Stylized Highlights for Cartoon Rendering and Animation

Ken-ichi Anjyo and Katsuaki Hiramitsu
OLM Digital

A novel highlight shader depicts cartoon-style highlights for 3D objects in cel animation. This shader creates cartoon-style highlighting through simple operations defined for the highlight vector field.

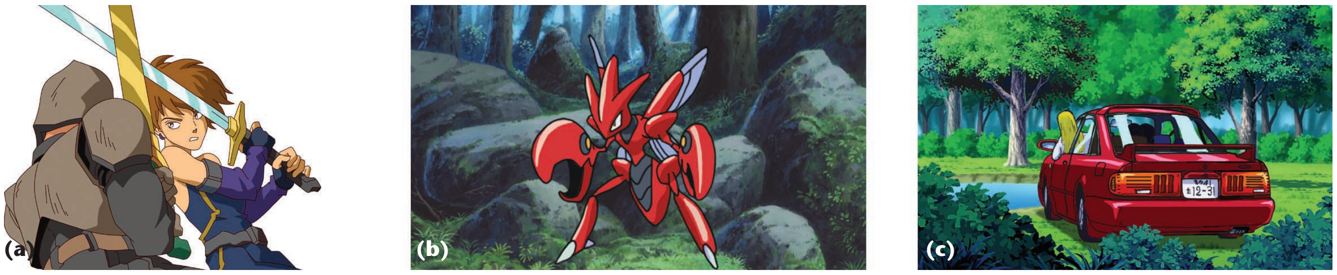
In traditional cel animation, lights and shadows are symbolic in that they imply an artistic interpretation of the characters and scene. For example, lighting plays an important role in suggesting a scene's mood, while a character's shadow shows where the character is standing. Highlighting also describes various aspects of the characters and objects in the scene. The highlight examples shown in Figure 1 were made with traditional cel animation techniques. In Figure 1a, we see a highlight on the swords, portraying that the swords are flat and shiny like plane glasses, and that they are so sharp that the heroine might be wounded in the next frame. The highlight on the monster's claws in Figure 1b suggests that the claws are very hard, and that they can hurt someone easily. In Figure 1c the highlight on the rear window of a car shows that the window is somewhat rounded, since the highlight area is not a simple rectangle, but a little bit deformed. This highlight could be a kind of environment reflection or refraction, rather than the brightest area on the rear window. In addition the highlight just shows the area where something is reflected; we don't see what is actually reflected on the window.

As these examples show, a highlight for cel animation must be a semantic notation rather than a part of physics. The highlight shape is relatively simple but not always rounded, suggesting stylistic, hand-drawn shape variations. Moreover, for augmenting the presence of highlighted objects in the animated scene, the highlight animation feature is indispensable.

Hybrid use of 3D and hand-drawn characters and scenes has achieved great success in the cel animation industry. With hybrids, you would usually add cartoon shade to 3D objects to fit into the traditional cel-animated scene. As shown in Figure 2a, with a character made of a 3D object, you could photorealistically render the highlight using a standard local shading model.[1,2] Introducing a threshold of specular intensity in the standard model gives the highlight a simpler configuration, such as Figure 2b. However, neither of these highlights is suited for cel animation (Figure 2a is too realistic, and Figure 2b is too simple). As illustrated in Figure 1, we want more cartoon-like highlights.

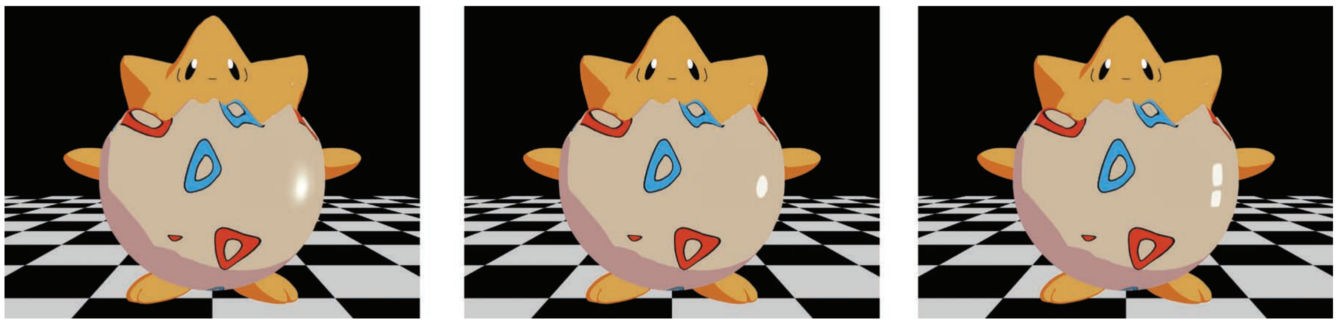
Therefore, in making cartoon-style highlights for 3D objects, we must simultaneously meet the following practical requirements:

- *Shape.* We should create a simply shaped highlight with a clear boundary. It won't always have a rounded shape, but can have rich variations such as crescents and squares.
- *Animation.* We should make smooth and dynamic highlight animation. We therefore need to describe



1 **Highlight effect in cel animation: Various highlights suggest different artistic meanings of objects in the scene.**

Published by the IEEE Computer Society 0272-1716/03/$17.00 © 2003 IEEE


(a) (b) (c)

2 **Highlight by shader: (a) standard 3D, (b) cartoon, and (c) our shader.**

temporary deformation of the highlights stylistically, rather than photorealistically.

