MOCK-3D WEB APPLICATION: INTERACTIVE LIGHTING, RENDERING AND SHADING FOR 2D ARTWORK

A Dissertation

by

YINAN XIONG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Ergun Akleman
Committee Members,	Bruce Gooch
	Felice House
Head of Department,	Timothy McLaughlin

Spring 2016

Major Subject: Visualization

ABSTRACT

In this thesis, we developed a web-based tool to allow artists to create 3D-looking stylized depictions based on 2D artwork with complete visual control. The controls include multiple lights with diffuse reflections and specular highlights, and refraction and mirror reflection with Fresnel control. Our controls do not necessarily correspond to underlying physical phenomena; however, they still provided results that are visually similar to 3D realistic rendering.

The core of this approach is using paintable shape maps, which are similar to normal maps. The shape maps do not have to correspond to 3D shapes and, therefore, they can allow the artist to obtain incoherent and impossible 2D shapes with 3D appearance.

Another contribution is that we linearized Fresnel Curve so that it can be controlled by two sliders. This allows it to achieve an intuitive blending of the results of refraction and reflection.

ACKNOWLEDGEMENTS

I would like to thank Dr.Ergun Akleman, my committee chair, for providing me continuous support and inspiration while I was working on my thesis. I am grateful for his patience and his trust. I would also like to thank the rest of my committee, professor Bruce Gooch and professor Felice House for giving me refreshed perspectives and helping me look at my work in different ways during the progress. Thanks also to all the students, faculties and staff of the department of Visualization for all your time and everything. Your efforts and passion fuel me with energy. Your patience and help make me feel warm in this viz family. Thank for making viz lab such a loveable place and nurturing environment.

A series of thank you goes to my friends: Shenyao Ke, Siran Liu, Cherise Castille, Xiaoyi Zhang, Zhao Yan, Jiahe Bian, Phillip Rollfing. Thank you for being the ones I need when I was confused, lost and stressed. I am grateful for meeting you and thanks for being a good friend by my side. At last, I would like to say thanks to my parents back in China. Thanks for all the love and support since I was born. It has always been the most powerful thing that keeps me moving. And one more thanks to Devkan Kaleci for techincal support.

NOMENCLATURE

2D	Two-dimensional
3D	Three-dimensional
CG	Computer Graphics
NPR	Non-Photorealistic Rendering
FG	Foreground
BG	Background

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

1. INTRODUCTION

1.1 Motivation

Despite the significant advances done in 3D computer graphics and shape modeling, according to a recent market research 3D Graphics is still only 8% of the whole graphics market, while 2D graphics market such as vector, image and video constitutes the rest, i.e. more than 90%, of the graphics market [13]. Moreover, the 3D modeling market does not grow as rapidly as 2D painting/editing market.

There are several usual suspects to explain the reluctance of adapting 3D modeling such as 3D modeling is less intuitive, more expensive and requires more training than 2D. We think that there exists an additional and important reason. Using 3D, it is hard to include all types of expressive depictions that are caused by impossible, inconsistent and incoherent shapes (see Figure 1.1). This reluctance suggests that there exists a critical need to develop hybrid systems that can provide 3D effects along with the convenience and expressive power of 2D.

In this work, we developed such a web-based hybrid system that can support expressive depictions of impossible, inconsistent and incoherent shapes and scenes. Although there exists a significant amount of research on non-(photo)realistic rendering (NPR), except Wang et al. preliminary work [14, 15], there has not yet been a comprehensive expressive depiction system that is capable of an integrated non-realistic approach for both modeling and rendering. One problem with his system is that it is Windows based and nobody currently use it. Our application turns his system into a web-based program such that artist can use the system without a need for a specific configuration.

(a) (b) (c)

Figure 1.1: (a) An example of incoherent scenes: A cubist self-portrait by Pablo Picasso from 1907. In cubist paintings, the artists create images based on their successive and subjective experiences in both space and time [1, 2]. (b) A landscape painting by Richard Davison from 2001. Davison intentionally introduced contradictory vanishing points in this painting [3]. (c) A hand-drawn compositing of an impossible object into a photograph, by Qiao Wang.

1.2 Introduction

Our project is a web-based version of Wang's [15] initial Mock-3D rendering system. We built this system using WebGL. The original Mock-3D rendering system is developed in windows and it is hard for the people to appreciate its power. Therefore, we moved the system into web environment such that artists who are interested in the creation of interactive and 3D looking artwork can use our system without a need for specific configuration. This system can also be useful for people other than artists who want to demonstrate their work with motion and interactive images instead of still images.

The interface design is one of the key elements for effective use of the system. Our goal is to develop a simple interface that can quickly give users a clear understanding of what functions Mock-3D could provide. This is essential since most of our users may not necessarily have a training in 3D computer graphics.

Creating dynamic imagery with the web-based Mock-3D renderer requires only providing a set of input images. These input images can simply be created using any image manipulation software such as Photoshop or Gimp. The input images can be as little as three images which include two control images and a corresponding shape map, which is also an image. It is also possible to provide additional images such as foreground and background to obtain interesting reflection and refraction effects.

The two control images can be created by artist easily by turning their original artwork into a dark (unlit) version and a bright (fully lighted) version. Final result will be created by (1) interpolating between the dark and bright images using shading information derived from the shape map image, then (2) a compositing process that interpolate the transparent regions of the diffuse image with deformed background and environment maps blended using Fresnel. The artist can interactively control the illumination and rendering processes to intuitively obtain desired visual results.

Shape maps, themselves, are images, so they can be represented by using any convenient 2D raster or vector image format. The fact that the shape map is itself an image makes it a very painter-friendly representation, subject to creation and manipulation by both algorithmic and direct approaches. The shape map encodes 2D gradient and thickness information for all visible points of a shape. This information does not have to be complete or consistent. There are three ways of obtaining shape maps: (1) converting 3D shapes into 2D images, (2) directly painting a gradient domain image or (3) modeling using a sketch based interface. The most interesting shape maps are those sketched or painted by an artist since they can reflect the artist's intention, even if this does not follow the normal rules of perspective.

To develop our Interactive Mock-3D tool, we took Conrad Egans program based on WebGL and GLSL as a basic back-end program. We designed and developed front-end, including all the interface, interactive experience research and studied

and implemented artistic shaders regarding style and multiple lights, refraction and reflection section and Fresnel blending methods.

2. RELATED WORK

Even though the main goal of this work resides in the rendering and compositing, the mock-3D representation, shape map, resembles normal maps, and most of the previous literature do not distinguish modeling from rendering, so we provide a survey on both modeling along with rendering.

2.1 Normal Map Modeling

Simplest Mock-3D shapes are normal maps. They can directly be modeled in 2D without any 3D interaction. One way of modeling Normal Maps is to model normals.

Johnston presented the first skectch-based normal map modeling method, called Lumo [4]. Johnston is also the first person who notices that outlines of 2D drawings typically provide well-defined normals. Once the normal in the outline is determined, it is possible to diffuse normals from outlines into the empty regions inside of the shape. Johnston applied an iterative Laplacian kernel to diffuse the normal vectors from outlines [4].

Since then, many researchers developed normal map modeling methods. Sun et al.[16] introduced Gradient Mesh to semi-automatically and quickly interpolate normals from edges, and Orzan et al. [17] calculate a diffusion from edges by solving the Poisson equation. Sykora et al. [18] proposed Lazy-Brush, which can propagate scribbles to accelerate the definition of constant color regions. Finch et al. build thin-plate splines which provide smoothness everywhere except at user-specified tears and creases [19]. The underlying splines are, then, used to interpolate normals. Wu et al. [20] proposed shape palette, where user can draw a simple 2D primitive in the 2D view and then specify its 3D orientation by drawing a corresponding primitive. This method also performs diffusion using a thin-plate spline. Shao et al.

Shao et al. developed CrossShade by using an explicit mathematical formulation of the relationships between cross-section curves and the geometry they aim to convey [5]. The specified cross-section point is used as an extra control point to control the normals. Bui et al. recently developed another method to generate normal maps from simple sketches having outlines and hatching strokes[6].

Vergne et al. [21] introduces surface flow from smooth differential analysis, which can be used to measure smooth variations of luminance. Therefore, the author also proposes to drawing the shadows and other shading effects. Sykora et al. [22] developed a user-assisted method to convert normal maps into Bass-Reliefs that can provide correct shadows in a commercial renderer, but this approach will fail if the normal maps do not correspond to shapes that can have an explicitly meaningful 3D geometry.

2.2 Rendering and Compositing

The rendering technique we developed follows the general branch of non-photorealistic rendering (NPR), where rendering mainly works on normal maps. NPR shading models are often simply functions of the surface normal and light direction that result in effects such as Gooch shading [23], cartographic hill shading[24], or other artist-specified effects. A more complex model includes curvature-based shading [25] and "exaggerated shading" [26]. However, all these techniques are only available in 3D with a normal and corresponding position information. However, shaded appearance may be designed through a painting interface, even though they may not be photorealistic, such as tweakable light and shade [27]. In tweakable light and shade [27], Anjyo et al. proposed to control light and shade inside a shape by dragging highlighted area, and an underlying normal map still needs to be estimated.

