

Tweaking stylized light and shade

Ken Anjyo*
OLM Digital, Inc.

Shuhei Wemler†
Silicon Studio Corporation

1. Introduction

Light and shade in non-photorealistic rendering, such as for digital cel animation, should be symbolic, rather than realistic, in order to depict the scene in a simple, yet impressive way. This sketch presents a new method of creating and editing the stylized light and shade in real-time by direct manipulation for an on-screen surface to be cartoon-shaded.

The manipulation simply means dragging the shaded area including highlight on the surface. The drag operations then offer an intuitive, interactive tool not only in changing the light source, but also in designing the stylized highlight shape. Our prototype system includes the algorithms in this method along with a real-time cartoon shader, and is built up with programmable graphics hardware.

2. Basic idea

In this sketch we consider the cases where 3D models are cartoon-shaded and merged into a traditional cel animation. For simplicity we assume that the light source is directional. Our basic strategy is to develop the algorithms of making the following works possible by drag operations onto the on-screen image: (a) Changing the directional light source, (b) Designing the highlight shape.

For (a), we first pick the two points \mathbf{p}_A and \mathbf{p}_B on a surface, so that $L(\mathbf{p}_A)$ is transported at \mathbf{p}_B , where $L(\mathbf{p}_A)$ is the local representation of the (input) directional light L_d with regard to the local coordinate system at \mathbf{p}_A . The new directional light L_d^* is then derived from $L(\mathbf{p}_A)$ in the local coordinate system at \mathbf{p}_B . The algorithm of tweaking the shaded area simply means to get L_d^* from L_d through pick and drag operations. As shown in our demonstration video, \mathbf{p}_B is the result of dragging \mathbf{p}_A , so that changing the directional light can be done smoothly in real-time. The left and center images in Figure 1 are taken from this video where tweaking the shaded area means changing the light source.

As for highlight shape design and animation, we developed a procedural approach in [Anjyo and Hiramitsu 2003]. This approach lets us make stylized highlights through several functional operations for translation, rotation, directional scaling, split, squaring, and Boolean operations among highlight areas. However, this requires a user to tune the parameters of these operations many times to get a desired result. Therefore we extend this approach, in order to replace laborious parameter tuning by intuitive drag operations. This extension is achieved by reformulating the functional operations in the original approach, because the original function parameters cannot be estimated by drag operations. An initial stylized highlight in our method is obtained by estimating the inner product of the normal vector and the halfway vector at a visible point. The highlight design algorithm then allows us to intuitively edit the highlight in real-time by the dragging operations associated with the highlight transform selected. For example, in making the window reflection effect in the right image of Figure 1, we selected translation, squaring and split operations as highlight transforms and created this image from the center image of Figure 1, which includes the original highlight. We

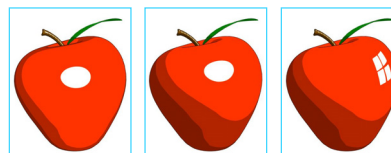


Figure 1: Tweaking shaded area and highlight

The directional light is changed simply by dragging the shaded area (left to center) on the apple. The rounded highlight (white area) is then continuously deformed by the drag operations (center to right), showing window reflection.

also note that, according to our new formulation of the functional operations, highlight shape deformation faithfully follows the drag operations. For instance, if a boundary point of the highlight area is picked and dragged, then the deformed highlight area always keeps the dragged point on its boundary during the operation. This would be a great help to a user in designing and editing the highlight.

3. Implementation and results

Our prototype system provides a real-time cartoon shader based on Blinn's classical shading model. Real-time keyframe animations are created by this system. Our system runs at interactive rates on 2.4Ghz Intel P4 CPU with nVidia Geforce FX5900 GPU. The rotation, translation, and directional scaling are done in the vertex shader, whereas the squaring and split are in the pixel shader. The frame rate ranges from 20 to 40 fps for the animations shown in this sketch.

Figure 2 illustrates practical effectiveness of this system, showing the frames from the three different animations created. The original one (left in Figure 2) was made with the conventional cartoon shader. In the second one (center in Figure 2), the shaded area was moved from the original position at each keyframe with the drag operations. The left image in Figure 2 is a frame from the third animation, which shows a highlight variation made by the similar drag operations. In this way, we can create a variety of stylized light and shade easily in real-time, without suffering from tedious parameter tuning.

Reference

ANJYO, K. AND HIRAMITSU, K. 2003. Stylized Highlights for Cartoon Rendering and Animation. *IEEE Computer Graphics and Applications* 23, 4, 54-61.



Figure 2: Frames from real-time animation examples.

Each of these images is taken from a different real-time animation with the same camera path. The shaded area is tweaked (from left to center) and the highlight is further deformed (from center to right). These variations are quickly made by real-time keyframing with the intuitive drag operations.

*e-mail: anjyo@olm.co.jp

†e-mail: wemler@siliconstudio.co.jp