EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2008)
C. Alvarado and M.- P. Cani (Editors)

Sketch and Paint-based Interface for Highlight Modeling

Romain Pacanowski[1,2] Xavier Granier[1] Christophe Schlick[1] Pierre Poulin[2]

[1]INRIA Bordeaux University† [2] Dép. I.R.O., Université de Montréal ‡

Abstract

In computer graphics, highlights capture much of the appearance of light reflection off a surface. They are generally limited to pre-defined models (e.g., Phong, Blinn) or to measured data. In this paper, we introduce new tools and a corresponding highlight model to provide computer graphics artists a more expressive approach to design highlights. For each defined light key-direction, the artist simply sketches and paints the main highlight features (shape, intensity, and color) on a plane oriented perpendicularly to the reflected direction. For other light-and-view configurations, our system smoothly blends the different user-defined highlights. Based on GPU abilities, our solution allows real-time editing and feedback. We illustrate our approach with a wide range of highlights, with complex shapes and varying colors. This solution also demonstrates the simplicity of introduced tools.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation

1. Motivation

3D modeling is generally decomposed in three components: geometry, animation, and appearance (i.e., lighting behavior). Over the years, and with an increasing interest since the publication of Teddy [IMT99], simple sketch-based interfaces have been proposed for geometry modeling and more recently, for animation (e.g., [TBvdP04]). This research in intuitive modeling has only recently emerged for the design of lighting behavior (e.g., [OMSI07, TABI07]), but is still in its early stages.

Over all possible lighting effects, some of the most noticeable are highlights: they play an important role in the final appearance of an object, providing the user with convincing materials. Unfortunately, a highlight is a complex 4D function that depends on both light and view directions. Therefore, in a creative and artistic context, edition and creation of plausible highlights are still challenging tasks. Currently, most systems rely on the selection over pre-defined shading models (such as Phong shading) and the modification of their intrinsic parameters: the user's choice and freedom is thus limited. Recent approaches extend the modeling freedom by using a painting approach: a user creates and edits highlights by painting (cf. Figure1a) the expected result on a sphere [CPK06]. This approach is still limited by the underlying analytical models.

(a) (b) (c)

Figure 1: *Three different styles of highlight obtained with different systems. (a) BRDF-Shop Physic based highlight, (b) Anjyo [AWB06] cartoon based highlight, and (c) example of highlight obtained with our system.*

In this paper, we present an intuitive and flexible system for highlight design. Our system allows the user to specify

† { pacanows | granier | schlick }@labri.fr
‡ poulin@iro.umontreal.ca

interactively through sketching, painting, and manipulation of vectorial data, the highlight's shape and its color gradient. With our new modeling tools, the user can create a wide variety of appearances. Thanks to the simple underlying model, our approach can provide real-time feedback and interactive rendering. These results are due to our two main contributions:

- we provide new *sketching and painting tools* associated with vectorial gradient edition in order to define easily different highlight characteristics, such as their shape and color variations.
- we present a new *highlight model* based on a curve as a global shape representation for a highlight, and a texture for its color and refined shape description.

After reviewing previous work with a focus on interactive shading edition, we present the main principles of our approach and its corresponding modeling tools. We also introduce the underlying highlight models. We then illustrate their user controls and the resulting appearance, and discuss the limitations of the current implementation.

2. Previous Work

In computer graphics, one of the common ways to represent surface appearance and thus highlights, is to use a Bidirectional Reflectance Distribution Function (BRDF). However, its specification relies on a non-intuitive choice by an artist of a pre-defined model (*e.g.*, [Pho75, Bli77, War92, Sch94]), and an adjustment of its parameters that can have a non-uniform perceptual behavior. For an improved selection of the expected appearance, Ngan *et al.* [NDM06] introduced a perceptually uniform navigation in a space defined by the different models and their parameters. On the edition side and once a given model has been selected, Ben-Artzi *et al.* [BAOR06] proposed to project it into a given basis and to factorize it in a set of 1D curves that can be edited directly, even under complex illumination. However, even if these solutions provide users with improved selection and edition tools of predefined highlights, they do not allow the creation of more freely designed ones.