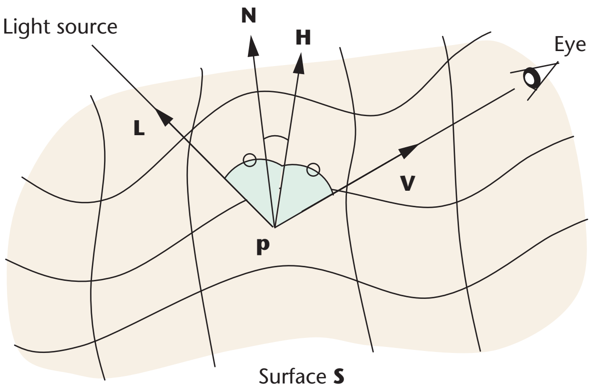
We also make the highlight animation as a keyframe animation. In practice, a keyframing technique is indispensable because it allows fine-tuning of stylistic animations. Besides, traditional cel animation is basically keyframe animation, which is preferable even when using 3D objects in the cel-animated scene.

A standard local shading model—restricted to rounded highlights such as in Figure 2b—could provide good highlight animation using a conventional keyframing technique. However, the standard shading model can't control the highlight shape. In addition, as mentioned previously, the concept of a highlight in cel animation includes uniform environment reflection (such as in Figure 1c). This suggests that we must make various highlight shapes beyond rounded ones. On the other hand, some conventional texture-mapping based approaches might satisfy both of these requirements. For example, environment mapping plus procedural shape animation techniques could work well, but we would need additional methods to make the procedural method suited for keyframing.

We propose a new highlight shader for the 3D objects used in cel animation. Without using a texture-mapping technique, our shader makes highlight shapes and animations in a cartoon style. Our shader makes an initial highlight shape using the traditional Blinn's specular model. [2] (Figure 2b is an initial shape example.) Then it interactively modifies the initial shape through geometric, stylistic, and Boolean transformations for the highlight until we get our final desired shape (such as Figure 2c). Moreover, once these operations specify highlight shapes for each keyframe, our shader automatically generates the highlight animation. In other words, our shader offers a new definition of highlighting 3D objects for cel animation.

Main ideas

Blinn's specular model is originally defined as $\Phi_s(\mathbf{p}) := k_s(\mathbf{N}, \mathbf{H})^n$ at point $\mathbf{p}$ on a surface $\mathbf{S}$, where $\mathbf{N}$ is the surface normal at $\mathbf{p}$, $\mathbf{H}$ is the halfway vector of the light vector $\mathbf{L}$ and the vector $\mathbf{V}$ pointing to the eye in Figure 3. We also assume $\mathbf{N}$, $\mathbf{L}$, and $\mathbf{H}$ are unit vectors. $\Phi_s(\mathbf{p})$ includes the following parameters: the specular reflection coefficient k_s and the exponent n controlling the highlight's sharpness. Our idea is to make the halfway vector $\mathbf{H}$ more flexible for creating a variety of highlights in 3D object cel animation. This means that we redefine the halfway vector. Of course, once $\mathbf{H}$ is specified, it follows that $\mathbf{L} = 2(\mathbf{H}, \mathbf{V})\,\mathbf{H} - \mathbf{V}$. Therefore, our approach can control a virtual light source for a highlight if $\mathbf{V}$ is specified in advance.



3 **Blinn's highlight model.**

We use Blinn's model for making cel animation highlights, assuming that $k_s = 1$ and $n = 1$. Then let us define the highlight (area) $\mathbf{H}_\varepsilon$ as follows:

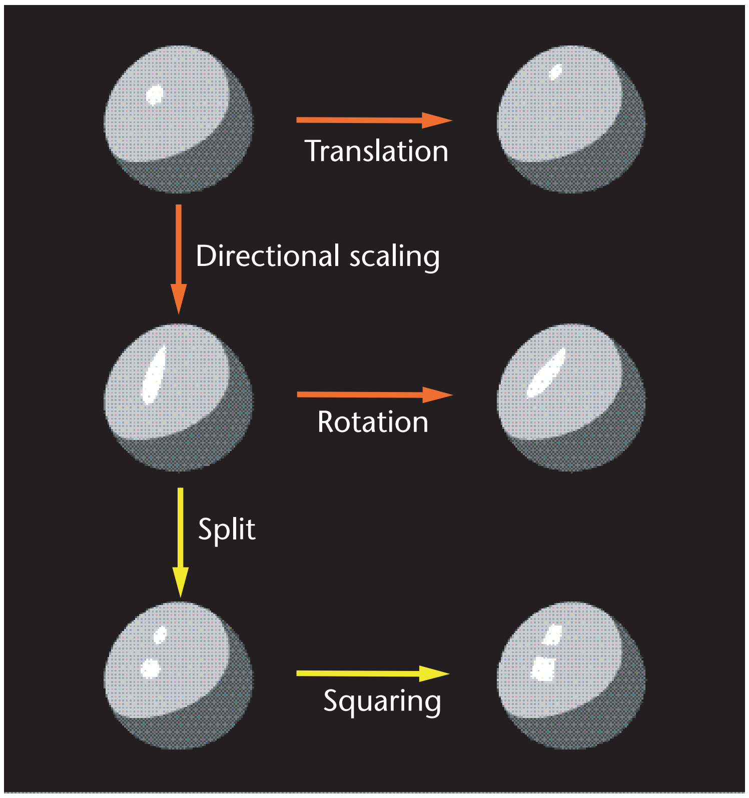
$$\mathbf{H}_\varepsilon := \{\mathbf{p} \in \mathbf{S} \mid \Phi_s(\mathbf{p}) > 1 - \varepsilon\} \qquad (1)$$

where ε is supposed to be a small positive number ($0 < \varepsilon << 1.0$). In the shading process, we give $\mathbf{H}_\varepsilon$ a uniform highlight color (usually white). In many practical cases we move the highlight area on an object for better artistic effect, even if it's physically incorrect. Similarly, we might also rotate the highlight area or make it larger in a certain direction. Along with these geometric operations, we need to develop more operations for obtaining rich highlight variations.

