: http://doi.acm.org/10.1145/2487276.2487278, doi:10.1145/2487276.2487278. 2

[Aut14] AUTODESK, INC.: FBX data exchange techonology, 2014. URL: http://www.autodesk.com/products/fbx/overview. 9

[Bez81] BEZDEK J. C.: *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981. 8

[BM58] BOX G. E. P., MULLER M. E.: A note on the generation of random normal deviates. *The Annals of Mathematical Statistics 29*, 2 (06 1958), 610–611. URL: http://dx.doi.org/10.1214/aoms/1177706645, doi:10.1214/aoms/1177706645. 8

[BNB13] BRADLEY D., NOWROUZEZAHRAI D., BEARDSLEY P.: Image-based reconstruction and synthesis of dense foliage. *ACM Trans. Graph. 32*, 4 (July 2013), 74:1–74:10. URL: http://doi.acm.org/10.1145/2461912.2461952, doi:10.1145/2461912.2461952. 3

[CGZ*05] CHUANG Y.-Y., GOLDMAN D. B., ZHENG K. C., CURLESS B., SALESIN D. H., SZELISKI R.: Animating pictures with stochastic motion textures. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 853–860. URL: http://doi.acm.org/10.1145/1186822.1073273, doi:10.1145/1186822.1073273. 3

[Chr76] CHRISTO AND JEANNE-CLAUDE: Running fence, 1972-76. "Running Fence was 18 feet (5.5 meters) high and 24.5 miles (39.4 kilometers) long. The art project consisted of 42 months of collaborative efforts, 18 public hearings, three sessions at the Superior Courts of California, the drafting of a 450-page Environmental Impact Report and the temporary use of the hills, the sky and the ocean at California's Bodega Bay." Quoted from the author's website. URL: http://christojeanneclaude.net/projects/running-fence. 3

[CK06] CRAIG R. R., KURDILA A. J.: *Fundamentals of structural dynamics*, 2nd ed. Wiley.com, 2006. Chapter.6 Numerical Evaluation of the Dynamic Response of SDOF Systems. 7, 8

[dB09] DE BROIN M.: La maîtress de la tour eiffel, 2009. Metal structure, 1000 mirrors, 5 light projectors, crane. URL: http://micheldebroin.org/la-maitresse-de-la-tour-eiffel. 2

[Goo15] GOOGLE INC.: Google maps web service, 2015. URL: https://www.google.com/maps/. 3

[HKW09] HABEL R., KUSTERNIG A., WIMMER M.: Physically guided animation of trees. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 523–532. 3, 10

[HLT*09] HURTUT T., LANDES P.-E., THOLLOT J., GOUSSEAU Y., DROUILLHET R., COEURJOLLY J.-F.: Appearance-guided synthesis of element arrangements by example. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2009), NPAR '09, ACM, pp. 51–60. URL: http://doi.acm.org/10.1145/1572614.1572623, doi:10.1145/1572614.1572623. 2

[KP09] KERR W. B., PELLACINI F.: Toward evaluating

lighting design interface paradigms for novice users. *ACM Trans. Graph. 28*, 3 (July 2009), 26:1–26:9. URL: http://doi.acm.org/10.1145/1531326.1531332, doi:10.1145/1531326.1531332. 2

[LBW*14] LU J., BARNES C., WAN C., ASENTE P., MECH R., FINKELSTEIN A.: Decobrush: Drawing structured decorative patterns by example. *ACM Trans. Graph. 33*, 4 (July 2014), 90:1–90:9. URL: http://doi.acm.org/10.1145/2601097.2601190, doi:10.1145/2601097.2601190. 2

[LDS*11] LI C., DEUSSEN O., SONG Y.-Z., WILLIS P., HALL P.: Modeling and generating moving trees from video. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (New York, NY, USA, 2011), SA '11, ACM, pp. 127:1–127:12. URL: http://doi.acm.org/10.1145/2024156.2024161, doi:10.1145/2024156.2024161. 3

[LH15] LOZANO-HEMMER R.: Pluse Corniche, 2015. Searchlights, heart rate sensor, computer and speakers. URL: http://www.lozano-hemmer.com/pulse_corniche.php. 2

[LPC*11] LIVNY Y., PIRK S., CHENG Z., YAN F., DEUSSEN O., COHEN-OR D., CHEN B.: Texture-lobes for tree modelling. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 53:1–53:10. URL: http://doi.acm.org/10.1145/1964921.1964948, doi:10.1145/1964921.1964948. 3

[Max91] MAX N.: Unified sun and sky illumination for shadows under trees. *CVGIP: Graphical Models and Image Processing 53*, 3 (1991), 223 – 230. URL: http://www.sciencedirect.com/science/article/pii/104996529190044K, doi:http://dx.doi.org/10.1016/1049-9652(91)90044-K. 10

[Mic15] MICROSOFT CORPORATION.: Direct x 11 compute shader. Software, 2015. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/ff476331%28v=vs.85%29.aspx. 10

[MWT11] MA C., WEI L.-Y., TONG X.: Discrete element textures. *ACM Trans. Graph. 30*, 4 (July 2011), 62:1–62:10. URL: http://doi.acm.org/10.1145/2010324.1964957, doi:10.1145/2010324.1964957. 2

