JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 273-288 (2017)

DOI:10.1688/JISE.2017.33.2.1

An Artist Friendly Material Design System

TSUNG-SHIAN HUANG, JEN-HAO LIAO, WEN-CHIEH LIN*
AND JUNG-HONG CHUANG*
Department of Computer Science
National Chiao Tung University
Hsinchu, 300 Taiwan
E-mail: {csie9624; howardlyliao}@gmail.com; {wclin; jhchuang}@cs.nctu.edu.tw

Material design is an important task in game and animation industry. It often requires artist's great efforts to create an elaborate material. Although several successful material editing systems already exist, it is still hard for artists to design a material on their own since these systems usually have neither an intuitive and sufficiently sophisticated material representation nor an artist-friendly editing interface. In this paper, we propose a simple material creation flow and develop a material design system based on an intuitive material representation. With the system, an artist can bring their creativity into play on designing materials that are realistic or imaginary.

Keywords: computer graphics, rendering, texture, BRDF editing, material design

1. INTRODUCTION

Materials such as brick, wood or cloth are everywhere in game and animation. It is an important feature to enrich a virtual world and makes objects look alive. Thus, creating an appealing rendering result of a material is crucial. Note that an appealing rendering result is not necessarily photo-realistic, since the world in game and animation is often full of imagination.

Material design is not an easy task, which often involves the collaboration of artists and technical artists. An artist designs the base textures and a technical artist designs the lighting phenomena of the material using the base textures as the input. Some parameters are then tuned by the artist for finer rendering results. Although several successful material editors have been developed, it is still hard for the artists to really manage these tools since the control of the appearance of a material is mainly restricted to the parameters provided by the technical artist. As there are no intuitive material representations and artist friendly editing tools to design the appearance of a material, many passes back and forth between the artist and the technical artist are usually needed to design an appealing material.

When considering the rendering of a material in reality, Bidirectional Texture Function (BTF) [2] is often a good choice. BTF is a six-dimensional function, involving lighting, viewing direction, and position. Since BTF data are captured from a real material, it can produce photo-realistic appearance. Many methods have been proposed to compress the BTF data. Some of the methods also come with some editing capabilities which are, however, usually limited. Such editing tools usually lack the editing abilities of a drawing system, which is much more friendly for an artist.

Received March 8, 2016; revised May 6, 2016; accepted May 26, 2016.
Communicated by Yung-Yu Chuang.
* Corresponding author.

Recently, Menzel and Guthe [10] proposed a novel representation called geometric BRDF (g-BRDF), which decomposes BTF data into meso- and micro-scale properties represented by images (texture maps). Although some BTF editing results are demonstrated, their work focused on BTF modeling problem. More importantly, the input to their system are BTF data of materials, which are not conveniently available. These observations motivate us to develop a system that allows artists to design general materials via a single or a few images and requires less effort from the technical artists.

We propose a simple material creation flow and a material design system based on an intuitive material representation. The representation consists of the geometrical part, diffuse part and specular part of the material. Such intuitive representation allows artists to design a material without much difficulty.

Given a single or a few images of a material, the artist can first generate the textures that represent the material by using some image processing tools and then manipulate and refine each texture using the proposed design system. Our design system is implemented as a plugin of an image manipulation program, thus all the manipulations can be viewed in real-time and the editing tools are much friendly to the artist. The capabilities of the editing tools are not restricted to what are currently provided in our system. Any other plugins can be added to strengthen the design process. With the proposed system, artists can bring their creativity into play to design complex materials intuitively with flexible controls over the appearance of the materials.

2. RELATED WORK

Material Design In game and animation industry, software and application programming interfaces, such as Unreal Engine, Frostbite Engine and RenderMan, provide a node-based shader synthesis tool to edit the appearance of a material. Users can design the appearance of a material in a visual programming way by connecting some operators (*e.g.*, multiply, add, *etc.*) and operands (*e.g.*, texture, texture coordinates, constant, *etc.*). Such connectivity is called a node expression, from which a shader code is automatically generated to render the material. Thus, users can design and render a material without knowing how to write a shader code. They can create a node expression that is well tailored to a specific material; however, it is difficult for an artist to design such expression since knowing how to manipulate all the operators and operands to create a material requires a lot of experiences and computer graphics knowledge. Moreover, the manipulation of the operands itself is not readily provided in these systems, *e.g.*, painting the texture or warping the geometry. Artists need to tune numerical parameters to modify the appearance of the materials, which is nonintuitive and can be very tedious sometimes. Some basic texture editing, such as changing the color of a region, which can be done very easily through an image manipulation program, would be troublesome by using the node expression.

Dong *et al.* proposed an interactive material modeling system [3]. Given a single image lit by a known directional light source, their system lets the user interactively separate the image into an albedo map and a shading map, and then reconstruct the normal from the shading map to recover the geometric features. Although this system can reproduce the appearance of a variety of materials, it cannot handle complex specular phenomena, such as anisotropic reflection. Besides, the system does not provide more sophisticated editing abilities, which are crucial for material design.

Nguyen *et al.* provide another way to edit materials [13]. Users can directly paint on the surface of a 3D scene, and the system modifies the nearby material to achieve the desired appearance. Their system also takes indirect lighting into account. Instead of editing the BRDF directly, they try to fit the expected exitant illumination. The editing result may not be able to apply to other scenes directly because it depends on the relative positions of the arranged objects and lights in the scene.

