A Painterly Rendering Approach to Create Still-Life Paintings with Dynamic Lighting

Meena Subramanian
Department of Visualization, Texas A&M University
College Station, Texas
meena.subramanian18@tamu.edu

Ergun Akleman
Departments of Visualization & Computer Science and Engineering, Texas A&M University
College Station, Texas
ergun.akleman@gmail.com

(a) *Original Painting.* **(b)** *Frame 15 of our animation.* **(c)** *Frame 28 of our animation.* **(d)** *Frame 87 of our animation.*

Figure 1: *An example of emulating a still life painting with dynamically changing shadows, specular highlights and caustics using a moving area light source. Our method allows direct artistic control to emulate look-and-feel of of the original painting.*

ABSTRACT

In this work, we present a method to turn still life paintings with global illumination effects into dynamic paintings with moving lights. Our method specifically focuses on still life images containing (1) glass objects, which can have specular highlights with Fresnel effect and (2) fruits, which can have reflection and subsurface scattering. Our goal is to preserve the original look-and-feel of the still-life paintings while allowing the user to move the light source anywhere in the scene, causing the shadows, diffuse shading, as well as reflections and specular highlights to move according to the new light position. We have provided a proof of concept based on an original digital painting. This method can be used to turn any similar still life painting into a dynamic painting.

CCS CONCEPTS

• **Computing methodologies** → **Non-Photorealistic Rendering**.

KEYWORDS

Rendering, Non-Photorealistic Rendering, Glass, Still Life Painting, Specular Highlights, Reflection

ACM Reference Format:
Meena Subramanian and Ergun Akleman . 2020. A Painterly Rendering Approach to Create Still-Life Paintings with Dynamic Lighting. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Posters (SIGGRAPH '20 Posters), August 17, 2020.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3388770.3407403

SIGGRAPH '20 Posters, August 17, 2020, Virtual Event, USA

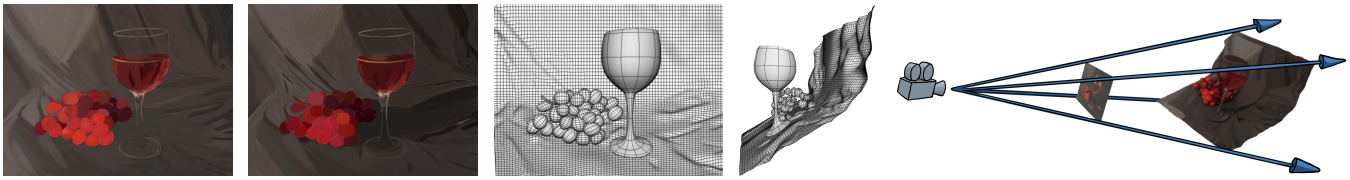
ACM ISBN 978-1-4503-7973-1/20/08.
https://doi.org/10.1145/3388770.3407403

1 INTRODUCTION AND MOTIVATION

Non-photorealistic rendering methods do not usually consider global illumination effects. However, global illumination effects such as reflections and refractions are an integral part of still life painting in particular. There have been several recent works to create dynamic paintings with global illumination effects using Barycentric shading [Akleman et al. 2016; Du and Akleman 2017]. However, these works only include physical reflections from other objects and do not include painterly control over reflection, Fresnel, or subsurface scattering. To obtain dynamic still life paintings, there is a need to have control over such global illumination effects. In this work, we have developed a process by extending Barycentric shading to control a wider variety of global effects including reflection, Fresnel, and subsurface scattering. An important part of our process is that it is still essentially simple to use and provides art-directable control to users. In this work, we have developed a process that uses an extended Barycentric shader. Our process of obtaining dynamic still life painting consists of six steps: (1) Creation of Control Paintings; (2) Modeling Proxy Geometry and Reflective Properties; (3) Painting Reflection Maps; (4) Projecting Control Images on to Proxy Geometry; (5) Building Reflection Planes; and (6) Rendering and Animation. Each of these steps is simple and intuitive. Any 2D artist with minor training in 3D modeling and animation can potentially produce a dynamic still life painting using this process.