We thus construct a new framework for 3D model highlights in cel animation by first defining a highlight function as

$$\Phi(\mathbf{p}, \mathbf{N}, \mathbf{H}) := (\mathbf{N}(\mathbf{p}), \mathbf{H}(\mathbf{p})) \qquad (2)$$

where $\mathbf{N}$ ($=\mathbf{N}(\mathbf{p})$) means a unit surface normal vector at $\mathbf{p}$ of $\mathbf{S}$, and $\mathbf{H}$ ($=\mathbf{H}(\mathbf{p})$) is a unit vector defined at $\mathbf{p}$ of $\mathbf{S}$.



4 **Basic operations. Top left shows an initial state. Orange arrows show local affine transforms and yellow arrows indicate stylistic transforms.**

Next, we suppose that $\mathbf{H}(\mathbf{p})$ is specified for each $\mathbf{p}$ of $\mathbf{S}$, that is, $\{\mathbf{H}(\mathbf{p})\}_{\mathbf{p}\in\mathbf{S}}$ is given as a vector field on $\mathbf{S}$. Similar to Equation 1 for Blinn's model, we then define the highlight area associated with Φ in Equation 2:

$$\mathbf{H}^*[\varepsilon] := \{\mathbf{p} \in \mathbf{S} \mid \Phi(\mathbf{p}, \mathbf{N}, \mathbf{H}) > 1 - \varepsilon\} \tag{3}$$

for a positive number ε ($0 < \varepsilon << 1.0$). If we take $\mathbf{H}$ as the halfway vector field—that is, $\mathbf{H} = (\mathbf{L}+\mathbf{V}) / ||\mathbf{L}+\mathbf{V}||$ as shown in Figure 3—we then have $\Phi = \Phi_s$ and $\mathbf{H}^*[\varepsilon] = \mathbf{H}_\varepsilon$. We then define a generalized highlight by the highlight function $\Phi(\mathbf{p}, \mathbf{N}, \mathbf{H})$ and its associated vector field $\{\mathbf{H}(\mathbf{p})\}_{\mathbf{p}\in\mathbf{S}}$ for a surface $\mathbf{S}$. Formally, we could arbitrarily specify the vector field. However, in this article, we only treat the vector field $\{\mathbf{H}(\mathbf{p})\}_{\mathbf{p}\in\mathbf{S}}$, which we obtain from the halfway vector field through several operations. Though we originally perform these operations on the vector field, they induce a variety of highlight shapes. From a user's point of view, these operations can be understood as intuitive and easy-to-use modifications on the highlight itself. The operations transform and deform the highlight area, such as in translation, scaling, or splitting. Then, our approach introduces Boolean operations, such as sum and subtraction. Likewise, we repeatedly and interactively modify the highlight area through our shader's GUI, until we achieve our desired cartoon-style shape and animation.

In this framework, we refer to $\{\mathbf{H}(\mathbf{p})\}_{\mathbf{p}\in\mathbf{S}}$ as a highlight vector field on $\mathbf{S}$; then $\mathbf{H}(\mathbf{p})$ as a highlight vector at $\mathbf{p}$. In particular, for the case where $\Phi = \Phi_s$ and $\mathbf{H}^*[\varepsilon] = \mathbf{H}_\varepsilon$, we refer to the halfway vector as the original highlight vector, and $\mathbf{H}_\varepsilon$ as the original highlight. Thus, efficient highlighting depends on how we can construct a good class of the highlight vector field.

Highlight vector field for cel animation

We describe various operations defined for the highlight vector fields on a surface $\mathbf{S}$ that we will highlight. We originally define each of these operations as a local operation, because we perform each one for a single patch. However, we can practically create a larger highlight that traverses over neighboring patches using these operations. In this section, we suppose that $\mathbf{S}$ is coordinated with a single patch (u, v). Then direct operations for a highlight vector, which we call basic operations, are introduced (see Figure 4 and the movie file, highlight1.avi, at http://www.computer.org/cga/cg2003/g4toc.htm). The operations performed directly for the highlight areas are called post operations. You can animate the generalized highlight using these operations.

Local affine transform on a highlight vector field

A few geometric operations on the highlight vector field can transform the highlight in Equation 3. These basic operations are translation, rotation, and directional scaling, which we call local affine transform.

Translation. First we translate the highlight area in Equation 3 to a slightly different position by modifying a highlight vector on $\mathbf{S}$. The modification means translation of a highlight vector in the following way. As noted earlier, translating a highlight vector means moving the virtual light source for the highlight area.