[OMSI07] OKABE M., MATSUSHITA Y., SHEN L., IGARASHI T.: Illumination brush: Interactive design of all-frequency lighting. In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on* (2007), IEEE, pp. 171–180. 2

[OTF*03] OTA S., TAMURA M., FUJITA K., FUJIMOTO T., MURAOKA K., CHIBA N.: 1/fb noise-based real-time animation of trees swaying in wind fields. In *Computer Graphics International, 2003. Proceedings* (2003), IEEE, pp. 52–59. 10

[OWY83] OPPENHEIM A., WILLSKY A., YOUNG I.: *Signals and systems*. Prentice-Hall signal processing series. Prentice-Hall, 2nd Edition, 1983. URL: http://books.google.ca/books?id=UQJRAAAAMAAJ. 8

[PBMF07] PELLACINI F., BATTAGLIA F., MORLEY R. K., FINKELSTEIN A.: Lighting with paint. *ACM Trans. Graph. 26*, 2 (June 2007). URL: http://doi.acm.org/10.1145/1243980.1243983, doi:10.1145/1243980.1243983. 2

[Pel10] PELLACINI F.: envylight: An interface for editing natural illumination. In *ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 34:1–34:8. URL: http://doi.acm.org/10.1145/1833349.1778771, doi:10.1145/1833349.1778771. 2

[PHL*09] PALUBICKI W., HOREL K., LONGAY S., RUNIONS A., LANE B., MĚCH R., PRUSINKIEWICZ P.: Self-organizing tree models for image synthesis. *ACM Trans. Graph. 28*, 3 (July 2009), 58:1–58:10. URL: http://doi.acm.org/10.1145/1531326.1531364, doi:10.1145/1531326.1531364. 3

[PLH*90] PRUSINKIEWICZ P., LINDENMAYER A., HANAN J. S., FRACCHIA F. D., FOWLER D. R., DE BOER M. J., MERCER L.: *The Algorithmic Beauty of Plants*, vol. 2. Springer-Verlag New York, 1990. 3

[PRJ97] POULIN P., RATIB K., JACQUES M.: Sketching shadows and highlights to position lights. In *Computer Graphics International, 1997. Proceedings* (jun 1997), pp. 56 –63. doi:10.1109/CGI.1997.601272. 2

[PS88] PEITGEN H.-O., SAUPE D. (Eds.): *The Science of Fractal Images*. Springer-Verlag New York, Inc., New York, NY, USA, 1988. 3, 7

[PSK*12] PIRK S., STAVA O., KRATT J., SAID M. A. M., NEUBERT B., MĚCH R., BENES B., DEUSSEN O.: Plastic trees: Interactive self-adapting botanical tree models. *ACM Trans. Graph. 31*, 4 (July 2012), 50:1–50:10. URL: http://doi.acm.org/10.1145/2185520.2185546, doi:10.1145/2185520.2185546. 3

[Ros12] ROSENTHAL N. E.: *Winter Blues: Everything You Need to Know to Beat Seasonal Affective Disorder (4th Edition)*. Guilford Press, 2012. 3

[RTD*10] RITSCHEL T., THORMÄHLEN T., DACHSBACHER C., KAUTZ J., SEIDEL H.-P.: Interactive on-surface signal deformation. *ACM Trans. Graph. 29*, 4 (July 2010), 36:1–36:8. URL: http://doi.acm.org/10.1145/1778765.1778773, doi:10.1145/1778765.1778773. 2

[San13] SANDISON C.: Terrestrial echo of solar storms, 2013. Customized computer programs, data projectors, the Central Exhibition Hall 'Manege', Moscow, Russia. 2

[SB10] SEEDS M., BACKMAN D.: *Foundations of Astronomy*. Cengage Learning; 11st edition, 2010. 10

[SDS*93] SCHOENEMAN C., DORSEY J., SMITS B., ARVO J., GREENBERG D.: Painting with light. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 143–146. URL: http://doi.acm.org/10.1145/166117.166135, doi:10.1145/166117.166135. 2

[SM08] SAITO M., MATSUMOTO M.: Simd-oriented fast mersenne twister: a 128-bit pseudorandom number generator. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Springer, 2008, pp. 607–622. 7

[Sni76] SNIDER G.: Rough Trade, 1976. Douglas fir, steel, 15'x15'x18'. URL: http://www.gregsnider.ca/sculpture/pre81/rough_trade.html. 3

[SRHC08] SONG Y.-Z., ROSIN P. L., HALL P. M., COLLOMOSSE J.: Arty shapes. In *Proceedings of the Fourth Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging* (Aire-la-Ville, Switzerland, Switzerland, 2008), Computational Aesthetics'08, Eurographics Association, pp. 65–72. URL: http://dx.doi.org/10.2312/COMPAESTH/COMPAESTH08/065-072, doi:10.2312/COMPAESTH/COMPAESTH08/065-072. 3

[SS98] SOLER C., SILLION F. X.: Fast calculation of soft shadow textures using convolution. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 321–332. URL: http://doi.acm.org/10.1145/280814.280927, doi:10.1145/280814.280927. 10