BTF Compression and Rendering To represent a material in reality, Dana *et al.* [2] introduced the BTFs. A BTF is obtained by taking lots of photos of a material under different lighting and viewing directions. Hence, a BTF of a single material may need several gigabytes for storage. Such enormous data size is not suitable for rendering a BTF on current graphics hardware. Thus several methods have been proposed to compress the BTF data, *e.g.*, principal component analysis (PCA), matrix factorization, optimized Spherical Radial-Basis Functions (SRBF) [17]. These approaches reduce the high-dimensional input data to low-dimensional subspaces containing most of the information. Thus, BTF can be rendered in real time while preserving good visual quality. The compressed data do not offer meaningful parameters to control the appearance of the material. For a nice survey on BTF modeling and compression, please refer to [4, 12].

BTF Editing BTF editing has its own practical demands due to the inconvenience and difficulty of BTF acquisition. Existing BTF editing approaches can be roughly divided into two categories: direct editing approach [8, 19] and fitting approach [10, 18]. Kautz *et al.* proposed an out-of-core data management for editing raw BTF data [8]. They provided sophisticated operators to edit BTF appearance such as shadows, specularity and meso-structure. A drawback of this approach is that if the meso-geometry of the material is changed, the corresponding feature (*e.g.*, shadows) does not change accordingly. Xu *et al.* proposed a novel editing propagation scheme to edit BTF data [19]. View-independent features, such as normals and reflectance reconstructed from each view, are used to guide the propagation process. Although their editing approach does not rely on explicit geometry, this also implies that it cannot edit the meso-geometry of a BTF material.

The fitting approaches adopt a BRDF model to fit the BTF data. These approaches are advantageous since BRDF models provide physically meaningful parameters for users to control; however, they usually lose accuracy as they either fit the BTF data using a single BRDF model or do not consider the underlying meso-geometry of the surface into account. Recently, Wu *et al.* [18] proposed a Sparse Parametric Mixture Model (SPMM) to represent general BTFs. They adopted the stagewise Lasso algorithm to fit a BTF to different types of multiple, rotated analytical BRDFs. This representation preserves the appearance while the storage size remains compact. The editing task is accomplished by changing the parameters of the analytical models; however, without modeling the meso-geometry of a material, the control over the meso-geometry of the material cannot be achieved.

In general, to accomplish the editing task, most BTF editing approaches need a specific user interface tailored for a specific algorithm. Although such a user interface provides simple editing, *e.g.*, changing color, adjusting specular intensity or surface roughness, these editing operations are usually global. There are almost no artist friendly tools that support editing for a certain local region. Moreover, as the user interface is not drawing-based, ed-

iting is less intuitive for artists. Recently, Menzel *et al.* [10] proposed a novel way called g-BRDF to represent a BTF. The BTF data is decomposed into a height map, which describes the meso-geometry, and some other maps that store the BRDF parameters. This method provides high compression rate and good rendering quality; however, as g-BRDF addressed the BTF modeling and editing problem, it is not intuitive for artists without technical background to use it for designing a material directly. Furthermore, it is also not easy to acquire the BTF of a material. This inspires us to develop a material design system that is intuitive and friendly for artists and other novice users.

3. OUR APPROACH

Our material design system is built upon a material representation which is easy to understand and manipulate. Artist-friendly material creation and editing can be achieved by directly applying image editing and drawing tools to texture maps that constitute the representation of a material. We introduce our material representation model and material design system in this section.

3.1 Material Representation Model

We modified the g-BRDF proposed by Menzel and Guthe to represent a material for its straightforward representation and photo-realistic rendering quality [10]. The g-BRDF is originally proposed for representing BTF data. For a given BTF, g-BRDF first extracts the meso-geometry (height map and normal map) from the BTF using the graph cut stereo technique [14] and then fits the BTF to a BRDF model with the meso- geometry information considered. Fig. 1 illustrates the workflow of the g-BRDF model. The g-BRDF employed Ashikhmin's distribution-based BRDF [1] as its BRDF model since it provides a good approximation for many real world materials with intuitive meaning. The BRDF model is defined as follows:

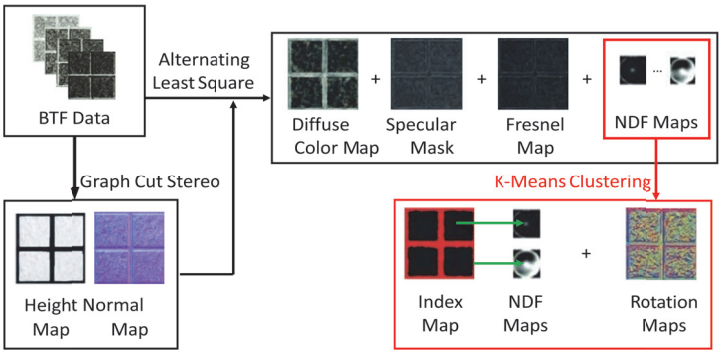


Fig. 1. g-BRDF decomposition workflow.

$$\rho(L,V) = \frac{C_d}{\pi} + \frac{C_{s\_int} P(H) F(L \cdot H)}{(L \cdot N) + (V \cdot N) - (L \cdot N)(V \cdot N)}, \tag{1}$$

where L and V are the lighting and viewing direction, respectively, C_d is the diffuse color, C_{s_int} is the specular intensity, $P(H)$ is the normal distribution function (NDF) depending on the half vector $H = (L + V)/|L + V|$, and $F(L \cdot H)$ is Schlick's Fresnel term defined by

$$F(L \cdot H) \approx r0 + (1 - r0)(1 - L \cdot H)^5, \tag{2}$$

where r^0 is the reflectance at normal incidence.