Let's consider the tangent plane at $\mathbf{p}$, which is spanned by the linearly independent unit vectors $\mathbf{du}$ and $\mathbf{dv}$. For a highlight vector $\mathbf{H}$ at $\mathbf{p}$ and, given real numbers α and β, we define translation $t(\mathbf{H})$ of $\mathbf{H}$ as $\mathbf{H}' := \mathbf{H} + \alpha\mathbf{du} + \beta\mathbf{dv}$, $t(\mathbf{H}) := \mathbf{H}' / ||\mathbf{H}'||$. Then t translates all the highlight vectors on $\mathbf{S}$, so that the new highlight vector field $\{t(\mathbf{H}(\mathbf{p}))\}$ can translate the highlight area in the specified direction (α, β). The top row in Figure 4 shows a translation example.

The two numbers α and β in t specify the direction in which we can move the highlight area. However, they don't prescribe the distance between the new and old highlight areas. Formally, we could define the translation operation t for arbitrarily large numbers α and β, but in practice they should be relatively small because, for large α and β (that is, $|\alpha| + |\beta| \to \infty$), $t(\mathbf{H})$ is almost parallel to the tangent plane and has no meaning.

Rotation. If we change the (u, v) coordinate system of the tangent plane by rotation, then we can also rotate $\mathbf{H}$. The rotation $r(\mathbf{H})$ for a highlight vector $\mathbf{H}$ simply means rotating $\mathbf{H}$, which is induced by the 2D rotation of the (u, v) coordinate system.

Directional scaling. We could enlarge the highlight area $\mathbf{H}^*[\varepsilon]$ in Equation 3 by taking a larger ε. However, we should endow the highlight area with a directional property, such as anisotropic reflection.

We define directional scaling $s(\mathbf{H})$ for a given number $\delta(0 < \delta \leq 1.0)$ in the **du** direction as

$$\mathbf{H}'' := \mathbf{H} - \delta(\mathbf{H}, \mathbf{du})\mathbf{du},$$
$$s(\mathbf{H}) := \mathbf{H}'' / \|\mathbf{H}''\| \quad (4)$$

We first consider replacing **H** with $\mathbf{H} + \delta(\mathbf{N} - \mathbf{H})$, so that it gets closer to **N** in the sense that the inner product of **N** and $\mathbf{H} + \delta(\mathbf{N} - \mathbf{H})$ approaches 1.0. Similarly, to approximate **N** only in the **du** direction, we replace **H** by

$$\begin{aligned}\mathbf{H}'' &= \mathbf{H} + \delta(\mathbf{N} - \mathbf{H}, \mathbf{du})\mathbf{du}\\ &= \mathbf{H} + \delta(\mathbf{N}, \mathbf{du})\mathbf{du} - \delta(\mathbf{H}, \mathbf{du})\mathbf{du}\\ &= \mathbf{H} - \delta(\mathbf{H}, \mathbf{du})\mathbf{du}\end{aligned}$$

since $(\mathbf{N}, \mathbf{du}) = 0$ at **p**. Therefore, we have Equation 4. Then we see that $\mathbf{H}''$ is almost equal to the original **H**, if **H** is close to **N**. Similarly, if **H** is far from **N**, then $\mathbf{H}''$ is closer to **N**. The highlight area would become larger if $\delta(>0)$ is close to 1. Also $\mathbf{H}^*[\varepsilon]$ includes $\mathbf{H}_\varepsilon$ since $\|\mathbf{H}\|^2 = (\mathbf{H}'', \mathbf{du})^2(\delta - 1)^2 + (1 - (\mathbf{H}, \mathbf{du})^2)$ is always less than 1.0 for $\delta(0 \leq \delta \leq 1.0)$. This means that directional scaling enlarges the original highlight area.

Though directional scaling is performed in the **du** direction, we can also make directional scaling s in an arbitrary direction, by combining rotation r. The middle row of Figure 4 illustrates these two operations.

Split and squaring

The local affine transform lets us endow the highlight area with a more symbolic and artistic meaning, apart from photoreality. In addition, we set up a few stylistic operations on the highlight vector field to get a wider variety of cartoon-style highlights. The additional basic operations are split and squaring.

Split. The split operation is a slight modification of the directional scaling operation. For given non-negative numbers γ_1, γ_2, and a highlight vector **H**, we define split operation $spl(\mathbf{H})$ as

$$\mathbf{H}^{\vee} := \mathbf{H} - \gamma_1 \mathrm{sgn}[(\mathbf{H}, \mathbf{du})]\mathbf{du} - \gamma_2 \mathrm{sgn}[(\mathbf{H}, \mathbf{dv})]\mathbf{dv},$$
$$spl(\mathbf{H}) := \mathbf{H}^{\vee} / \|\mathbf{H}^{\vee}\|$$

where sgn[ ] is the signature function such that $\mathrm{sgn}[x] = 1$ if $x = 0$, and $x / |x|$ otherwise.

Unlike directional scaling $s(\mathbf{H})$, $spl(\mathbf{H})$ can be away from **N** in the sense that $(spl(\mathbf{H}(\mathbf{p})), \mathbf{N}(\mathbf{p})) \leq 1 - \varepsilon$, for $\mathbf{p} \in \mathbf{H}_\varepsilon$. This means that $\mathbf{p} \notin \mathbf{H}^*[\varepsilon]$. This could happen around the center of $\mathbf{H}_\varepsilon$, where $\mathbf{H}(\mathbf{p})$ is almost equal to $\mathbf{N}(\mathbf{p})$. In Figure 4 the left column shows a split operation example, where the split operation divides the highlight into the two parts along the **du** axis, by setting $\gamma_1 > 0$ and $\gamma_2 = 0$. In the movie highlight1.avi, we divide a highlight into four smaller highlights by split operation, with both γ_1 and γ_2 being positive.