To increase modeling freedom, Colbert *et al.* [CPK06] develop Poulin and Fournier's work [PF95] by proposing a painting interface. In their tool, called *BRDF-Shop*, different "painting" operations are introduced for the design of highlights. The resulting highlight is approximated by a sum of Ward lobes [War92], but other lobe models can be used for this fitting process [NDM05]. Their approach does not guarantee that the painted highlights correspond to a reasonable fit because an arbitrary large number of lobes can be required. An increasing number of lobes will thus result in reduced interactivity. In constrat, Edwards *et al.* [EBJ*06], extending the work of Ashikhmin *et al.* [APS00], create a unique lobe by designing a probability distribution function of normals. A similar indirect control has been proposed also

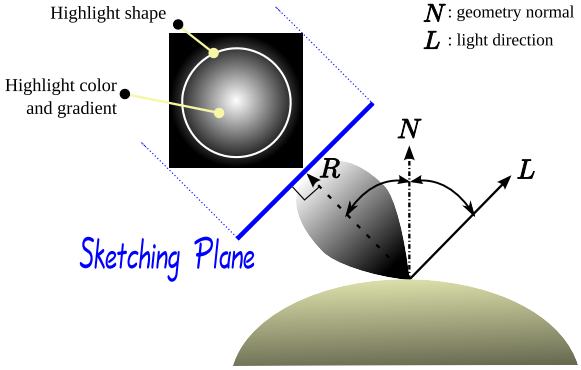


Figure 2: ***The Sketching Plane****. The user sketches the highlight shape and paints the highlight color on a Sketching Plane oriented perpendicularly to the light mirror direction (R).*

by Neumann *et al.* [NNSK99], where the shape of a lobe is defined on the tangent plane of the surface. Unfortunately, these indirect controls can be non-intuitive.

By removing constraints on realism, more freedom is provided to users in highlight design in a non-photorealistic context. One of the first solutions, based also on a painting metaphor, is the *Lit Sphere* [SMGG01] by Sloan *et al.*. In their approach, the painted appearance on a sphere, for given viewpoint and light direction, is used as a texture projected on 3D surface, taking into account the similarity of the configuration between the viewpoint and the normal. Therefore their approach is only possible for a fixed lighting direction. Recently, Okabe *et al.* [OMSI07] directly painted the lighting on a 3D surface. From this input, a 3D environment map is constructed and used as light source, but no control on highlights can be provided. Anjyo *et al.* [AWB06] tweaked the shape of a highlight in cartoon-like (cf. Figure 1b) shading. However they do not really edit the BRDF: they locally move, scale, split, and merge the light sources in order to obtain the requested shape which is the only editable parameter. Smoothness, glossiness, and color variations are not taken into account. Similarly, Todo *et al.* [TABI07] remove or add some highlights in toon shading, using offset texture defined for light key-directions. Unfortunately, this solution is limited to stylized shading.

Of all the previous solutions, the most closely related to sketching is the work of Pellacini and Lawrence [PL07]: They use strokes to select different areas of a BTF to interactively modify materials. In our approach, we also want to provide users with a fully interactive edition, but for the design of highlights. For user friendliness, our solution is based on sketching and painting metaphors. For efficiency and interactivity reasons, we directly manipulate the highlight shape and colors.

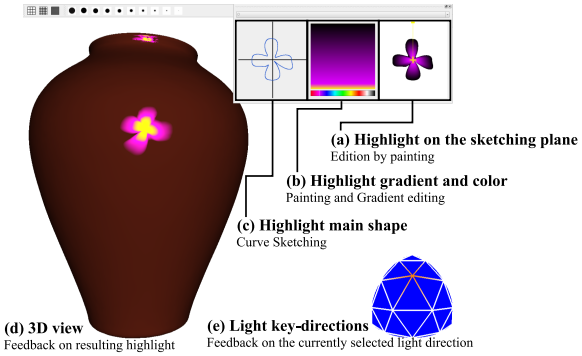


Figure 3: *User Interface Main View.*

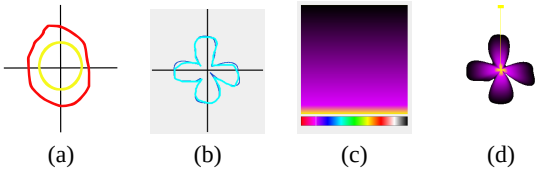


Figure 4: *(a) The external curve represents the user's selection sketch.(b) The cyan curve represents the user's sketch whereas the blue one represents the fitting result. (c) A vectorial color gradient defined by the user. (d) Sketching Plane view that displays the resulting highlight when applying (b) and (c).*