The rendering technique we developed is mostly related to the work of Gois et

al. [28]. Gois proposed a technique relies on the graphics pipeline to infer relief and to simulate the 3D rotation of the shading effects inside the 2D models in real-time. They demonstrate the application on Phong, Gooch and cel shadings, as well as environment mapping, fur simulation, animated texture mapping, and (object-space and screen-space) texture hatchings. Our rendering method is also related to the work of [29]. In this work, Toler et al. created non-photorealistic illustrations from a type of data lying between simple 2D images and full 3D models, named RGBN image, which contains both color and a surface normal information. However, some limitation exists such as shadows, which are only considered at discontinuities of normal. The proposed reflection/refraction methods share similarity with the work of Ritschel et al. [30], as we both conduct reflection/refraction in a non-physical way. Ritschel et al. introduce a sketch-based interface for artists to create reflection effects easily. Later on, they [31] improved the interface that can also edit shadows, caustics, and indirect illumination. However, their results highly depend on the scene, the camera motion, and the performed edit. And all of these have to be tuned carefully by the user.

3. MOCK-3D OBJECTS

The concept of Mock-3D objects are introduced by Youyou Wang et al [15] based on Lumo [4]. In this chapter, we will give the defination of Mock-3D objects, which has its shape and its material properties. Then, We will talk about how to create the shape of a Mock-3D object.

3.1 Defination of Mock-3D Objects

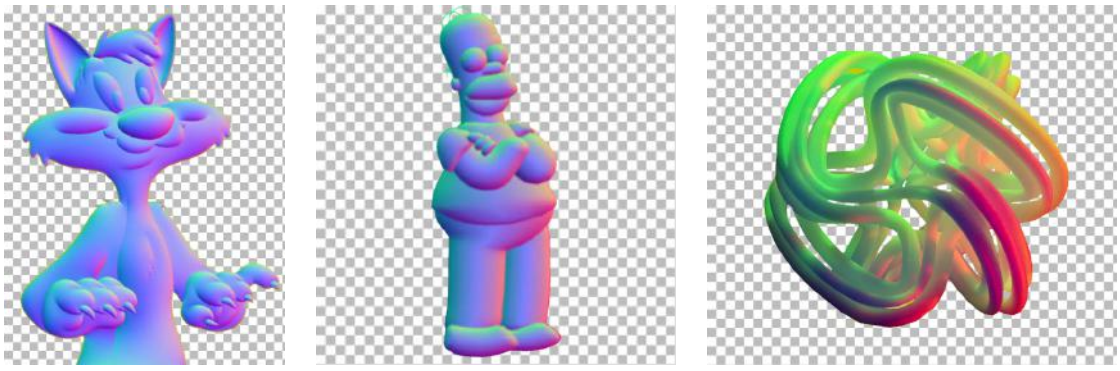
A Mock-3D object consists of two elements: its shape and its material properties.

3.1.1 Shapes of Mock-3D Objects

In our thesis, the shape of a Mock-3D object is given by a image, which is called Shape map[14]. Two examples of shape maps are shown in Figure 3.1. In these images, r, g and b values provide a 2D vector field and thickness field. The 2D vector field is used to construct normals to the shape and thickness field is used to provide an approximate thickness for the object (see Figure 3.2).

The concept of shape map image is inspired by Johnson's Lumo idea. As Johnston said in Lumo [4], "The primary components to illuminate a point on a surface is its position and its normal map". The main advantage of using images to represent the shapes of objects is that images are easy to create and modify. In other words, creating and modifying a shape map is much more simple than building a model in a 3D application. There are several ways to create this shape image, which will be explained in next section.

Unlike the traditional normal map, our shape map image can have pixels $alpha = 0$. RGB channels of a shape image represent the normal of the point face to the top, right or toward the camera, alpha channel represent if the object is exist or not. We



(a) (b) (c)

Figure 3.1: Two examples of shapes of Mock-3D objects as shape map images. (a) is a cat model created by Lumo application by Johnson [4]. (b) is a cartoon character created by the shady program developed in Texas A&M. (c) is a shape map created by photographing a real object illuminated by red and green lights.

(a) Vector field (b)Thickness field

Figure 3.2: RGB channel of a shape map: R & G channel represent Vector field, B channel represent Thickness field

treat the shape map as the object, and it is separated to the background. If the object does not exist, Mock-3D will composite the main character with Background Image.

Since a piece of artwork has already embedded perspective view or may not which are all intendedly designed by artists. So that Mock3d project applied an orthographic view to render our scene. Therefore, the aspect ratio of the result is only based on the shape map image, which I call *"Shape Image"*.

As shown in Figure 3.3, it demonstrate the orthographic view of a camera, it also show our Mock-3D coordinate system. We calculate (x, y) pixel position of the result rendering image by using the according (u, v) position of our shape image. Since we are not considering shadow at this phase, what we need for illumination is the according normal vector of the point we get from shape image. So we consider the result image is mapped on the plane of $z = 0$, in which the center of the image is on the coordinate origin - black dot on the figure.

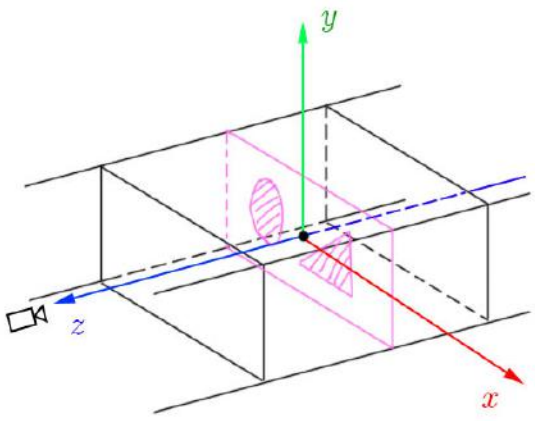


Figure 3.3: Mock-3D orthographic rendering and coordinate

3.1.2 Material Properties of Mock-3D Objects

For every point of a Mock-3D shape, we need to describe material properties. For instance, some objects may be transparent like glass or water; some other objects may be reflective such as mirrors, human eyes or waxed surfaces. Some materials can only be diffuse. To represent a wide variety of materials we use (r,g,b,α) images, where (r,g,b) terms are used to represent diffuse properties and α term represent both reflectivity and transparency. To differentiate only-reflective materials from transparent materials, we use Fresnel. A constant Fresnel function of one turn a transparent material into only-reflective material.

For rendering, we use barrycentric algebra suggested by [32]. In that approach, rendering is obtained as a weighted average of a set of control texture images mapped on 3D shapes. Since shape maps are simply rectangular images in our case, there is no need for texture mapping and images that are the same size of shape maps can simply be used a control textures. Therefore, material properties of Mock-3D objects are provided by a set of images [32]. In the web application, we use two images: a Bright Image and a Dark Image to represent look of materials.

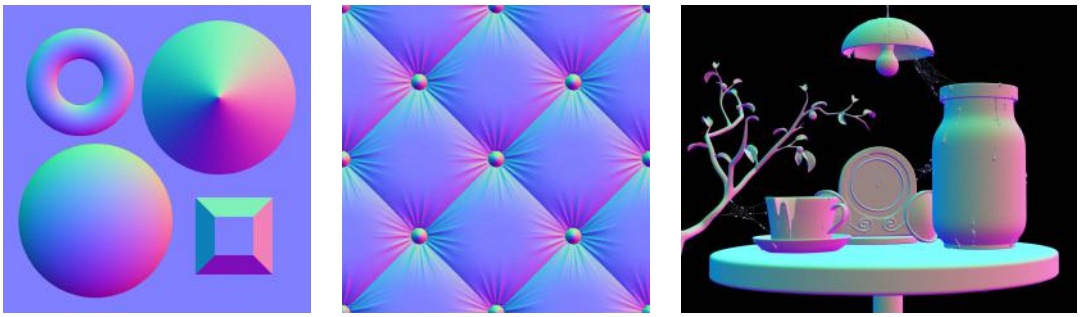
3.2 Creation of Mock-3D Objects

In this section, we will discuss how to create a Shape Image. As we mentioned, the advantage of creating a shape of Mock-3D objects is that we can just use an image to represent the shape. That makes the shape of the object is simple to be changed. As we list the following methods to create a Shape Image, it should also be noted that artists can combine any of these methods, and modify these images through a 2D software such as Gimp or Photoshop.