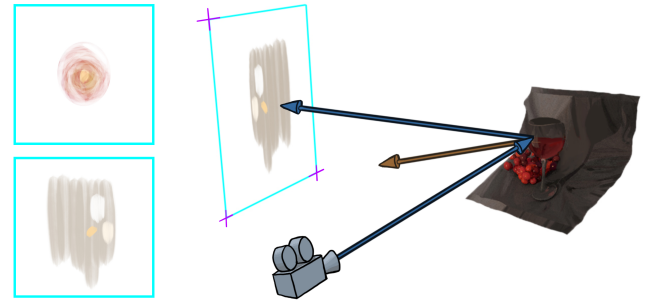
2 PROCESS

The first step, creation of control paintings, is the creation of two control images that are used to calculate the diffuse lighting. If



(a) *Diffuse control image T_0.* **(b)** *Shadow control image T_1.* **(c)** *Proxy Geometry Front View.* **(d)** *Proxy Geometry Side View.* **(e)** *Projection of textures to proxy geometry.*

Figure 2: *Two texture images T_0 and T_1 and front and side views of the proxy geometry that is used to compute illumination and shading.*



(a) *Reflection maps of grape and glass.* **(b)** *Visualization of specular calculation, with reflection ray intersecting the reflection plane in the scene.*

Figure 3: *Computing reflections.*

there is only one key light in the scene [Akleman et al. 2016], we just need two control images one for shadow, called T_0 and another one for fully illuminated diffuse T_1 (see Figures 2a, and 2b respectively). These two texture images can be created by using any digital painting software. For the diffuse image, the goal is to make every part of the image look as though it's illuminated by the light source. To accomplish this, we can paint over or color correct the areas that were painted to be in shadow, or had a specular highlight, reflection, or caustic. For the shadow image we need to do the opposite, but still eliminate all the global illumination effects.

The goal of the second step, modeling proxy geometry and reflective properties,is to have the boundaries of the 3D models match the silhouette edges of the objects in the original painting, so that the projections of the diffuse and shadow images onto the models will line up precisely (see Figures 2c, and 2d). As long as the boundaries of the models match those of the control images, the rest of the geometry can be a quick approximation of the form and does not need to be perfectly accurate. Notice in Figure 2c the geometry looks good from the front view, which we'll be rendering from. However, it is clearly an approximation from the side view as shown in Figure 2d. We also assign reflection coefficient k_s and index of refraction coefficient η values to each object to control the strength of reflections.

The third step, painting reflection maps, along with the first step, is what gives the artist full control of the artistic style of the final image. For the reflection map, we essentially paint the shape of just the specular highlight or reflection of an object with no diffuse or shadow information. Opacity is a critical part of making these maps look nice in the final result; many of my maps for the grapes are opaque in the center, i.e. $\alpha \approx 1$, and then fade out around the edges, i.e. $\alpha \approx 0$. The Figure 3a shows examples of reflection maps created for a grape and the glass.

The forth step, projecting control images on to proxy geometry, is to project the control images onto the geometry. Since the boundaries of the proxy geometry match with control images, this can be done by a simple camera projection as shown in Figure 2e. Both the diffuse and shadow images will have their own separate projections, and the Barycentric shader blends between these images.

The fifth step of the process, building reflection planes, is to set up reflection planes that will be used to reflect painted highlights (See Figure 3b).To allow for the reflections to be calculated through our shader we first set up a "reflection plane" using three locators to define the boundaries of a square plane. The locator positions are inputs to our shader that allow us to calculate the length, width, and center of the plane. The light source for computing the diffuse reflection is also located at the center of this reflection plane. By moving this group of locators around in the scene, an artist can effectively move the "light source" around. This group of locators can be static or animated, and is what allows our result to have dynamic lighting.

The last step of the process is rendering and animation. We developed an extended Barycentric shader (See [Akleman et al. 2016]) that uses four shading parameters, which are all positive numbers between 0 to 1. The first parameter t is used to compute diffuse illumination. It represents what percentage of light can reach a given point and it is computed as $t = max(\cos\theta, 0)$. The other three parameters are used to compute the specular color. Let k_s denote reflection coefficient of any given shading point, let f denote Fresnel coefficient for given incident angle and index of refraction η, and let α denote transparency of texture position, (u_r, v_r) on the reflection map that is computed as the intersection of reflected ray with the reflection plane. We compute these parameters by using basic ray tracing. Based on these parameters, we render an image by following Barycentric shading equation:

$$C_d(t,u,v) = (T_0(u,v)(t)+T_1(u,v)(1-t))(1-k_s\alpha f)+k_s\alpha f C_R(u_r,v_r).$$

REFERENCES

Ergun Akleman, S Liu, and Donald House. 2016. Barycentric shaders: Art directed shading using control images. In *Proceedings of Expressive'2016*. Eurographics Association, Lizbon, Portugal, 39–49.

Yuxiao Du and Ergun Akleman. 2017. Designing look-and-feel using generalized crosshatching. In *ACM SIGGRAPH 2017 Talks*. ACM, New York City, NW, 40.