Overview. As illustrated in Figure 2, our approach is based on the direct edition of highlight features, displayed on a plane oriented perpendicularly to the light mirror direction (R). On this plane, the artist changes the highlight shape and color-gradient with 2D editing tools. Consequently, the highlight features are defined for a given light **key-direction** (L), similar to the solution of Todo *et al.* [TABI07]. The set of highlight features associated with a light key-direction is called a **lighting configuration**. A typical workflow in our system is firstly to define the highlight characteristics with our 2D tools (first shape, then color and gradient, as shown in Figure 4), and secondly to observe the highlight behavior for different light or view directions. By default, our system replicates the same highlight features for all lighting configurations. However, the user can edit them for a defined light direction. For undefined ones, our system smoothly interpolates every highlight features to ensure a coherent highlight behavior. Furthermore, at any time our system provides real-time feedback to user's editing actions.

3. Sketch-based Interface for Highlight

To provide intuitive tools for highlight modeling, we rely on three different interaction approaches adapted for each modeling action. For simplicity of use, each interaction is associated with one specific screen area (cf. Figure 3). As illustrated in Figure 4, user can sketch the shape, paint the highlight color or edit its gradient with vectorial tools.

Sketching has been recognized as an efficient tool to define a global shape. Therefore, we rely on this approach to modify the highlight shape. Through sketched strokes in a specialized area (cf. Figure 3(c)), the user selects part of or the whole highlight shape (cf. (Figure 4(a)). Once selected, a new target curve is drawn. A fitting process then approximates the sketch (cf. Figure 4(b)) using the underlying representation of curves (defined in Section 4). This sketch-based definition of the shape also controls the highlight shininess: the smaller the shape is, the shinier the highlight is. To increase the shininess, users have simply to scale down the curve.

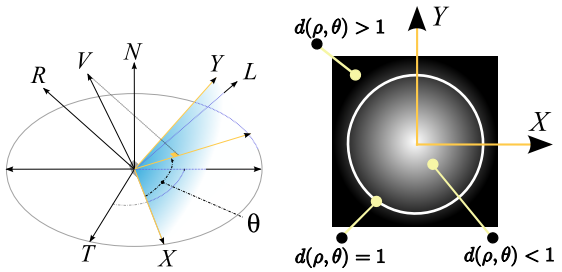
To define color variations and gradients, painting is the most well-suited approach. The user thus uses a set of brush and gradient tools to refine the highlight shape and to control its color behavior. Brushes allow the user to edit the highlight color while the filters let her adjust the highlight intensity. This is also done in specialized areas, one for the gradient manipulation (cf. Figure 3(b) and 4(c)), one for a precise edition on the final results (cf. Figure 3(a) and 4(d)). The combination of the shape and the gradient texture is explained in Section 4. If the embedded tools are not sufficient, an experimented artist can load an image created with any other image manipulation software.

For an interactive modeling system, real-time feedback is crucial. Thanks to our representation based on a unique lobe defined with a simple curve and a texture (see Section 4); each modification is directly viewed on a selected 3D model (cf. Figure 3(d)). Since the highlight depends both on the light and on the view directions we provide also the user a visual representation of the current light key-direction (cf. Figure 3(e)). Furthermore, the user can select a part of the 3D object to retrieve local parameters, such as the most important light configuration or the highlight color and shape. Although this is not a 3D painting interface, it greatly facilitates the user's work when it comes to modifying precisely parts of the highlight.

4. Highlight Representation

For each lighting configuration, a distance field defined by a curve represents the highlight shape and a texture represents the highlight colors and intensity.