Squaring. The squaring operation, which we denote by $sqr(\mathbf{H})$, makes a highlight area more square shaped. We define this operation for a highlight vector **H** as follows:

The simplicity of our shader's algorithms allows for interactive rate parameter setting in highlight shape design at each keyframe.

$$\theta := \min(\cos^{-1}(\mathbf{H}, \mathbf{du}), \cos^{-1}(\mathbf{H}, \mathbf{dv})),$$
$$\mathrm{sqrnorm} := \sin(2\theta)^n,$$
$$\mathbf{H}^{\wedge} := \mathbf{H} - \sigma \times \mathrm{sqrnorm}((\mathbf{H}, \mathbf{du})\mathbf{du} + (\mathbf{H}, \mathbf{dv})\mathbf{dv}),$$
$$sqr(\mathbf{H}) := \mathbf{H}^{\wedge} / \|\mathbf{H}^{\wedge}\|$$

where integer n and positive number $\sigma(0.0 \leq \sigma \leq 1.0)$ are given. The highlight area would then be square shaped along the **du**- and **dv**-axis. With a rotation operation, we can get the highlight area square shaped in a desired direction. With a larger n, the area becomes more sharpened, whereas σ prescribes the magnitude of the squared area. The bottom row in Figure 4 shows a squaring operation example.

Post operations

By repeated use of the basic operations, we can create various highlight shapes. In some cases, we further need post operations directly performed for highlight areas—rather than for highlight vectors. Here we briefly mention a few typical post operations.

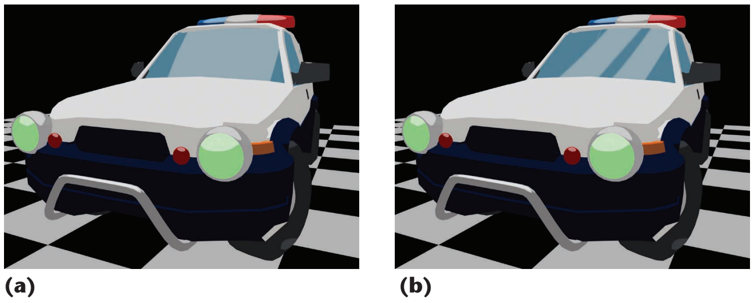
For example, in making a crescent highlight, the Boolean operations such as sum and subtraction among highlight areas would be more useful than the basic operations. To make this possible, we define the Boolean operations for the highlight areas as follows. Let **G** and **H** be highlight areas on **S**. This means that we have the associated highlight vector fields $\mathbf{g}(\mathbf{p})$ and $\mathbf{h}(\mathbf{p})$ such that $\mathbf{G} := \{\mathbf{p} \in \mathbf{S} \mid \Phi(\mathbf{p}, \mathbf{N}, \mathbf{g}) > 1 - \varepsilon\}$ and $\mathbf{H} := \{\mathbf{p} \in \mathbf{S} \mid \Phi(\mathbf{p}, \mathbf{N}, \mathbf{h}) > 1 - \eta\}$, where ε and η are properly specified $(0 < \varepsilon, \eta << 1.0)$.

Both ε and η can be given different values. Moreover, we can easily define the post operations of sum $(\mathbf{G} \cup \mathbf{H})$ or subtraction $(\mathbf{G} \setminus \mathbf{H})$ through simple calculation of the highlight function Φ. This leads us to the wider highlight variations. These operations have intuitive meanings so that they are easy to use in designing a highlight shape. We will demonstrate a practical example created with the post operations in the "Results and discussion" section.

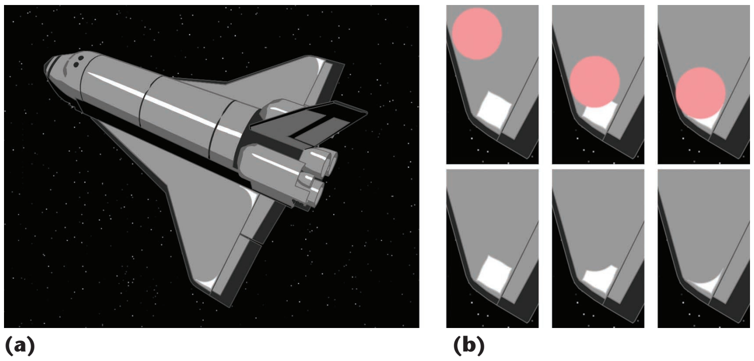
If we find an unnecessary part of a highlight, we should eliminate it with a post operation. For this purpose, we've added the off-specular operation to our shader. We also provide a few additional post operations for practical use. In this article, however, we don't explain these auxiliary operations to keep the focus on the main features of our shader.