3.2.1 Rendering Normal Map as Shape Image

One way to create shape map image is that, we can use a traditional 3D rendering software to build the 3D model in the scene and assign it a normal shader based on the camera position. If we take Renderman Shading Language as example, the equation for the normal shader will be:

```
normal N_n = normalize(ntransform('camera', N));
float r = N_n[0]/2+0.5;
float g = N_n[1]/2+0.5;
float b = N_n[2]/2+0.5;
result = (r,g,b);
```

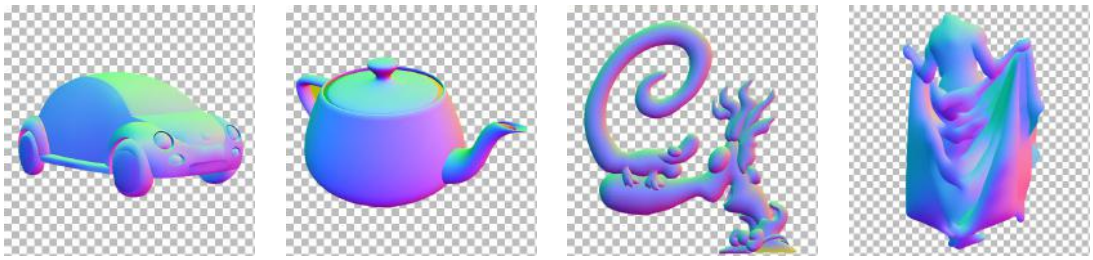


(a) (b) (c)

Figure 3.4: Shape Image create by normal map rendering

3.2.2 Use Existing 2D Modeling Applications

Another way of creating a shape map image is to use a existing 2D modeling tool, such as Lumo [4], CrossShade [5] or Bui's [6] program etc. (See Figure 3.5).



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

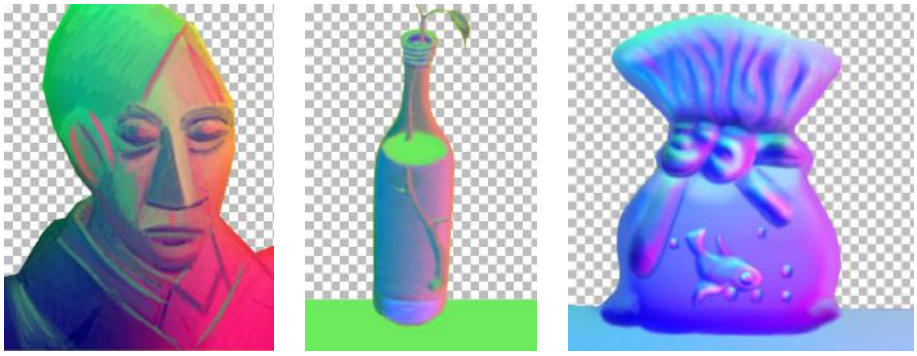
Figure 3.5: Shape Image create by existing software. From left to right: (a) and (b)from CrossShade [5], (c)from Shady (d) from Bui's program [6].

3.2.3 Hand Painting Shape Image

The third way to create a shape map image is that we can manually paint our Shape Image separately through its RGB channels, and then add them together. As explained in section 3.1.1, red & green channels represent the vector field, and blue channel represents the thickness of the object. We can think about the red channel as the result of our object being cast by a red light on the right side, and the green channel as the result of being cast by a green light on the top side. We can draw more blue on the place which you think is thicker than other place. Usually, more blue in the center of a shape, less blue near the contour (see Figure 3.6).

3.2.4 Use Red and Green Lights to Photograph a Shape Image

Another way to create Shape Image is photographing physical objects. According to the concept of the vector field of shape map image we only concern the Red and Green channel of shape map. We can use a red light cast on the right side of an object, and a green light cast on the top of it, then, we photograph the object under the lights (see Figure 3.7).



(a) (b) (c)

Figure 3.6: Shape Image painted by artist

(a) (b) (c)

Figure 3.7: Shape Image photographed by real objects lit in red and green lights

4. WEB-BASED MOCK-3D RENDERING SYSTEM DESIGN

For building a friendly and pragmatic web application, it is critical to develop the web application using user-centered interface design principles. In this chapter, we discuss our approach to optimize the product around how users can, want, or need to use the product rather than forcing the user to change their behavior to accommodate the product [33].

For the processes of a user interface design, we start with the functionality requirement gathering, working on user and task analysis, followed by information architecture and prototyping, along with usability inspection and iterative test, finally focusing on graphical user interface details [34].

4.1 User Goals and Product Requirements

To lead to better design decisions and defining the product's feature set, we set a list of persona [35, 36] and finally focus on 4 types of users:

(1) Illustrators who are technically traditional artists but want to obtain a 3D look and feel for their art. (2) Web and mobile designers who want to obtain high-quality materials and use lights, shadows, and some physical elements to represent the space relationship on their designs. (3) Product designers who love to explore latest trends and get inspiration from new technology. (4) CG lovers who are curious about artistic style shading and want to explore more.

Based on these simplified personas, we identified the following characteristics of our Mock-3D users:

- Mostly designers are artists, or at least interested in aesthetics.
- Most users do not have a skill of 3D software or not clear about the knowledge

of computer graphic.

A good product design first meets user goals according to personas expectation such as user needs to have a tool simple to use, get inspiration. In this project, I have extended Noessels model [36] to design a user interface for Mock3D. This model build over Don Normans concept of Emotional Design [37] by adapting Normans cognition model into the practice of design or user research. The model allows linking user goals with top-level user motivations.

Noessels model present 3 steps of goal: Experience goals, Ends goals and Life goals. Since life goal is considering branding strategy and we are not in the stage of this level. We primarily consider the first two types of user goals in this paper: (1) Experience goals, (2) Ends goals.

Experience goals are the most immediate level of cognitive processing, they express how people want to feel while using a product. In our case, experience goal is to make artists feel it is easy to get a touch on 3D field, and feel cool, have fun when they interact with the product. Therefore, our user interface should be simple, self-explanatory, and the interaction response should be immediate and obvious to see the change.

Ends goals are the middle level of cognitive processing. It leads user gradually understand the product, and let users manage simple everyday behaviors. According to Norman, these constitute the majority of human activity. As a research project, the key for developing Mock-3D is to present our product features such as: able to create 3D work without using complex 3D software, try different artistic styles, able to turn impossible, incoherent or inconsistent 2D artwork into 3D. To let users have an idea about these features. We will set up some preset examples and make tutorial videos in the future, to allow user change images from the selected examples.

Combined with our user experience goal and End goal, we concluded the following requirements when we construct the prototype of Mock-3D.

- Since our major users don't have the experience of 3D software, the first sight of our application should directly present the result in a most simple and understandable way - that will be our primary functions.
- To provide a variety of manipulation opportunity for users, the secondary functions should be hidden, but at the same time we remain links and instruct user to make further explorations.
- Since each artist have different working environments, and it is a creative art tool, our tool is not only used for work but also for pleasure, it need to be fit on the medium screen such as iPad and larger screens. It will be a better to fit on mobile screens, but not necessary. Hence, compatibility through different browsers and a responsive interface design is the key for our technical requirements.

4.2 Mock-3D Prototype

According to the product requirements proposed by user research, we collected our functions and separate them into 3 parts: (1) File import and export (2) Variables/ parameters controls (3) Result area. In the Control Sections, we seperate them to basic controls, and optional controls.

4.2.1 Framework

After iterations of discussion, we presented the prototype of our landing page of Mock-3D website (see Figure 4.1).

We decided to apply a layout that sidebars on the left side and right side with same width and the result output window in the center. Since the scanning path of a

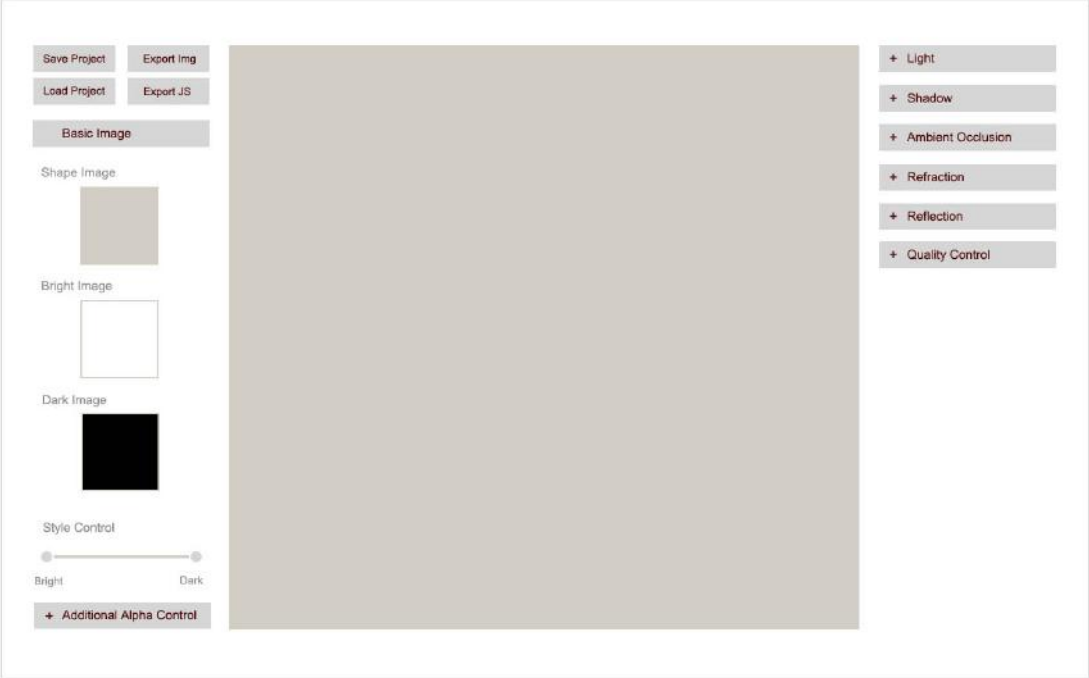


Figure 4.1: Mock-3D prototype

user is always from left to right. The left side includes our primary controls, the right side includes our optional controls. In primary controls part, it has our file import and export section, then followed by basic image control section. In optional controls part, it includes Light, Diffuse alpha, Refraction, Reflection and Fresnel controls.