Highlight Shape. We define the highlight shape as a curve, the size of which controls the highlight **shininess**. The key idea here is to use this curve as the outline for the whole highlight. Outside of this curve, the highlight intensity is null, whereas inside the intensity is modulated by a color texture. In order to fill the shape, our representation should



(a) (b)

Figure 5: *(a) The X and Y directions constitute the Sketching Plane local frame. They are computed according to the light mirror direction R and the geometry local tangent T. To compute the highlight's intensity, we project the view direction V on the Sketching Plane. (b) The distance field* $d(\rho,\theta)$ *defined by the polar curve parameterizes the Sketching Plane.*

allow a simple and intuitive access to the color texture. Furthermore, we also need a representation that can be easily interpolated between light key-directions. This is required for the reconstruction of highlight shape for unspecified light directions.

Therefore, we use the spline-based polar curve proposed by Crespin *et al.* [CBS96]. Each control point for the curve is located by an angle θ and is defined by its radius ρ, and its left and right tangents. These parameters (radius and tangents) are easily interpolated. Furthermore, this curve representation enables the definition of a distance field $d(\rho,\theta)$ that starts from the center of the Sketching Plane (cf. Figure 5(b)). This distance field is then used to fill the shape with a color texture.

Texture Definition. The color texture represents the highlight color and intensity inside the shape. Therefore, black texels indicate areas where the highlight intensity is null. This allows a refinement of the curve-defined shape. Furthermore, by controlling the intensity gradient, we control the highlight **glossiness**.

Two parameterizations (polar or Cartesian) are used for texture lookup. We found it easier to work with polar coordinates when the texture is used to define a simple color gradient, and Cartesian coordinates for more generic textures, such as those created using painting tools. When using the polar parameterization (cf. Figure 4), the horizontal axis of the texture color component represents the angular variation whereas the vertical axis represents the radial one.

Since the color texture is enclosed by the star-shape polar curve one, we can also use it to define more complex shapes. To further ease this process, we separate, as it is done in common painting software, the color texture into two multiplicative layers. One layer stores the highlight color and the other stores the highlight intensity as gray level (cf. Figures 9 and 10).

Parameterization. Finally, we need to define the parameters ρ and θ as functions of the view and light directions:

$$\rho = V \cdot R \qquad \cos\theta = V \cdot X \qquad \sin\theta = V \cdot Y \tag{1}$$

where $\cdot$ denotes the dot product, R the light mirror direction, and V the view direction. The directions X and Y are defined as follows:

$$X = T \times R \qquad Y = R \times X \tag{2}$$

where $\times$ denotes the cross product, and T the local geometry tangent (cf. Figure 5)(a)). With polar coordinates, to access the color texture, the system uses the following (u,v) texture coordinates:

$$u = \frac{\rho}{1 - d(\rho,\theta)} \qquad v = \frac{\theta}{2\pi}. \tag{3}$$

To summarize, for a given light direction, the system smoothly interpolates a distance field $d(\rho,\theta)$ and a color texture that are both used and accessed, at rendering time, according to the current view direction.

Implementation. We only use the CPU to perform the least-square fitting of the polar curve. To achieve interactive feedback, we rely on current graphics hardware capabilities using *OpenGL Shading Language*.

The shape of the highlight is stored as a floating 3D texture. A slice of the 3D texture contains parameters (radius and tangents) of one curve's control point for each lighting configuration. The number of slices is the number of control points of the polar curve. Therefore, a slice parameterizes a spherical triangle representing all possible lighting configurations. We pack every color texture (one for each lighting configuration) into a single texture array (ARB_texture_array extension).

In practice we use between 20 and 60 control points with 3 different lighting configurations, thus requiring less than 40 KB to store the highlight features (shape + color). Therefore, our representation is a very low-consumption memory solution and one may use it to store many different shaders on GPU resident memory.

5. Use-case Scenarios and Results

We perform all the presented experiments on an Intel Core 2 Duo T7500 CPU workstation with a GeForce 8600 M GT. For a rendering resolution of 1280×1024 and an object complexity of 35 000 triangles, we report a frame rate of 60 frames per second. On the interaction side, each action leads to an update of the interface in less than 100 ms. Through all presented results, keep in mind that the final highlight appearance results from the combination of diffuse and highlight colors.