As a BTF can be viewed as a combination of spatially-varying BRDF and meso-geometry, the parameters in Eq. (1) vary at different pixel locations. These parameters can be obtained by fitting the BTF data using an alternative least square approach and then stored in texture maps, which include a diffuse color map, a specular intensity map, a Fresnel map and several NDF maps. Each NDF map represents the NDF at each pixel location of the BTF (for a BTF with 256×256 image resolution, the number of the NDF maps would be 256×256). To reduce the storage size of NDF maps, all the NDF maps are approximated by some NDF maps with different rotations around the center using the K-Means clustering. An index map and a rotation map are additionally defined to indicate which NDF Map is used and how much rotation is needed for each pixel, respectively. To reproduce the NDF map of a pixel, the corresponding NDF map is retrieved according to the index stored in the index map, and then rotated around the center with the degree stored in the rotation map.

In g-BRDF, the same NDF is used for different color channels. The NDF function describes the percentages of the micro-facet normal at every direction on a hemisphere. It is stored in a parabolic map [7] and accessed by the half vector (more specifically, the inclination angle and the azimuth angle of H). To obey the law of conservation of energy, the sum of probabilities over hemisphere cannot exceed one. However, it is obscure for artists to manipulate NDF while satisfying this constraint.

To model more complex specular effects and provide better control, we use different NDFs for different regions of a material. Because $P(H)$ represent the micro-facet normal direction that can reflect the light into the viewing direction, the NDFs indeed is the specular color. Therefore, we replace the gray-level NDF maps by the RGB specular color maps in our representation. Also, we multiply the specular color maps by *Specular Mask Map* to allow a user to specify the specular regions easily. Finally, we introduce the *Rotation Map* and *Tilted Reflection Map* to control the reflection of the material. With these modifications to g-BRDF, the BRDF model in our material representation is expressed as

$$\rho(L,V) = \frac{C_d}{\pi} + \frac{C_{s\_mask} C_s(I_s, \tilde{H}_{RT}) F(L \cdot H_T)}{(L \cdot N) + (V \cdot N) - (L \cdot N)(V \cdot N)}, \tag{3}$$

where C_{s_mask} is the specular mask value; $C_s(I_s, \tilde{H}_{RT})$ is the specular color depending on a map index I_s, a rotated and tilted half vector HRT ($\tilde{H}_{RT}$ denotes the inclination and azimuth angles of HRT). HT represents a tilted half vector. Eq. (3) denotes the BRDF at a pixel location. To encode the spatial variation of C_d, C_{s_mask}, Is, rotation angle of the half vector, and the Fresnel terms, we store these material-dependent parameters into texture maps to represent the diffuse and specular appearance of a material. Fig. 2 illustrates our material representation. We describe more details of these maps in the following section.

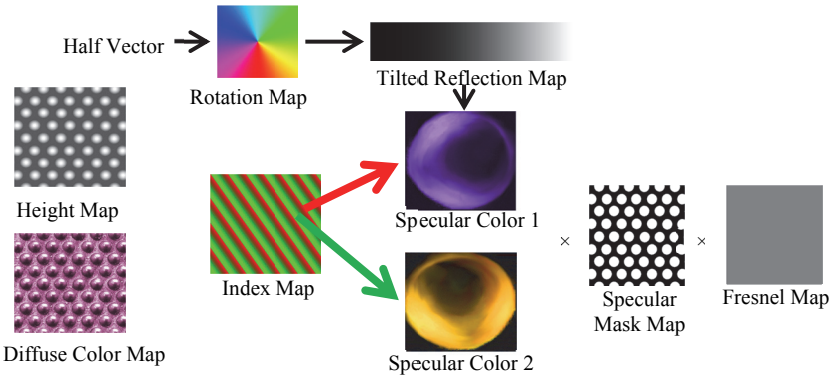


Fig. 2. Our material representation model.

3.2 Material Design System

Based on the proposed material representation model, we develop a material design system. Given a single or a few images for a material, our system constructs the texture maps that represent the material's meso-geometry, diffuse part, and specular part. Then a user can manipulate each part through image editing or drawing to design a desired material. We describe the construction and modification processes of the texture maps for representing a material and the intuitive interpretations of these processes.

3.2.1 Building and editing Meso-geometry

Some materials may contain self-shadowing and parallax effects due to its meso-geometry. To obtain meso-geometry information without the BTF data, we can either use photometric stereo (for the input with more than one images) or use the gray-level intensity of the image (for the input with a single image) to get the approximated height map and normal map.

Normal map and height map In the case of multiple input images, the photometric stereo method proposed in [15] is adopted to recover the normal map by assuming that the material is a Lambertian surface and illuminated from a distant point light source. The normal integration method proposed by Frankot and Chellappa is then applied to recover the height map [5].