Since we want to let user understand the feature of our tool, which consist of 3 basic images - Shape Image, Bright Image and Dark Image - as the key images for a simple Mock-3D result. The result is a combination of Bright Image and Dark Image which can tweak the line of light and dark by Style control slidebar based on the shape information provided by Shape Image. Other functions are all optional. Thus, we decided to keep other controls collapsed at the first landing page of our application to make the interface simple and neat, and as the right side is the secondary sight of

a user which implies they are all optional.

4.2.2 Responsive Interface

In order to fit different widths of devices, we need to design a responsive interface to let user get access to Mock-3D in different screens, such as large or medium device desktop, small device tablet, or extra small device phones.

As shown in Figure 4.2, in our Mock-3D case, the key of the responsive layout is the width of sidebars and the size of the result image.

In order to able to see our result in a quite large display even when we use a extra small device, we hide the left sidebar when width of browser is less than 768px (see Phone section). The sidebars width is set to 25% full width of browser, and no larger than 300px when the browser width is larger than 1200px, and not smaller than 160px when it is displayed on phones. Either too long or too short for the width of sidebar will cause the components in sidebars uncomfortable for user to control. The size of result image is based on the aspect ratio of Shape Image. It has the same aspect ratio as shape image, but display as large as possilbe and always leave a 10px spacing to the left and right, or to the top and bottom. Each component's width and height will be recalculated when the browser is resized.

4.3 Component Design

For a better navigation experience and a clear hierarchy presentation, we want to only expand one of our optional controls section at a time. That makes us choose to use "accordion" component. Figure 4.3 shows the application of accordion in multiple lights under light control section. Apart from this "accordion" component, the left sidebar and right sidebar individually using an accordion component to arrange different control sections.

According to the control of function requirements such as image replacement,

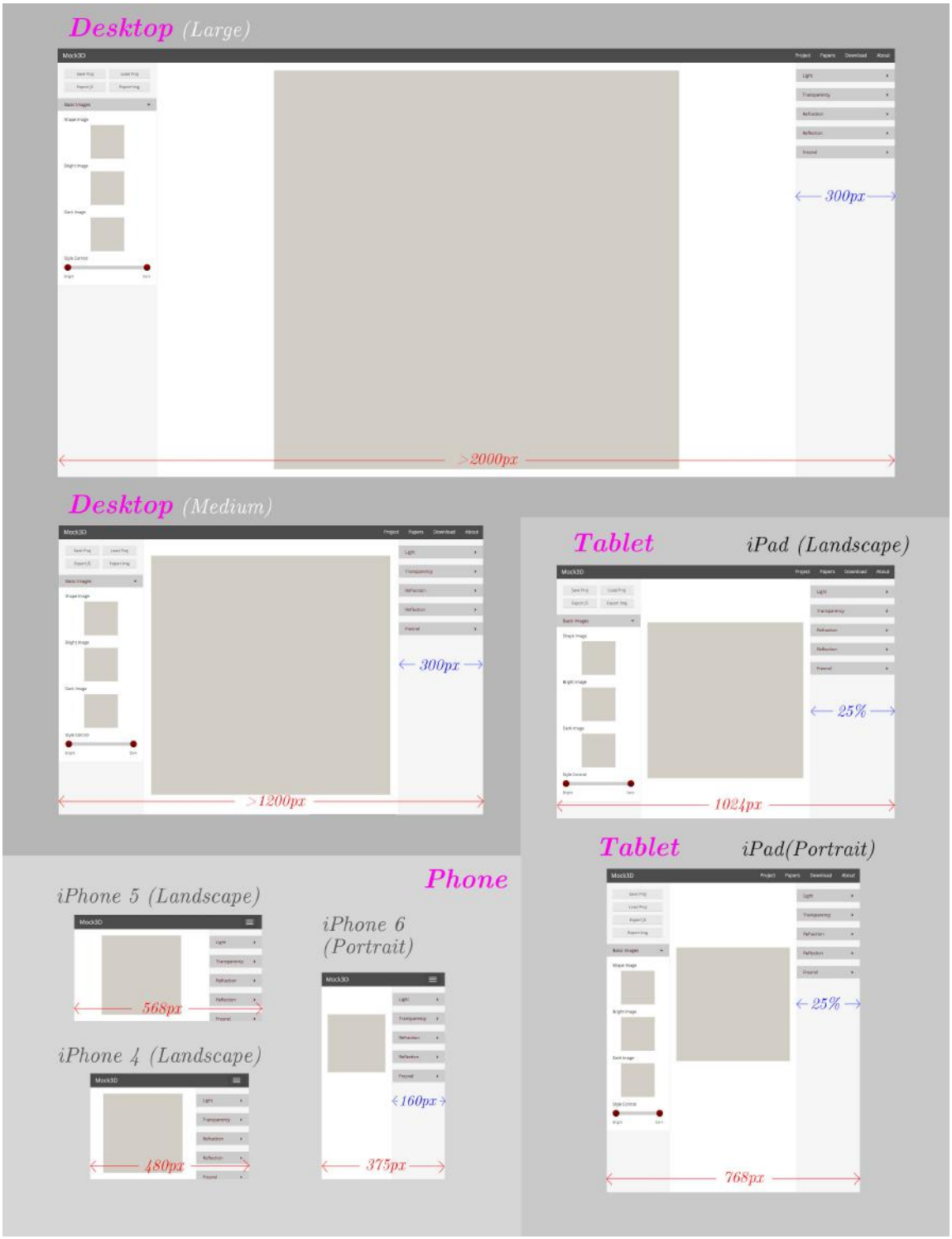


Figure 4.2: Style Control slidebar and corresponding results

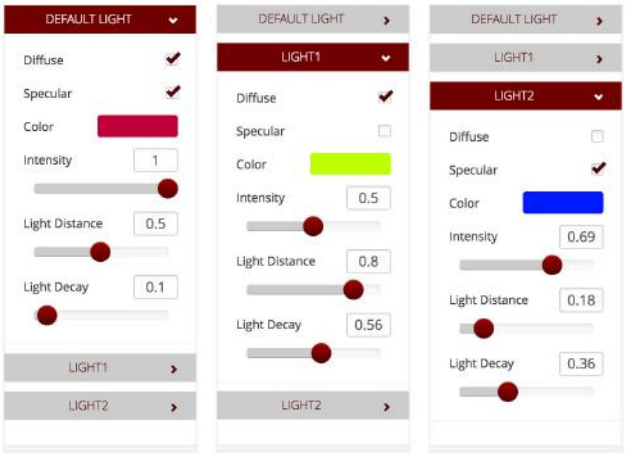


Figure 4.3: Expand different item in multiple lights accordion component

parameter tweaking, functions turn on/off, method selection, and in order to keep the consistent experience for user, we use the following five most commonly used template components in websites to control parameters: (1) Image thumbnails with unloading function, (2) Checkboxes, (3) Slidebars, (4) Dropdown Menus, (5) Color Picker. We show these four types of template components in Figure 4.4.

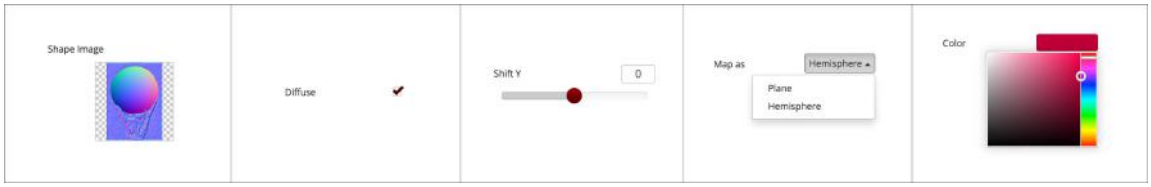


Figure 4.4: Examples of five types of components (From left to right: image thumbnail, checkbox, slidebar, dropdown menus and color picker)

We, now, want to discuss two of these components and their features in detail: Thumbnail and the slidebar.