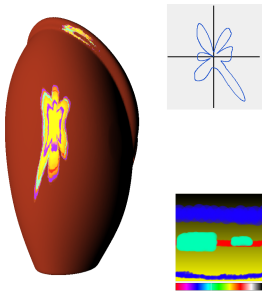


Figure 6: ***A more complex highlight shape.***

Figure 7 shows a scenario where the user sketches a trefoil-like shape with different sizes and vectorial gradients for each light key-direction (cf. Figure 7(d)). For the normal incidence direction, the color texture is a green gradient; at tangent (resp. bitangent) incidence, the color texture is a blue (resp. red) gradient. As demonstrated in Figures 7(b) and (c), when the light shifts toward the tangent (resp. bitangent), the highlight becomes more blue (resp. red). The yellow color in Figure 7(c) comes when interpolating between the red and green colors stored for two lighting key-directions. This explains why the upper part of the sphere, where light configurations for normal and bitangent key-directions have equal importance, is yellower while the lower part of it, where the configuration for bitangent direction is more important, appears redder. The user can sketch more complex shapes, such as the one presented in Figure 6. Contrary to the previous example, painting tools were used to create a more complex color texture. For both previous examples, our system uses polar coordinates for texture look-up.

With a similar approach, the user can create complex appearances, close to realistic ones, such as the dispersion-like effect of Figure 8. This is simply done by creating two different color textures (using Cartesian coordinates) and two different shapes (cf. Figure 8(d)). For the lighting configuration at normal incidence, the texture is a white gradient, where for grazing configurations, it is a rectangular rainbow gradient. Furthermore, a circular shape is used for the normal incident light key-direction and a rectangular shape for the others.

The two layers of the color texture can be used to refine the star shape created with the sketched curve (cf. Figures 9 and 10). We model the highlights of Figure 9 with a smiley bitmap texture (cf. upper right corner of Figure 10). As explained earlier, points of the sketching plane falling outside the bitmap texture are not shaded (i.e., the highlight intensity is null). Since the color texture is enclosed in the polar curve shape, it can be deformed to achieve new effects, like the unhappy smiley of Figure 9(b). Finally, Figure 10 shows an example with more complex color texture. The color layer is filled according to the lighting configuration, whereas the intensity layer, which contains the shape, remains constant (cf. Figure 10(d)).

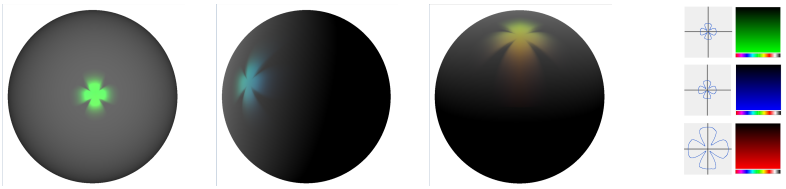
Discussion. Our system and its highlight representation are only a first step and some limitations remain. First, since the painted and sketched highlights are projected from the sketching plane on the surface, which is not necessarily planar, some shape deformation can occur. However, our system allows changes and adjustments of the shape in real time, such that the user may compensate for this deformation. This was done, for example, in Figures 9(a) and (b) to compensate the vase curvature. Even though our curve-based representation ensures a smooth interpolation between different shapes, some interpolation artefacts may occur when using complex color textures (as in Figure 10). This limitation is related to the partially (un)solved general problem on how to morph smoothly between two arbitrarily different images. Third, remember that our system models a highlight expressed as a function of the view and light directions. However, these directions are parameterized according to the local geometry normal. Therefore, a complex surface with many normal variations exhibits many highlights but not necessarily on a large area. This explains why highlights in Figures 8 and 9 on the upper edge of the vase are almost indistinguishable.

Although it might still be complex to design a highlight from 2D sketches and paintings rather than directly paint it on the object, we think that this drawback is compensated by our interactive feedback and our visual hints. Finally, even though we show examples with only three different lighting configurations, one may use more lighting configurations (within hardware limits). However, we found, through empirical testing, that increasing the number of lighting configurations increases the highlight design complexity for the artist.