In the case of a single input image, the gray-level intensity of the image is a good approximation of the meso-geometry of those materials with less specularity and color variations. To use the gray-level intensity of an image as the height map of a material, it is important that the material is lit evenly from all directions to avoid the shadowing and specular effects. As the approximated meso-geometry may not be very accurate and a user may also want to make the material smoother or rougher, our system allows the user to edit the meso-geometry by manipulating the height map of the material; however, simply editing the height map may cause the inconsistency between the normal and height maps. To solve this problem, our system only stores the height map and the normal map is calculated by applying 3×3 Sobel filters to the height map at runtime. By

convoluting the height map with the Sobel filters, we can obtain the derivatives along the *x* direction (*dx*) and y direction (*dy*), respectively. The derivative along the *z* direction (*dz*) can be calculated via $1-\sqrt{dx^2+dy^2}$. Thus the normal at a pixel location would be (*dx*, *dy*, *dz*). Since we calculate the normal map at runtime, the user can see the changes on the fly without manually rebuilding the normal map.

3.2.2 Building and editing diffuse and specular parts

Diffuse color map The diffuse part of material scatters the incident light uniformly over all directions. It provides the foundational color which can be seen from every viewing angle. We simply use the original input image as the diffuse color map of the material. If users are not satisfied with the diffuse color map, they can further edit it to achieve the desired result. When the surface of the input image is not perfectly diffuse, the intensity of the diffuse color map can be decreased to exclude the specular intensity.

Non-diffuse terms To model complex specular effects, such as anisotropic reflection, and provide intuitive editing of specular effects, such as adjusting the shape, region, and angular dependency of specular highlights, we represent the specular part as a combination of an index map, specular color maps, a specular mask map, rotation map, a Fresnel map, and a tilted reflection map. Index map, specular mask map, Fresnel map, and rotation map are position dependent, *i.e.*, the map value varying with spatial location, while the specular color map is angle dependent. In particular, index map can be considered as a rough classification of the specular properties of the material. For each class, a specular color map is used to describe the shape and color of specular highlights under different half vectors. The specular mask map is used to define if a region has any specular effects. The rotation map and tilted reflection map affect the orientation of highlight areas in anisotropic reflection. These maps not only provide a sophisticated model for specular effects but also have intuitive interpretations for artist friendly material design.

Index map From the index map of different materials in [10], we observed that regions of a material having similar diffuse colors often share the same index. Thus we can use the color selection, edge detection, color mapping/remapping tools or any other image processing tools to segment the input image of the material into parts of similar color. Each segmented part is then indexed by an index map. Note that there are multiple specular color maps in our representation. An index map may contain multiple channels, where each channel stores the weight for each specular color map.

Specular color map After segmentation, the next step is to design the specular color map for each region. The specular color is stored in a parabolic map and accessed using the half vector. Although using the half vector to access the specular color is straightforward, it may be still difficult for artists to design a parabolic map. It is not clear for artists to determine which pixel in the parabolic map is accessed from given lighting and viewing directions. To tackle this problem, our system provides a special designed window with nine viewports. Each viewport shows the appearance of the material at different lighting and viewing directions over the upper hemisphere (Fig. 3). To let the user know which pixel in a specular color map corresponds to which lighting and viewing directions,

we mark the lighting and viewing direction of each viewport as a red point on the 'Specular Position' block at the bottom right. The user can draw the parabolic map and receive the feedback in real-time. For artists, the specular color map can be intuitively considered as the shape of specular highlights. They can manipulate it by drawing on the specular color map. Fig. 4 shows some examples.

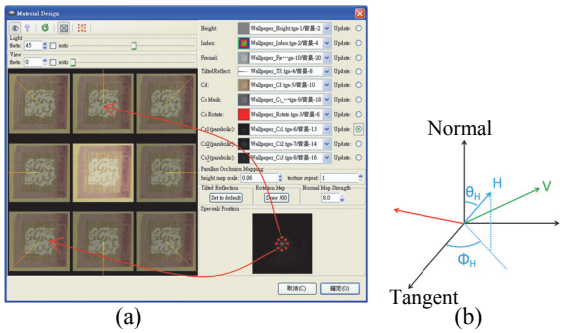


Fig. 3. (a) Design interface for specular color maps; (b) The half vector can be represented by θ_H and φ_H.

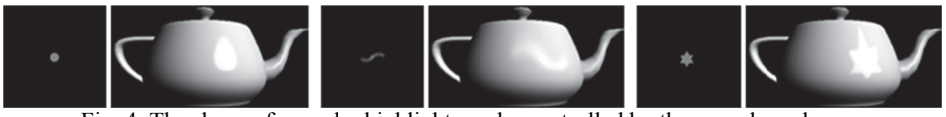


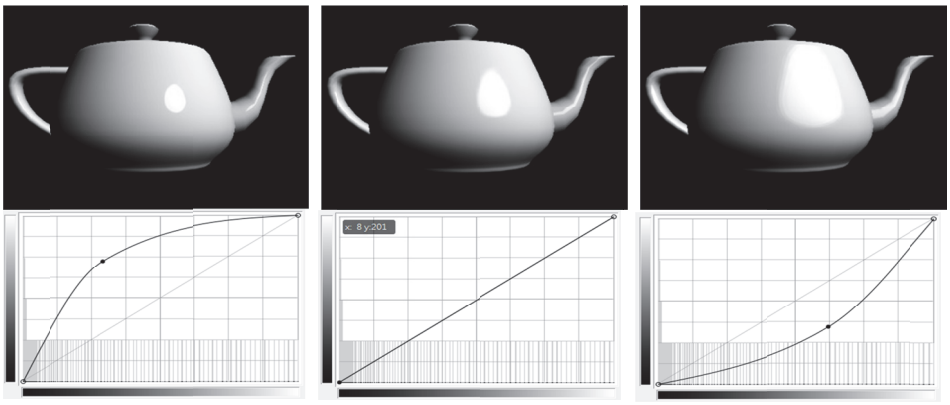
Fig. 4. The shape of specular highlight can be controlled by the specular color map.