Thumbnail: The thumbnail is not only provides a thumbnail view of the uploading image but also able to upload an image by clicking on the image or draging image format file in it. All the thumb images will show the actual aspect ratio of the original image user uploaded. However, since the result rendering is based on the Shape Image, if other images (Dark, Bright, Additional Alpha, Background, Foreground Image) provided by users is not the same aspect ratio as the Shape Image, these images will be automatically stretched or shrank to the same aspect ratio of Shape Image.

Slidebar: Each slidebar allows to control the value of a parameter in a given range of numbers using a handle. We set an appropriate range of the parameter to let users get a reasonable result even when the handle is moved to one of the two ends of the slider. The slider controls also provide an input text area to directly control the values of the parameters. Using this input text area, users can set values beyond the given range of maximum and minumum values.

5. RENDERING AND COMPOSITING WITH MOCK-3D OBJECTS

In this chapter, we will introduce the general principles of qualitatively acceptable rendering of Mock-3D shapes.

To achieve artistic rendering purpose with more understandable changing associated to parameters, we provide different shading algorithms differ from traditional shading method.

As shown in Figure 5.1, the workflow of Mock-3D rendering and compositing system mainly has 5 input images. Left side, started with Bright, Dark, Shape Image, is the workflow to generate a diffuse result. The right side is the workflow of generation of Reflection result and refraction result. The combination of reflection and refraction base on a Fresnel equation according to the material of the object. Then, an appearance result comes from the mix of Diffuse result and Fresnel result base on the alpha channel of the result of diffuse. Finally, we use the alpha channel of Shape Image to composite appearance with Background Image.

The red arrow in Figure 5.1 and 5.2 pointing to the result of diffuse alpha stand for the percentage of Diffuse in the process of combining with Fresnel, it comes from the alpha channel in both Bright Image and Dark Image. It can be considered as the transparent part of the Mock-3D object.

We also provided an additional Alpha Control Image to provide the convenience of adjusting alpha of diffuse result. The workflow will not change if we don't adjust the slider bars in Diffuse Alpha section. If we want to use this control, the workflow of Mock-3D will add a process on the diffuse alpha channel before generating appearance result, then continue the workflow. By using this additional Alpha Image, We don't need to set some part of our Bright and Dark Image transparency. Instead, we can

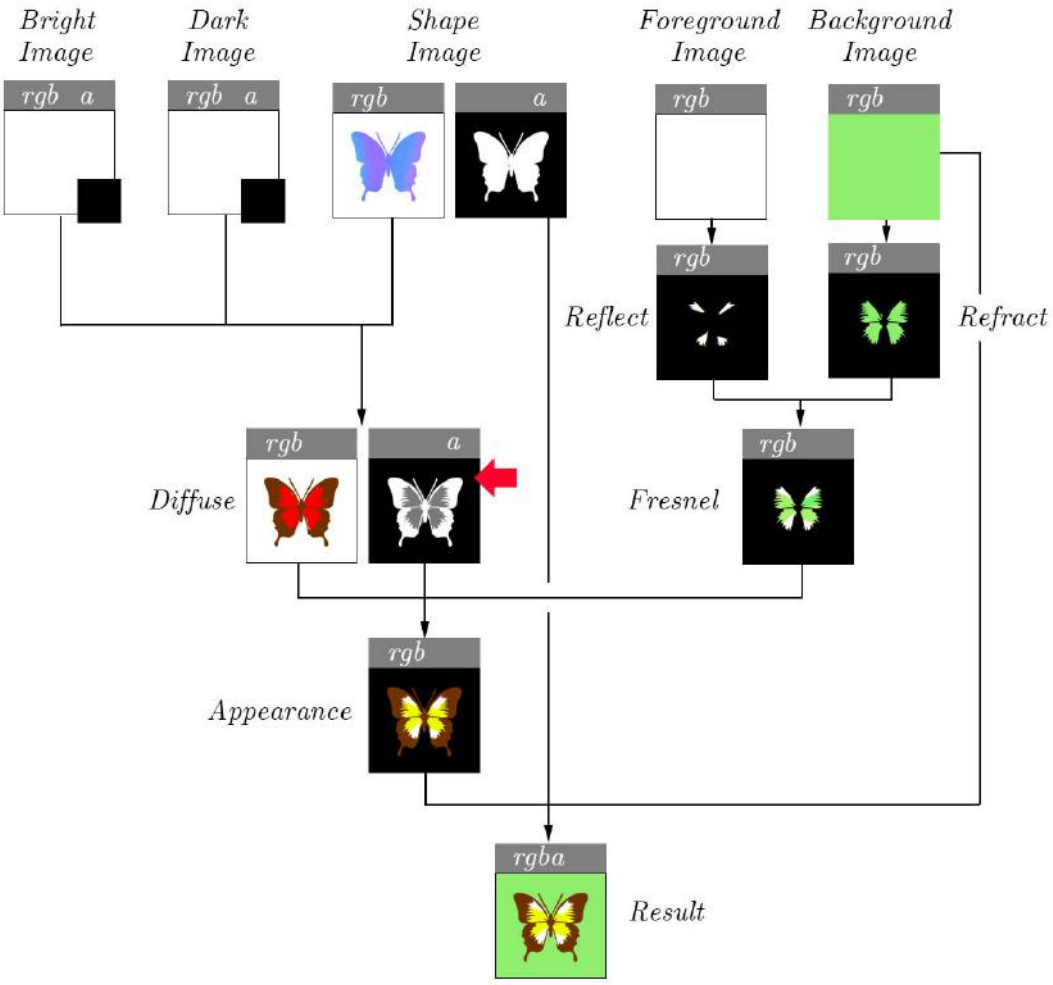


Figure 5.1: Workflow of Mock-3D system

use slidebars to intuitively control how transparency a certain part in the mock-3D object should be.

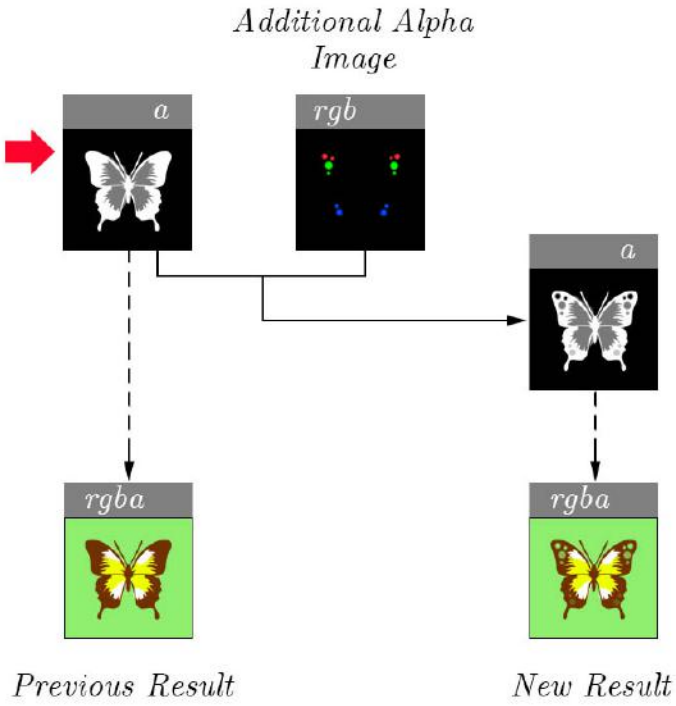
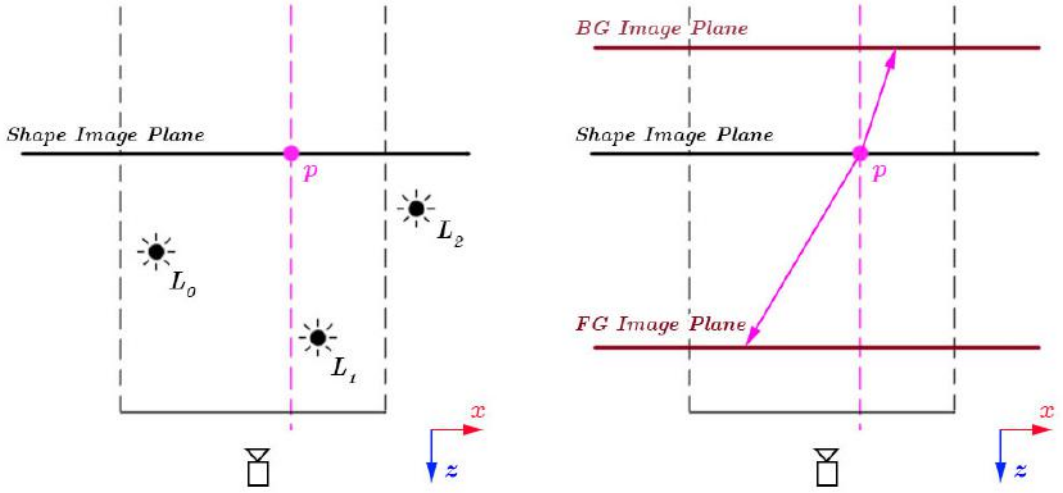


Figure 5.2: Additional alpha control in the workflow of Mock-3D system

5.1 Rendering

Rendering scene in Mock-3D includes lights, shading method (our style control), reflection, refraction and Fresnel combination. As we introduced in chapter three, we use orthographic view in rendering. Mock-3D Shape Image is on the plane $z = 0$ and the image center is on the coordinate origin. Figure 5.3 shows the top view of our Mock-3D scene.