Making highlight animation

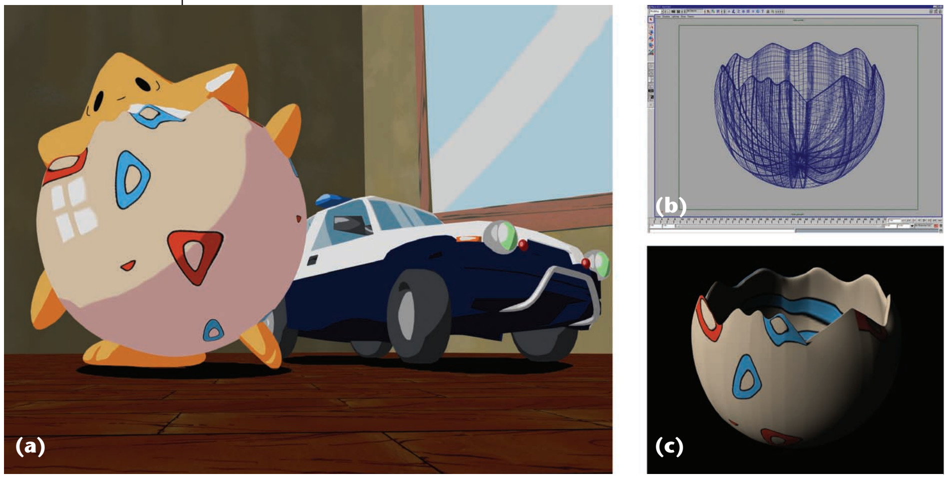
All the operations explained in this article are implemented and integrated in our highlight shader. To ani-



5 Typical highlight example using (a) conventional cartoon shader and (b) our shader.



6 Stylized highlight with post operations: (a) spaceship and (b) Boolean operations.



7 3D models used in the first animation example: (a) 3D character and scene, (b) geometry of the 3D character, and (c) texture on the character.

mate a generalized highlight, we employ the standard keyframing technique, which linearly interpolates the parameter values of the operations in our shader. Then keyframing the parameters occurs only for the patches on which the original highlight should be modified. The simplicity of our shader's algorithms allows for interactive rate parameter setting in highlight shape design at each keyframe.

Except for camera or light control for the whole scene we want to animate, we adopt the linear interpolation technique for highlight animation, because of its easy implementation and fast processing. We might explore alternative interpolation methods; however, our approach works well with the simple interpolation technique.

Results and discussion

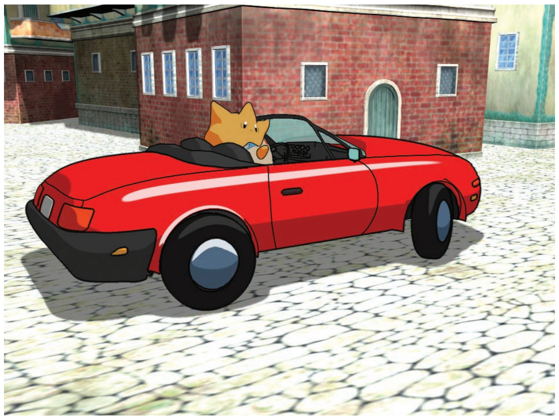
The prototype system of our highlight shader is currently a series of plug-ins for Alias|Wavefront's Maya on a 600-MHz Pentium III PC with 1,024 Mbytes of main memory. The prototype system lets us make cartoon-shaded animation using 3D models along with stylized highlights.

Figure 5 shows a typical example made with our shader. The car model in this example consists of about 30 patches. We used a standard shader to make the car image in Figure 5a, where the highlight area dominates the front window. No matter how we chose ε for Hε in Equation 1, the highlight area would cover almost all the front window, since it was almost flat. On the other hand, as shown in Figure 5b, we successfully rendered the car with our shader using rotation, directional scaling, and split. The highlight area indicates the window's material in a much more impressive way than in Figure 5a. Figure 6 illustrates how we can apply post operations for making the highlight variations. We applied a few Boolean operations to make the highlights on the spaceship in Figure 6a. Figure 6b demonstrates how we applied the sum and subtraction operations to get the highlights on the spaceship wing.

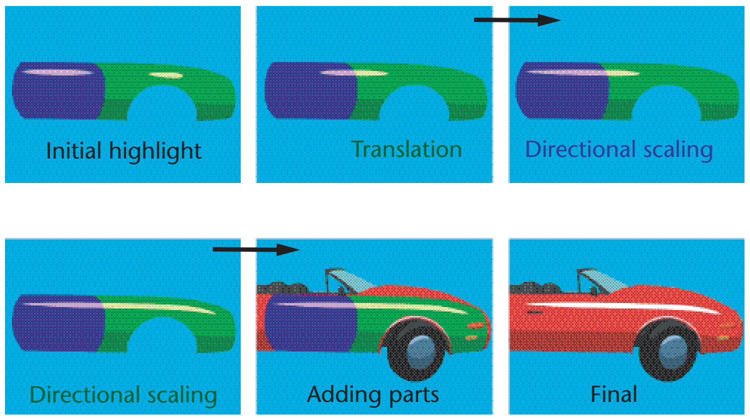
In the following experimental results and keyframe animations created with our shader we typically design the highlight at a keyframe after fixing the camera path for the animation.

In the movie Experimental Animation 1 (see the movie file highlight2.avi at http://www.computer.org/cga/cg2003/g4toc.htm), the car in Figure 5 rotates with a fixed directional light source. The movie demonstrates that our shader achieves smooth and natural highlight change. We indicated all of the keyframes in the movie to illustrate how we created this highlight animation with our shader. In Experimental Animation 2 (see the movie file highlight3.avi), the car is at a fixed position, whereas a directional light source is moving around the car. This shows that we successfully created a dynamic change in a cartoon-style highlight.