Specular mask map and Fresnel map For a material that is lit evenly from all directions, brighter regions tend to have lower ambient occlusion or higher reflection. Hence, we use the intensity of the diffuse color map as the initial specular mask map, and the user can change its brightness and contrast via our design system for better results. The Fresnel term expresses the reflection of light on material. We adopt Schlick's Frensel term, which increases from r_0 to 1 as the angle between L and H increases from 0 to 90 degrees. The materials become brighter when they are illuminated at the grazing angle and viewed from the opposite side. Therefore, similar to the construction of the specular mask map, we also use the intensity of the diffuse color map as the initial Fresnel map. Its brightness and contrast can be adjusted later for better results. The specular mask map can be roughly considered as the location of the specular highlights.

Rotation map and Tilted reflection map When dealing with materials with anisotropic reflection, *i.e.*, the highlight changes continuously over the phi angle (*e.g.*, disk and metal surface), it is hard to use a lot of discrete specular colors for representation. To solve this problem, the rotation map is introduced. Each pixel of the rotation map stores a rotation angle $\Delta\Phi$ for the corresponding specular color map. Given a half vector H, whose direction is (θ_H, Φ_H) as defined in Fig. 3, the rotated half vector is represented as $\tilde{H}_{RT} = (T(\theta_H), \Phi_H + \Delta\Phi)$. Here $T(\theta_H)$ is called a tilted reflection map, which is indeed a 1D mapping representing a tilting to the half vector. $T(\theta_H) = \theta_H$ if no tilting is applied. The same tilt reflection map can also be applied to the Fresnel term through $H_T =$

$Vec(T(\theta_H), \Phi_H)$, where $Vec(\theta, \Phi)$ is a unit-length vector with direction (θ, Φ).

The rotation map is encoded as hue component in HSV color space, *i.e.*, the hue component is used to define the rotation angle in our material editing system. Our system provides a function allowing the user to specify arbitrary rotation at any region by drawing, so a rotation map can be generated easily. The purpose of the tilted reflection map is to allow artists to control the visual roughness of surface by adjusting the size of highlight area. To achieve this, the tilted reflection map remaps the angle between half vector and surface normal, *i.e.*, θ_H, to a new value. By remapping θ_H, an artist can visually tune the highlight area as shown in Fig. 5. The effect controlled by tilted reflection map is similar to the shininess of Phong lighting model. The default tilted reflection map is no tilting, *i.e.*, a linear mapping (a straight line of slope one).



(a) Small highlight area. (b) Default linear mapping. (c) Enlarged highlight area.

Fig. 5. The highlight area is controlled by tilted map which can be turned by curve tool.

4. EXPERIMENTAL RESULTS

We implement the proposed material design system using OpenGL with GLSL. To ease the user interface development, we implement our system as a plugin of GIMP (GNU Image Manipulation Program), which provides many standard image processing and drawing tools. Considering the real-time feedback, we carefully manage the resources by sharing the same texture objects and shader context for the object preview window and specular design window. Also, to reduce texture accesses time, all the specular color maps are combined into a 2D texture array. When changes are applied to a specific specular map, we only update the corresponding layer of the 2D texture array rather than regenerate the whole texture array. For real-time applications, the normal map is precomputed from the height map and stored with the height map as an RGBA texture. Also, the specular mask map, Fresnel map, and the rotation map are stored as an RGB texture. All the specular color maps are merged into a 3D texture to reduce the storage. In our experiments, three specular color maps along with the index map, specular mask map and rotation map are sufficient to represent most materials. If a material is very complex, an additional index map can be used to indicate more specular color maps. In that case, all the index maps are combined into a 3D texture. Besides, parallax occlusion mapping

[16] is applied to simulate the self-shadowing effect of the material. On an Intel Core 2 Dual E8400 3.0GHz with Nvidia GTX 260 desktop, our system achieves 1450 fps when rendering 10 viewports (one 512×512 object preview viewport and nine 170×170 specular design viewports) at the same time while a 256×256 texture is continuously updated. All the materials in our results are created from a single photo or edited from an image.

4.1 Material Design and Editing

As our system represents a material using texture maps, material editing can be achieved easily by manipulating these texture maps as images. Moreover, since our system is built upon GIMP that provides handy image processing and drawing tools, a user can modify these maps and see the edited material immediately. In Fig. 6, we change the brightness and contrast of the height map to make the material's surface smoother globally or locally. We can also use warping tools to create artistic examples.