As shown in Figure 5.3(a), to calcualte pixel (x, y) in the final result image, we need to calculate the point $p(x, y, 0)$ in the Mock-3D coordinate system. Lights L_0, L_1, L_2 is in the space of $z > 0$. The diffuse shading can be received from those lights' positions and the according (x, y) information from Bright, Dark, and Shape Images. Figure 5.3(b) shows a ray traces from point p on the Shape Image plane to the Foreground (FG) Image plane to get the reflection information and trace to Background (BG) Image plane to get the refraction information. In which FG Image is on the plane of $z = a$ $(a > 0)$. BG Image is on the plane of $z = b$ $(b < 0)$, in which a and b can be adjusted by users.



(a) (b)

Figure 5.3: Mock-3D coordinate top view. (a) shows camera and lights position, (b) shows reflection and refraction rays trace from a position p on Shape Image plane to the Foreground Image plane and Background Image plane.

5.1.1 Light and Style

In the Mock-3D system, shading result (style) is based on light's properties. By tweaking the parameters of lights, we can create different art styles of result. At the same time, we also provided a style control slidebar to intuitively control the shading of our Mock-3D objects.

5.1.1.1 Light Type and Properties

In the Mock-3D system, light can be considered as a brush to add color variety on the result. Take the brush tool in Adobe Photoshop as an example, since different types of brush can present different types of artist styles, we have options to choose brush style from hard edge to soft edge, it can also have different sizes. Therefore, we use this idea, and choose to use pointlight instead of directional light to have "size" and "style" controls of our light brush. The distance from the light to the canvas is used to control the "size" of the light, the intensity and decay of light controls are similar to the opacity of a brush in Photoshop. Furthermore, we have our style control slidebar for the soft or hard edge control.

Lights in Mock-3D system are represented by mouse position. In order to apply on different aspect ratios of images, the x, and y of our normalized mouse position $p_{Mn}(X_{Mn}, Y_{Mn})$ on the Mock-3D coordinate can be calculated as following:

$$X_{Mn} = \frac{X_M - X_0}{X_1 - X_0} - 0.5$$

$$Y_{Mn} = -\frac{Y_M - Y_0}{Y_1 - Y_0} + 0.5$$

As shown in Figure 5.4, $p_M(X_M, Y_M)$ is the mouse position on the browser,

$p_0(X_0, Y_0)$ is the left top position of canvas on the browser, $p_1(X_1, Y_1)$ is the right bottom position of canvas the on browser.

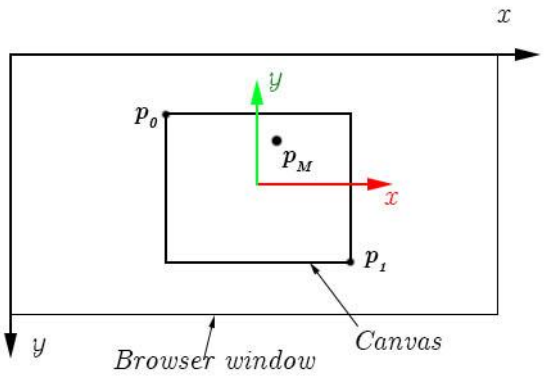


Figure 5.4: Mouse and canvas position on browser coordinate (in black). The origin of Mock-3D coordinate (in color) is on the center of canvas.

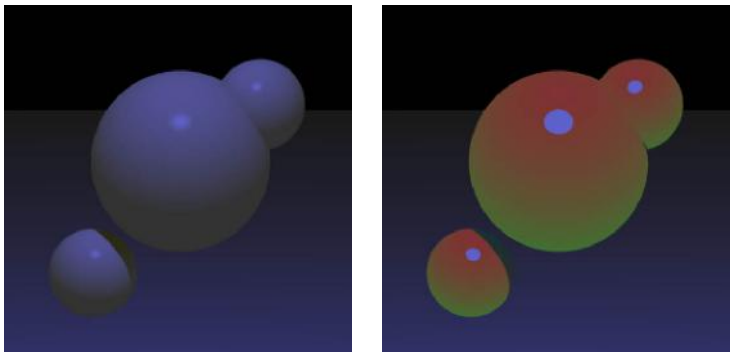
Considering when artists move their brush, the brush size should not be changed, therefore, we set our pointlight position on a plane d away from the canvas. Since our shape image is on the plane of $z = 0$ in the Mock-3D coordinate system. Thus, our pointlight position p_L will be:

$$p_L = (X_{Mn}, Y_{Mn}, d)$$

5.1.1.2 Style

We noticed that the significant difference from a photorealistic rendering is that, non-photorealistic rendering (NPR) has a comparatively more saturated color in the shadow (Figure 5.5). For the artistic purpose, we hope the shadow shade can be given directly by artists.

Therefore, we came up with the idea of using two images, a fully illuminated image, and an unlit image to represent the shades of objects. That becomes our Bright Image and Dark Image. The diffuse result is an equation derived from $RGB\alpha$ channel from both Bright and Dark Image.



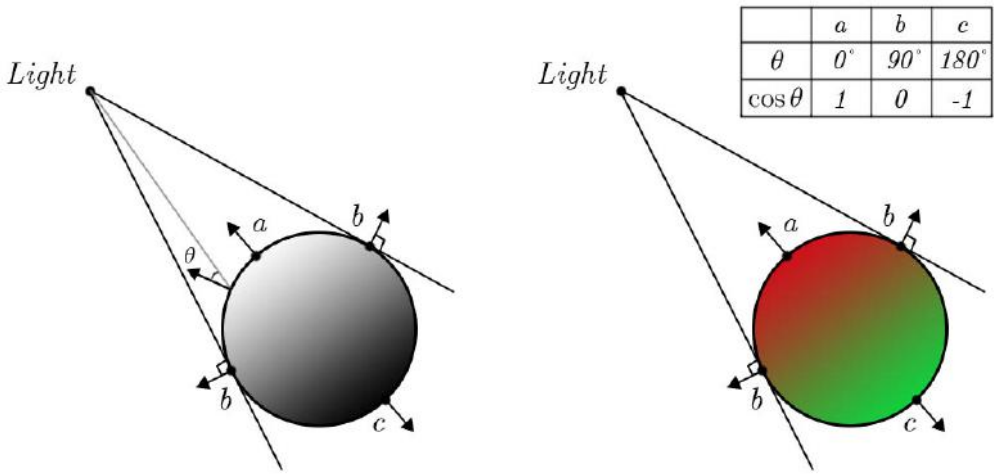
(a) Photorealistic Rendering (b) NPR Rendering

Figure 5.5: Shadow comparison of Photorealistic rendering and NPR

As shown in Figure 5.6, we denoted θ as the degree between the normal vector of a point on a sphere and the ray from the point to the light position. The $\cos\theta$ from the position a (lighted part) on the sphere to b (half lighted part), to c (shadow part) equal to 1, 0, and -1 respectively. We can use an linear interpolation to achieve a gradient from one color C_1 in the light to another color C_0 in the dark. Then, the shading equation of result color C will be:

$$C = C_0(1 - t) + C_1 t \tag{5.1}$$

Where t is between 0 to 1. This formulation closely resemble to Gooch shading formulation [23]. Note that since $\cos\theta$ is in this range for lights in front of object,

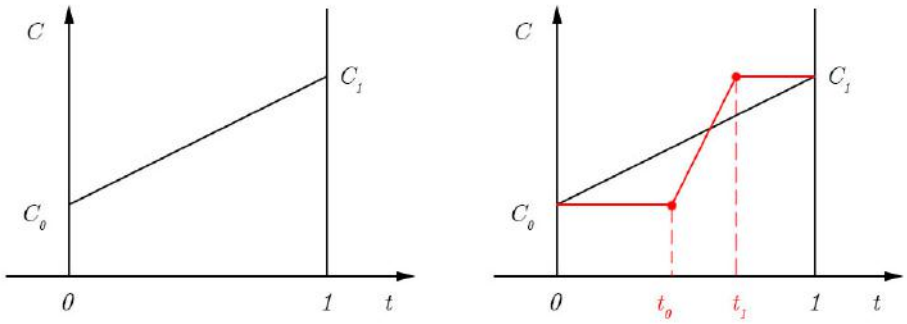


	a	b	c
θ	0°	90°	180°
cos θ	1	0	-1

(a) (b)

Figure 5.6: The relationship between $\cos\theta$ on position a, b and c and the shade result on them.

we can directly use it as our t parameter. Diagram in 5.7(a) shows the relationship between t and C given in Equation 5.1.



