A Painterly Rendering Approach to Create Still-Life Paintings with Dynamic Lighting
Meena Subramanian and Ergun Akleman; Texas A&M University

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.

1. Introduction and Motivation

Non-photorrealistic rendering methods do not usually consider global illumination effects such as reflections and refractions, which are an integral part of still life paintings. There have been several recent works to create dynamic paintings with global illumination effects using Barycentric shading [1, 2]. However, these works only include physical reflections from other objects and do not include painterly control over global illumination effects. 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. In this work, we have developed a process to obtain dynamic still life painting. This process is still essentially simple to use and provides art-directable control to users in six well-defined steps.

2. Description of Process

Diffuse control image $T_0$, Shadow control image $T_1$

Figure 2: Two texture images $T_0$ and $T_1$ that are used to compute illumination and shading.

1. Creation of Control Paintings: This step is the creation of control images that are used to calculate the diffuse lighting. If there is only one key light in the scene [1], we just need two control images one for shadow, called $T_1$; and another one for fully illuminated diffuse $T_0$ (see Figure 2). These two texture images can be created by using any digital painting software by adjusting colors of original paintings and hand painting some areas as needed. The main goal of creating the control images is to take out any abnormalities in the color of the objects. 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 of the highlights, reflections, and caustics.

2. Modeling Proxy Geometry and Reflective Properties: The goal of this step 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 Figure 3). 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 3 the geometry looks good from the front view, which we’ll be rendering from. However, it is clearly an approximation from the side view. We also assign reflection coefficient $k_i$ and index of refraction coefficient $n_i$ values to each object to control the strength of reflections.

3. Painting Reflection Maps: This step, along with the creation of the diffuse and shadow images, 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. $a = 0$, and then fade out around the edges, i.e. $a = 1$. The Figure 4 shows examples of reflection maps for a grape and the glass.

4. Projecting Control Images on to Proxy Geometry: This step of the process 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 3. Both the diffuse and shadow images will have their own separate projections, and the Barycentric shader blends between these images.

5. Building Reflection Planes: This step of the process is to set up reflection planes that will be used to reflect painted highlights (See Figure 4). 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.

6. Rendering and Animation: We developed a Barycentric shader with four shading parameters [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 also percentages. Let $k_i$ denote reflection coefficient of any given shading point, let $j$ denote Fresnel coefficient for given incident angle and index of refraction $n_i$, and let $\alpha$ denote transparency of texture position. $(u_i, v_i)$ 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 images by using the following Barycentric shading equation:

$$C_d(u,v) = (T_0(u,v)) + T_1(u,v)(1 - t)(1 - k_f)) + k_f j C_d(u_i,v_i)$$

Figure 3: The proxy geometry that is used to compute illumination and shading.

Figure 4: Visualization of specular calculation, with reflection ray intersecting the reflection plane in the scene.

References