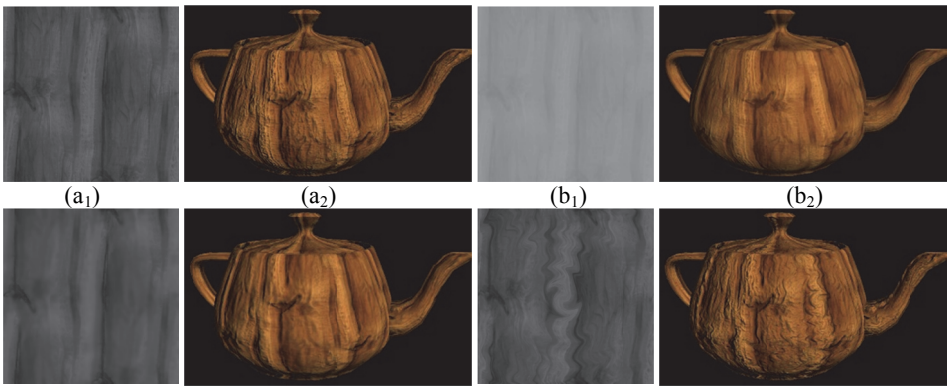


Fig. 6. Material editing results by changing the height map of wood; (a_1)-(d_1) are the height map used to generate (a_2)-(d_2), respectively; (a_1) Original height map; (b_1) Changing the brightness and contrast to smooth the height map globally; (c_1) Using 'Blur/Sharpen' brush to smooth the bright color part locally; (d_1) Using 'Interactive Warp' to warp the height map counterclockwise.

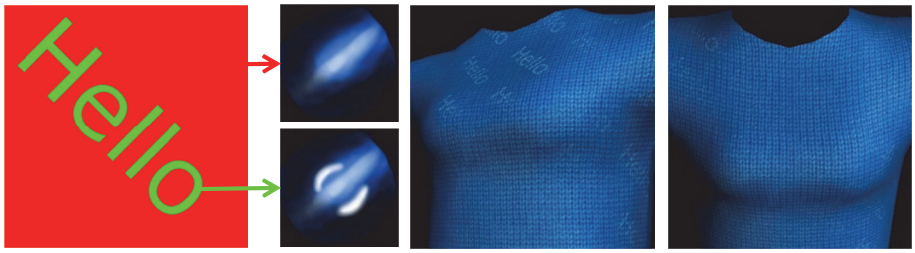


Fig. 7. Specially designed specular color maps and their rendering result. The 'Hello' text can only be viewed under certain viewing directions.

A specular color map stores the specular color under different lighting and viewing directions. It can be designed to embed some marks in the material which can only be observed in some specific viewing or lighting directions. This kind of appearance mostly appears on sports coat or aluminum paper with embossed patterns. In Fig. 7, we duplicate the first specular color map, and then use blush to add two bright curves into the second specular map. After that, the text tool is used to place the "Hello" pattern in the index map which indicates the spatial domain for these two specular effects. The "Hello" text can be seen on the chest of the cloth in the first view direction, but the text disappears for the second view direction.

Rotation map can be utilized to produce special effects. In Fig. 8, anisotropic reflection is achieved by editing specular color maps together with rotation map. The rotation map is filled with the color wheel which circles the center. And a colorful belt is attached to the specular color map which affects the angular behavior of the highlight. One can see the color of highlight changes when lighting or viewing direction rotates. The colorful specular highlight makes our result quite different to the traditional texture mapping.

Fig. 8. Anisotropic reflection on disk.

Our system can also be used to create materials that do not exist in reality. In Fig. 9, we specially design the specular color map such that the light blue dots of the material have blinking effects. The first specular color map is created by spray tool. The spots of specular map make the highlight visible in multiple directions. And the index map is extracted from diffuse map using segmentation tool. The color of index map controls the spatial distribution of the blinking effect. The sparky highlight can be observed easily by comparing our result with the one of texture mapping.

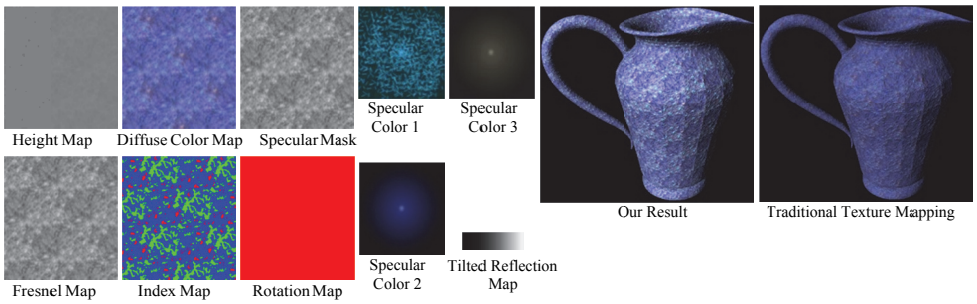


Fig. 9. Artificial porcelain vase.

Since BTF data is not easy to acquire, artists can create BTF liked material from a single image using our design system. To verify the quality and the capability of our ma-

terial design system, we compare our result with that rendered by using raw BTF dataset. One image of a BTF dataset is chosen as the input, and it is manipulated to create the representation maps for a material that is expected to resemble the given BTF material. Fig. 10 shows the wool material and the corduroy material. Note that the BTF data does not record the appearance of the material when the θ angle is larger than 80 degrees. In these regions, the BTF data is approximated by those with $\theta = 80$ degrees. The slight differences between our result and the BTF rendering result are caused by inaccuracy of height map and the specular map. Although it is hard to create a material that has exactly the same appearance as the BTF material in reality, our results capture the main features of the tested materials and thus produce quite similar results.

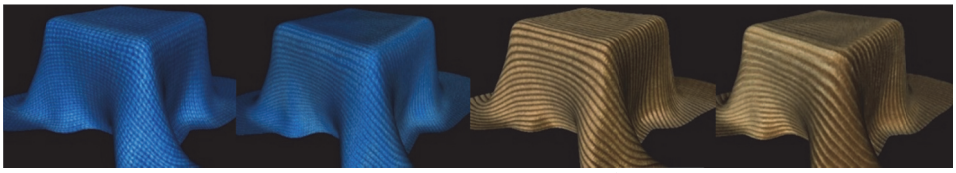


Fig. 10. Comparison with BTF.