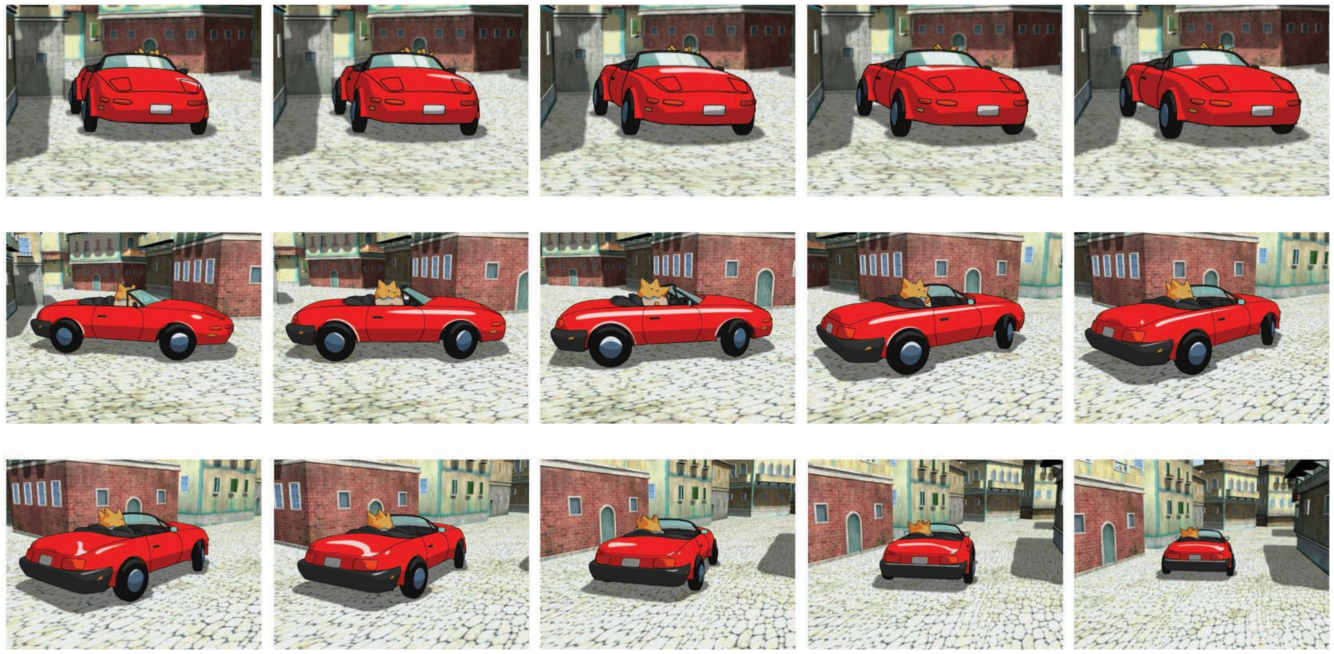
The next two animation examples demonstrate how our shader works in more practical situations. Of course, in some cases, existing approaches would also work well. For example, the flow animation of the highlight stripe on the sword in Figure 1a would be



8 Frame from our final animation example.



9 Making long highlight traversing patches.



10 Dynamic highlighting in our final animation example.

relatively easy to make with the ramp texture technique in Maya.

Figure 7a shows the character and the scene used for the first animation piece (see the movie file highlight4.avi), where we semantically animated all the highlights. For instance, our shader successfully synchronized the highlights on the character and the windows. We made this character from 13 nonuniform rational B-spline patches and rendered it at about 33,000 polygons. Figure 7b presents the wireframe model of this character's shell; Figure 7c shows its 3D-shaded image. The highlights on this shell were smoothly deformed, because of smooth transition among the highlight vector fields through the basic operations.

We have not yet considered a highlight that traverses neighboring patches, since we originally defined operations in our shader for a single patch. However, with our final animation example, we show that we can eliminate this restriction in practice.

In the final example (see the movie file highlight5.avi), we animated several highlights on a fast-moving car. Figure 8 shows a frame taken from this animation, where we see a long highlight on the side body of the car. The long highlight actually traversed different patches of the side body. Figure 9 explains how to make a keyframe for the long highlight. First, we put the initial highlight on each patch (this frame in the figure shows where the patches are distinguished to each other by color). We made each highlight using the original highlight vector field. Using directional scaling and translation operations repeatedly, we successfully made the final long highlight. As Figure 10 demonstrates, the movie clearly describes the dynamic change of highlight. The resultant highlight shapes were so various that some were rather complicated according to the car model's surface geometry. This demonstrates our approach's advantage: we can naturally coordinate generated highlights with the surface parameters. We only used 15 keyframes to make the movie, whereas we rendered

Related work

Early milestones of the hybrid use of 3D and hand-drawn characters include some of Disney's animated movies, such as *The Great Mouse Detective* (1986) or *Beauty and the Beast* (1991). Following these, a variety of cel-animated films have been made possible by the rapid progress of 3D computer graphics technology (see Anjyo et al. for instance[1]).

One of the recent successes of 3D computer graphics techniques applied to the cel animation industry is cartoon shaders. The shaders render 3D characters and objects in a cartoon style. For example, Softimage's Toon Shaders[2] contributed to famous films such as *The Princess Mononoke* (1997) by Studio Ghibli and *The Prince of Egypt* (1999) and *The Road to El Dorado* (2000) by Universal Studios/DreamWorks. Another successful method is applying complex textures on 3D objects to hand-drawn characters in cel animation.[3] With this method, animators can combine complex textures and hand-drawn artwork. The view-dependent geometry also provides a useful 3D model that successfully inherits 2D artistic expressions by allowing view-dependent distortion.[4] A semiautomatic method for generating shadow mattes uses 3D inflation models for hand-drawn characters.[5] This method avoids the hand-drawn task in shadow matting, while enriching complex shadow variations. Unfortunately these techniques don't address the practical requirements for highlighting.