(a) (b)

Figure 5.7: Artistic shading equation in diagrams. (b) add t_0 and t_1 parameters to provide flexiable controls to blend two color C_0 and C_1

.

Cartoon shader can be considered as the flatten combination of C_0 and C_1, in order to have a flexible control of the gradient step point of C_0 and C_1, we introduce two parameters t_0 and t_1 in between 0 to 1 (see Figure 5.7(b)), so that, the two parameters turn into two handles of the slidebar in our interface. Figure 5.8 demonstrates how the shading of a sphere shape is controlled by our slidebar.

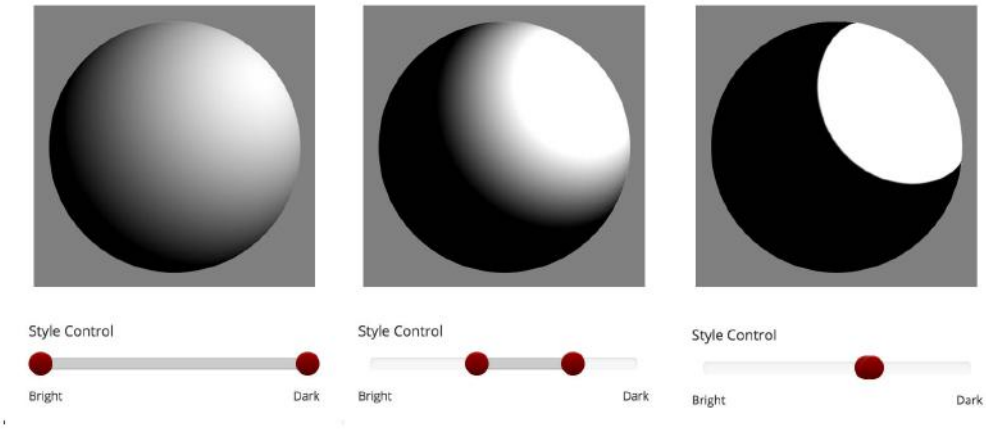


Figure 5.8: Results of tweaking style control slidebar with two handles

Since C_0 and C_1 can be replaced by two images, we use I_0 and I_1 to represent them, that is our Dark Image and Bright Image. Then, our diffuse result I_D will be:

$$I_D = I_0 t' + I_1(1 - t')E_L$$

Where,

$$T' = \frac{t - t_0}{t_1 - t_0}$$

$$t' = \begin{cases} 0, & \text{If } T' < 0 \\ 1, & \text{If } T' > 1 \\ T', & \text{Otherwise} \end{cases}$$

E_L is the energy of a incident light and in Mock-3D system, it is computed from light properties such as: light color, light intensity, light plane distance and light decay. Here, we simplified it as the multiplication of the intensity of a light I_L and the color of a light C_L. Thus, the energy of incident light is: $E_L = I_L C_L$.

5.1.1.3 Multiple Lights

Lights have two functions in Mock-3D system: (1) For illumination fuction (2) For creative manipulation - lights work as brushes. In order to implement the second function, we provided multiple lights with independent light properties. In this section, we will talk about how to apply them together on the result.

Derived from one light diffuse equation in the previous section 5.1.1.2, let T denote the overall contribution of lights, it can be calculated by the sum of each light's energy multiplied by $(1 - t')$ as the following:

$$T = \sum_{i=0}^{n} ((1 - t')E_L)$$

Then, our diffuse result I_D of multiple lights will be:

$$I_D = I_0 T + I_1(1 - T)$$

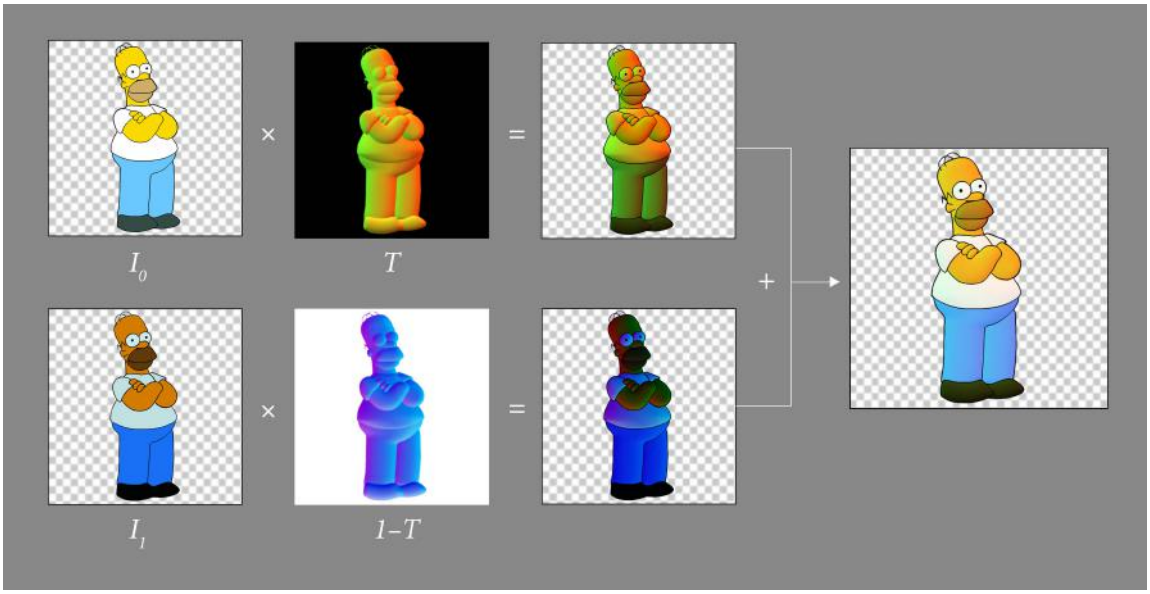


Figure 5.9: Multiple lights equation demonstrated in images. Artwork from a cartoon character of Homer Simpson

The equation can be visually demonstrated as in Figure 5.9, where $1 - T$ is the reverse of the image of T. And T can be represented as the rendering diffuse result of multiple lights cast on a white shape of the character.

5.1.2 Refraction

We observe that index of refraction in nature is mostly in the range from 1/2 to 2. To achieve a linear interpolation, we set the parameter $\log_2 \eta$ as the power of 2, so that our index of refraction parameter on the interface ranges from -1 to 1. Noted that when $\log_2 \eta = 0$, which is in the middle position of our slidebar, the index of refraction is 2 to the 0, that equals to 1 which is the index of refraction of the water medium.

As shown in Figure 5.15, when a refraction ray l transits to a different medium, the ray will change direction. If the index of refraction of boundary η is larger than

1, that is to say, $log_2\eta$ is larger than 0, its direction will be more inclined to the normal vector line of that point. Vice versa, if the index of refraction of boundary η is smaller than 1, $log_2\eta$ is smaller than 0, it will be inclined to the surface plane.

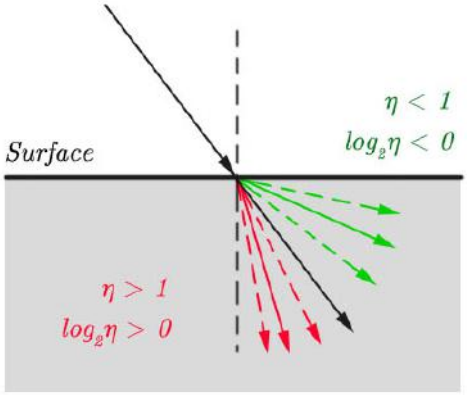
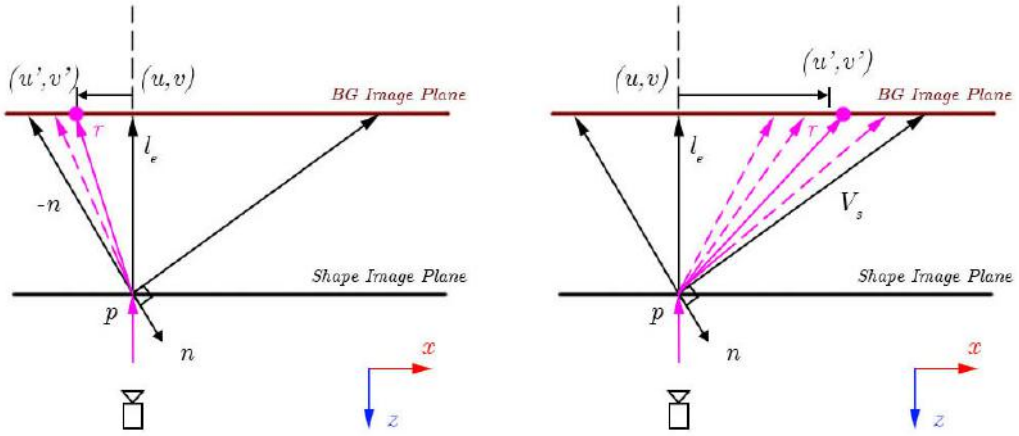


Figure 5.10: Refraction ray changes direction when it transits to a different medium.