4.2 Testing by Artists

To test our system and get user feedback, we invited three game artists to use our system. Figs. 11 and 12 show two examples designed by the artists. In both examples, imaginary materials with complex anisotropic reflection are generated. One can see the special highlight in both examples created by artists. Note that the light source in Fig. 12 is a single directional light, but three different specular highlights can be seen in the result. These two examples show that the artists can utilize our material design system to create materials with special artistic effects they want.

We also interviewed the artists after they used our system. All of them reported that the control of the spatial domain is quite easy to understand, such as diffuse map, specular mask, and the height map. For angular domain, artists can quickly realize how the specular color maps affect the glossy effect and the shape of the highlight area by observing the real-time response when painting the specular color maps. The rotation map and index map were also easy for the artists to understand and use. The artists even tried to mix multiple specular effects by blending multiple colors into the index map, as shown in Fig. 12. However, the concept of Fresnel term was difficult for artists, and we rarely saw the artists utilized the Fresnel map. Though we already implemented the system as the plug-in of GIMP, which is 2D painting software, artists still tended to edit the image by their favor 2D painting software, such as Photoshop. They then input the edited images into our system to see the result.

4.3 Limitations

Although our system can create many general materials using the proposed material representation, it does not support the translucent materials because the BRDF model we

used cannot handle the subsurface-scattering and inter-reflection phenomena. To extend our material design system for more versatile materials, a node-based shader synthesis tool can be incorporated. Another limitation is the accuracy of obtained data representation from a single image. For the materials without the BTF data, the photo-metric stereo is unable to be applied, and the data representation must be created from a single or few images. Although the meso-geometry we derived using gray-level intensity of the image can produce realistic visual effects, it is not accurate in reality. For accurate meso-geometry information, a reconstruction approach in [3] or other depth acquisition hardware can be applied. The similar issues also occur in the diffuse and specular parts since these parameters are mainly constructed from the image editing. Although artists can tune the parameters and examine the result from arbitrarily viewpoints, the connection between physical parameters and the edited parameters can still be loose. For instance, multiple sets of parameters, which may be quite different or physically incorrect, can still produce very similar appearance.

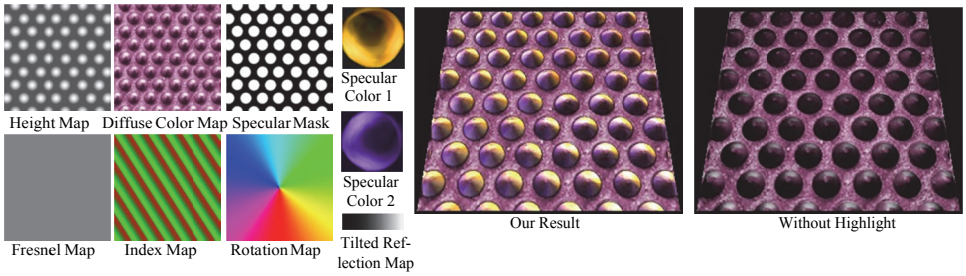


Fig. 11. An imaginary material made by artist.

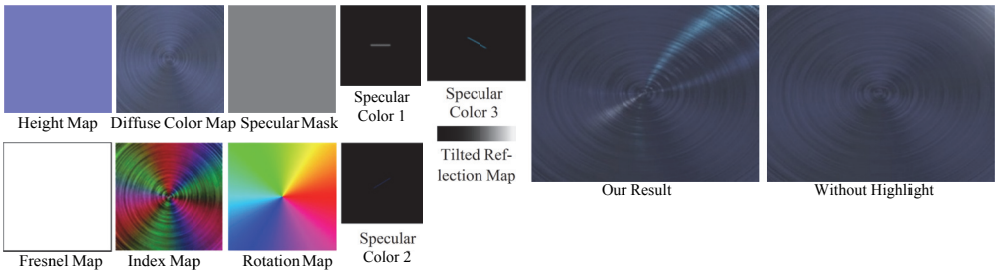


Fig. 12. Another imaginary material made by an artist. Note that the light source in this example is just a directional light, but multiple highlights appear on the material.

5. CONCLUSION AND FUTURE WORK

We propose a simple material creation flow and an artist-friendly material design system for artists and novice users in this paper. Artists can create appealing results from a single or a few images intuitively. Our system generates the texture maps to representing the material's meso-geometry, diffuse part, and specular part and user can design and edit a material by manipulating these texture maps. The material editing capability of our system is not restricted to the image editing tools provided in our system. Artists can

combine our system with any other plugins or image editing tools to help them design and edit materials. As a result, our system has a great strength in material editing. It can reproduce the appearance of general isotropic and anisotropic phenomena for real or imaginary materials. Complex specular phenomena, such as anisotropic reflection, can also be produced easily. Furthermore, material representation of our system stores all the lighting phenomena of a material in texture maps. Thus, given two or three materials, we may find a reasonable way to create a new in-between material using the texture transition idea in [9].

ACKNOWLEDGEMENT

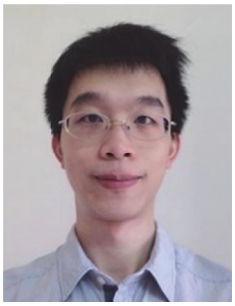
This work was supported in part by the Taiwan National Science Council (NSC 98-2221-E-009-101-MY3). We would like to thank Prof. Yu-Ting Tsai at Yuan Ze University for providing the BTF rendering program.