6. Conclusion and Future Work

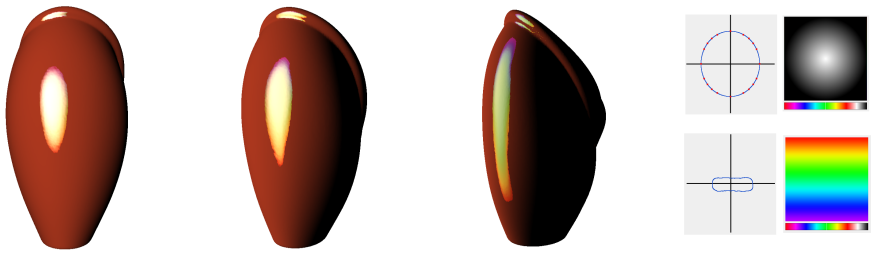
We have presented a new system based on sketching and painting metaphors for highlight design. A highlight is represented with a polar curve for its shape and a texture for its color gradient and for precise shape enhancement. Our simple and compact representation, suitable for hardware rendering, allows real-time edition of highlights, which can be various and reach some realistic aspects. Our tools and representation also provide more editing and creative freedom than previous approaches. We believe that our solution is more intuitive for highlight control in shading design.

Our upcoming work follows two streams. First we would like to extend it as a full 3D painting and sketching interface where the user can draw directly the appearance for simple and complex illuminations. We believe that this would be possible with the same parameterization, but a new representation. Second, we want to develop new tools and a new highlight model with spatially varying properties to help the user control highlights and highly detailed geometry.



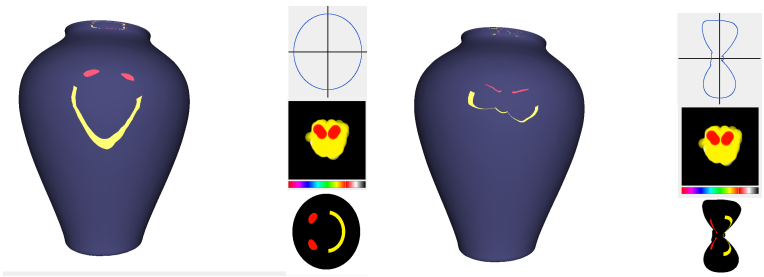
(a) (b) (c) (d)

Figure 7: ***Light key-directions illustration***. *The highlight color and size is set according to the light direction. When the light direction is collinear with the geometry normal (a), the highlight exhibits a green color, whereas when the light shifts toward the tangent (b) (resp. bitangent (c)), the highlight becomes more blue (resp. red). For configurations (b) and (c), the highlight size is larger than for (a) to increase the color-shift effect.*



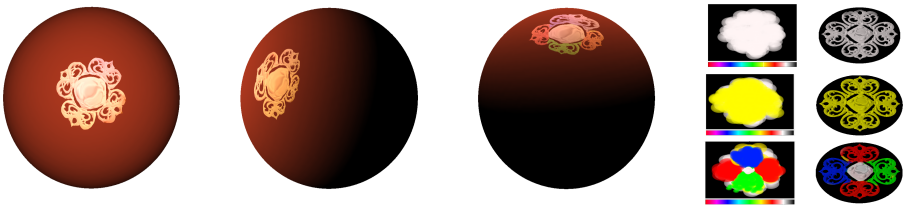
(a) (b) (c) (d)

Figure 8: ***Dispersion example***. *(a)-(c) The highlight exhibits a dispersion behavior according to the light direction. Top (resp. bottom) row of (d) shows the color and shape configuration for the configuration at normal (resp. grazing) incidence..*



(a) (b)

Figure 9: ***Bitmap deformation example***. *(a) The initial highlight shape, generated with a smiley-like bitmap texture, is changed to (b) an unhappy smiley by modifying the highlight curve.*



(a) (b) (c) (d)

Figure 10: *(a)-(c) Highlight rendered under different light directions. (d) The first (resp. second and third) row shows the lighting configuration when the light is collinear with the geometry normal (resp. tangent and bitangent). All configurations use a circular shape.*

References

[APS00] ASHIKHMIN M., PREMOZE S., SHIRLEY P.: A microfacet-based BRDF generator. In *ACM SIGGRAPH '00* (2000), pp. 65–74.