Lake et al. propose an efficient interactive rendering system with the several real-time algorithms that emulate various cartoon styles.[6] These real-time methods include a technique that can deal with cartoon highlighting. Since this technique is essentially based on a local shading model, it's hard to control highlight shape. Other conventional approaches to making highlights would be of use with projected textures, light maps, or virtual light sources. In a projected texture approach, we consider highlight as texture, and can easily make its static variations. In making highlight animation, we would make 2D animation in the texture parameter space using a procedural technique. Similarly, a light map approach would be effective for highlight shape, but again this would require additional (mostly procedural) operations in the light map space for highlight animation. Keyframing might cause some difficulties for both approaches. Alternatively, use of virtual light sources for highlighting means corresponding a virtual light source to each highlight area. This method could work well, supposing that the highlight shape is relatively simple. If the highlight shape becomes complex, then we must complete the number of virtual light sources, and they might become hard to control in making the highlight animation.

Our approach starts with generalizing Blinn's specular model, so that the generalized model lets us easily make various stylized highlights. We make the generalized highlight directly on a surface in the sense that we perform the highlight transformations mentioned in the text introduction in the surface's parametric space. This distinguishes our approach from existing mapping-based ones, such as projected textures or light maps. We achieve the highlight transformations by intuitive, interactive, and direct manipulation on the highlight area to get a desired highlight shape. Moreover, we obtain the generalized highlight animation as keyframe animation by interpolating the parameters that prescribe those transformations. This ensures the greater flexibility of our approach over previous works.

References

1. K. Anjyo et al., "Digital Cel Animation in Japan," *Siggraph 2000 Conf. Abstracts and Applications*, ACM Press, 2000, pp.115-117.
2. M. Arias, "Non-Photorealistic Rendering," *Softimage|XSI Power Creators' Guide*, Aspects Corp., 2002, pp. 284-290.
3. W. Corrêa et al., "Texture Mapping for Cel Animation," *Proc. Siggraph 98*, ACM Press, 1998, pp. 435-446.
4. P. Rademacher, "View-Dependent Geometry," *Proc. Siggraph 99*, ACM Press, 1999, pp. 439-446.
5. L. Petrovic et al., "Shadows for Cel Animation," *Proc. Siggraph 2000*, ACM Press, 2000, pp. 511-516.
6. A. Lake et al., "Stylized Rendering Techniques For Scalable Real-Time 3D Animation," *Proc. 1st Int'l Symp. Non-Photorealistic Rendering* (NPAR 00), ACM Press, 2000, pp. 13-20.

180 frames in total. These results suggest that our approach can create various cartoon-look highlights for 3D animation using few keyframes.

Future work

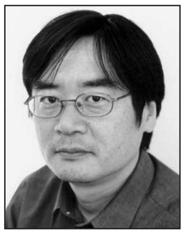
We have demonstrated an approach for rendering and animating stylized highlights on 3D objects used in cel animation. This approach starts with the initial highlight design, based on the halfway vectors in Blinn's specular model. Then various highlight shape and animation are created through several functional operations for the highlight vector field. We are currently making our approach usable in real time, by exploiting the power of graphics hardware. We are also generalizing the vector-field-based approach, so that we can arbitrarily specify the initial highlight shape independently of the halfway vector field. We believe that the generalized framework should lead us to a unified approach for efficient cartoon shading and shadowing. ■

Acknowledgments

We thank the Pikachu Project for providing the character models used in this article, and Katsumi Takao and Shinji Morohashi for their help in creating the images in Figure 1. We also thank Yoshinori Dobashi of Hokkaido University for discussions during the early development of this work, and the anonymous reviewers for their significant help and many constructive comments. Finally, we are very grateful to Toshiaki Okuno for his continuing support and encouragement, which made this work all possible.

References

1. B.-T. Phong, "Illumination for Computer Generated Pictures," *Comm. ACM*, vol. 18, no. 6, 1975, pp. 311-317.
2. J.F. Blinn, "Models of Light Reflection for Computer Synthesized Pictures," *Computer Graphics*, vol. 11, no. 2, 1977, pp. 192-198.



Ken-ichi Anjyo *is technical director of the R&D group of OLM Digital. He is also a guest professor at the Tokyo University of Technology. His research interests include exploring techniques that might someday clarify the principles in our visual understanding of human behaviors, natural phenomena, and nonphotorealistic imagery. Anjyo received a BS in mathematics from Saitama University, an MS in mathematics from Kyushu University, and a PhD in information engineering from Nagoya University.*

Katsuaki Hiramitsu *was a software engineer of the R&D group of OLM Digital. He is now at OptGraph, Tokyo, Japan. His research interests include development of the nonphotorealistic rendering tools used in production work, such as those used for the Pokémon movies, and developing rendering algorithms. Hiramitsu received a BS in mathematics from Meiji University, Tokyo.*

Readers may contact Ken-ichi Anjyo at OLM Digital, 1-8-8 Wakabayashi Setagaya, Tokyo, 154-0023, Japan; anjyo@olm.co.jp.