REFERENCES

1. M. Ashikhmin and S. Premoze, "Distribution-based brdfs," Unpublished Technical Report, University of Utah, 2007.
2. K. J. Dana, B. Van-Ginneken, S. K. Nayar, and J. J. Koenderink, "Reflectance and texture of real-world surfaces," *ACM Transactions on Graphics*, Vol. 18, 1999, pp. 1-34.
3. Y. Dong, X. Tong, F. Pellacini, and B. Guo, "Appgen: Interactive material modeling from a single image," *SIGGRAPH Asia*, 2011.
4. J. Filip and M. Haindl, "Bidirectional texture function modeling: A state of the art survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, 2009, pp. 1921-1940.
5. R. T. Frankot and R. Chellappa, "A method for enforcing integrability in shape from shading algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, 1988, pp. 439-451.
6. V. Havran, J. Filip, and K. Myszkowski, "Bidirectional texture function compression based on multi-level vector quantization," in *Proceedings of Computer Graphics Forum*, Vol. 29, 2010, pp. 175-190.
7. W. Heidrich and H. P. Seidel. "View-independent environment maps," in *Proceedings of ACM SIGGRAPH/EG Workshop on Graphics Hardware*, 1998, pp. 39-45.
8. J. Kautz, S. Boulos, and F. Durand. "Interactive editing and modeling of bidirectional texture functions," *ACM Transactions on Graphics*, Vol. 26, 2007, p. 53.
9. W. Matusik, M. Zwicker, and F. Durand. "Texture design using a simplicial complex of morphable textures," in *Proceedings of ACM SIGGRAPH*, 2005, pp. 787-794.
10. N. Menzel and M. Guthe, "g-BRDFs: An intuitive and editable BTF representation," *Computer Graphics Forum*, Vol. 28, 2009, pp. 2189-2200.
11. G. Mu¨ller, J. Meseth, and R. Klein. "Compression and real-time rendering of measured BTFs using local PCA," *Vision*, *Modeling and Visualization*, 2003, pp. 271-280.

12. G. Müller, J. Meseth, M. Sattler, R. Sarlette, and R. Klein, "Acquisition, synthesis and rendering of bidirectional texture functions," in *Proceedings of Computer Graphics Forum*, Vol. 24, 2005, pp. 83-109.
13. C. H. Nguyen, D. Scherzer, T. Ritschel, and H. P. Seidel, "Material editing in complex scenes by surface light field manipulation and reflectance optimization," in *Computer Graphics Forum*, Vol. 32, 2013.
14. S. Roy and I. J. Cox, "A maximum-flow formulation of the n-camera stereo correspondence problem," in *Proceedings of the 6th IEEE International Conference on Computer Vision*, 1998, pp. 492-499.
15. H. Rushmeier, G. Taubin, and A. Gueziec, "Applying shape from lighting variation to bump map capture,"*Eurographics Rendering Techniques*, 1997, pp. 35-44.
16. N. Tatarchuk, "Dynamic parallax occlusion mapping with approximate soft shadows," in *Proceedings of I3D*, 2006, pp. 63-69.
17. Y. T. Tsai, K. L. Fang, W. C. Lin, and Z. C. Shih, "Modeling bidirectional texture functions with multivariate spherical radial basis functions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, 2011, pp. 1356-1369.
18. H. Wu, J. Dorsey, and H. Rushmeier, "A sparse parametric mixture model for BTF compression, editing and rendering," in *Proceedings of Computer Graphics Forum*, Vol. 30, 2011, pp. 465-473.
19. K. Xu, J. Wang, X. Tong, S. M. Hu, and B. Guo, "Edit propagation on bidirectional texture functions," in *Proceedings of Computer Graphics Forum*, Vol. 28, 2009, pp. 1871-1877.

Tsung-Shian Huang (黃聰賢) received his BS degree from the Department of Computer Science, National Chiao Tung University, Taiwan in 2007. Currently, he is a Ph.D. student of National Chiao Tung University. He is interested in computer graphics and software engineering.



Jen-Hao Liao (廖仁豪) is a Software Engineer in MediaTek, Taiwan. He received his BS degree in Computer Science from National Sun Yat-Sen University, Taiwan, in 2009, and MS degree in Computer Science – Multimedia Engineering from National Chiao Tung University, Taiwan in 2011. His passion lies in computer graphics, software engineering, multimedia and game developments.

Wen-Chieh Lin (林文杰) is an Associate Professor at the Department of Computer Science, National Chiao Tung University, Taiwan. He received the BS and MS degrees in Control Engineering from National Chiao Tung University, Taiwan, in 1994 and 1996, respectively, and the Ph.D. degree in Robotics from the School of Computer Science at Carnegie Mellon University, Pittsburgh, in 2005. His research interests span several areas of computer graphics, human computer interaction, and visualization.

Jung-Hong Chuang (莊榮宏) is a Professor of Computer Science and Information Engineering Department at National Chiao Tung University, Taiwan. His research interests include 3D computer graphics, virtual reality, geometric modeling, and visualization. Chuang received his BS degree in Applied Mathematics from National Chiao Tung University, Taiwan, in 1978, and MS and Ph.D. degrees in Computer Science from Purdue University in 1987 and 1990, respectively.