[AWB06] ANJYO K.-I., WEMLER S., BAXTER W.: Tweakable light and shade for cartoon animation. In *Proc. international symposium on Non-Photorealistic Animation and Rendering* (2006), ACM, pp. 133–139.

[BAOR06] BEN-ARTZI A., OVERBECK R., RAMAMOORTHI R.: Real-time BRDF editing in complex lighting. *ACM Trans. Graph. 25*, 3 (2006), 945–954.

[Bli77] BLINN J. F.: Models of light reflection for computer synthesized pictures. In *ACM SIGGRAPH '77* (1977), pp. 192–198.

[CBS96] CRESPIN B., BLANC C., SCHLICK C.: Implicit sweep objects. *Computer Graphics Forum 15*, 3 (1996), 165–174.

[CPK06] COLBERT M., PATTANAIK S., KRIVÁNEK J.: BRDF-Shop: Creating Physically Correct Bidirectional Reflectance Distribution Functions. *IEEE Computer Graphics and Applications 26*, 1 (2006), 30–36.

[EBJ*06] EDWARDS D., BOULOS S., JOHNSON J., SHIRLEY P., ASHIKHMIN M., STARK M., WYMAN C.: The halfway vector disk for BRDF modeling. *ACM Trans. Graph. 25*, 1 (2006), 1–18.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *Proc. ACM SIGGRAPH '99* (1999), pp. 409–416.

[NDM05] NGAN A., DURAND F., MATUSIK W.: Experimental Analysis of BRDF Models. In *Proc. Eurographics Symposium on Rendering* (2005), pp. 117–226.

[NDM06] NGAN A., DURAND F., MATUSIK W.: Image-driven Navigation of Analytical BRDF Models. In *Proc. Eurographics Workshop on Rendering* (June 2006), pp. 399–408.

[NNSK99] NEUMANN L., NEUMANN A., SZIRMAY-KALOS L.: Reflectance Models with Fast Importance Sampling. *Computer Graphics Forum 18*, 4 (1999), 249–265.

[OMSI07] OKABE M., MATSUSHITA Y., SHEN L., IGARASHI T.: Illumination brush: Interactive design of all-frequency lighting. In *Proc. Pacific Conference on Computer Graphics and Applications* (2007), IEEE Computer Society, pp. 171–180.

[PF95] POULIN P., FOURNIER A.: Painting Surface Characteristics. In *Proc. Eurographics Workshop on Rendering* (June 1995), Springer, pp. 160–169.

[Pho75] PHONG B. T.: Illumination for computer generated pictures. *Commun. ACM 18*, 6 (1975), 311–317.

[PL07] PELLACINI F., LAWRENCE J.: AppWand: Editing Measured Materials using Appearance-Driven Optimization. *ACM Trans. Graph. 26*, 3 (2007).

[Sch94] SCHLICK C.: An Inexpensive BRDF Model for Physically-Based Rendering. *Computer Graphics Forum 13*, 3 (1994), 233–246.

[SMGG01] SLOAN P.-P., MARTIN W., GOOCH A., GOOCH B.: The Lit Sphere: A Model for Capturing NPR Shading from Art. In *Proc. Graphics Interface* (2001), pp. 143–150.

[TABI07] TODO H., ANJYO K.-I., BAXTER W., IGARASHI T.: Locally controllable stylized shading. *ACM Trans. Graph. 26*, 3 (2007), 17.

[TBvdP04] THORNE M., BURKE D., VAN DE PANNE M.: Motion doodles: an interface for sketching character motion. *ACM Trans. Graph. 23*, 3 (2004), 424–431.

[War92] WARD G. J.: Measuring and modeling anisotropic reflection. In *ACM SIGGRAPH '92* (1992), pp. 265–272.