According to the principle before, as shown in Figure 5.11, we can calculate our refraction ray r by interpolating three rays: (1) l_e the camera ray which always is $(0, 0, -1)$, (2) $-n$ the opposite direction of the normal vector of current pixel position: P, and (3) V_s which is on the same plane of $-n$ and l_e, and perpendicular to $-n$. It can be calculated as following:

$$V_s = -n \times l_e \times -n$$

When $\log_2 \eta$ is in the range of -1 to 0, refraction ray can be interpolated by ray (1) and ray (2) (see Figure 5.11 (a)), when $\log_2 \eta$ is in the range of 0 to 1, it can be interpolated by ray (1) and ray (3) (see Figure 5.11 (b)). Therefore, the refraction



(a) (b)

Figure 5.11: Refraction ray (in pink) trace from position p on shape Image to the Background Image plane

ray r equation will be:

$$r = \begin{cases} l_e(1 - \log_2 \eta) + (-n)\log_2 \eta, & \text{If} \quad \log_2 \eta > 0 \\ l_e(1 - (-\log_2 \eta)) + V_s(-\log_2 \eta), & \text{Otherwise} \end{cases}$$

As shown in Figure 5.12, let d denote the distance in z direction from our Background Image plane to the Shape Image plane. Then the shift unit vector in x and y direction of the refract ray $r = (x_r, y_r, z_r)$ is $\frac{(x_r, y_r)}{z_r}$. Therefore, the shift vector $\overline{V}$ of UV position of our background image can be calculated as following:

$$\overline{V} = (x_r, y_r)\frac{d}{z_r}$$

Examples are shown in Figure 5.13. According to different $log_2\eta$, the results of setting the value of refraction slidebar larger than 0 makes the Background Image

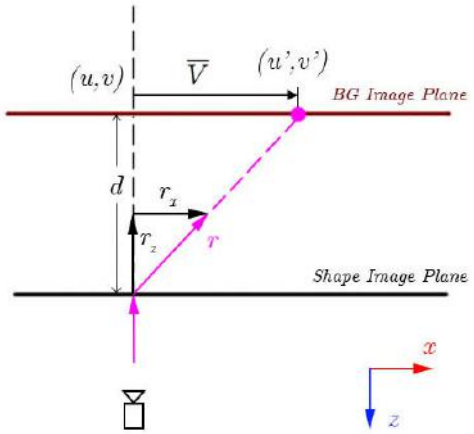


Figure 5.12: Refraction ray cause a shift of detected UV position of Background Image

expand through the bottle. Vice versa, Background Image looks shrunken when the value is set smaller than 0. In our application, we use the term "refraction" to stand for $\log_2 \eta$.

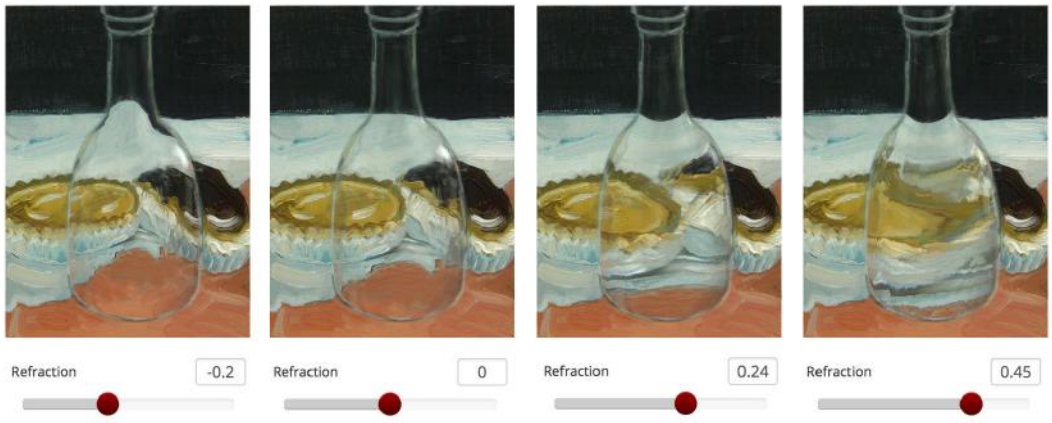
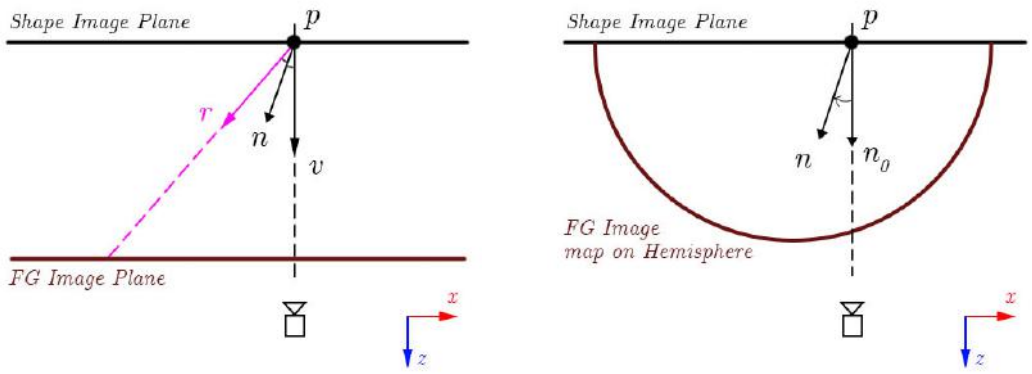


Figure 5.13: Results of seting refraction slidebar under refraction section to different values in Mock-3D application affects on a bottle shape. The artwork is modified by an painting from Alison Mackay [7].

5.1.3 Reflection

For our reflection rendering, we have two options for users. (1) Take the reflection Foreground Image as a plane, set it parallel to the Shape Image plane and a certain distance away (see Figure 5.14 (a)). (2) Map the reflection Foreground Image on an environment hemisphere (see Figure 5.14 (b)).



(a)Mapping Foreground Image on a Plane (b)Mapping Foreground Image on a Hemisphere

Figure 5.14: Two different methods of mapping Foreground Image

5.1.3.1 Mapping Foreground Image on an Ultimate Plane

Let n denote the normal vector of a point on our Shape Image plane, v is the normalized vector from the point to eye/camera (it points to positive direction of axis z in Mock-3D system). As shown in Figure 5.15, reflection vector r can be represented as: $r = -v + 2(n \cdot v)n$.

let d denote the distance in z direction from our Foreground Image plane to the Shape Image plane. Then the shift unit vector in x and y direction of the reflect ray $r = (x_r, y_r, z_r)$ is $\frac{(x_r, y_r)}{z_r}$. Therefore, the shift vector $\overline{V}$ of UV position of our

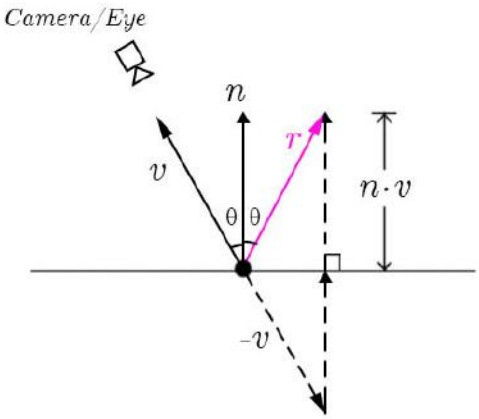


Figure 5.15: Calculate reflection ray r by coming ray incidence $-v$ and the normal vector n

Foreground Image will be: $\overline{V} = (x_r, y_r)\frac{d}{z_r}$.

To increase the interactive experience of reflection with the mouse, we add an additional mouse position p_{Mn} in the end. Therefore, our detected UV position (u', v') of our Foreground Image can be calculated as following:

$$(u', v') = (u, v) + (x_r, y_r)\frac{d}{x_z} + p_{Mn}$$

Where (u, v) is our original UV position of the Shape Image.

5.1.3.2 Mapping Foreground Image as Hemisphere

Another mapping method is to map our Foreground Image on a hemisphere. It works similar as adopting an environment sphere. In our case we only consider the front part of the sphere which is before our Shape Image plane (see Figure 5.14(b)).

The UV cordinate on the environment sphere depends on the normal vector of

the calculated point. Consider that if there isn't a change of the normal vector, which comes from Shape Image, the detected UV position of Foreground Image is supposed to be exactly the same position as the UV position on Shape Image. Since the Shape Image provides a normal direction changing, the shift vector of our detected UV position of our Foreground Image can simply use the x and y direction of the normal vector $n = (x_n, y_n, z_n)$ to represent, then our detected UV position (u', v') of Foreground Image can be calculated as following:

$$(u', v') = (u, v) + (x_n, y_n)\frac{1}{z_n}$$

Where (u, v) is our original UV position of the Shape Image.

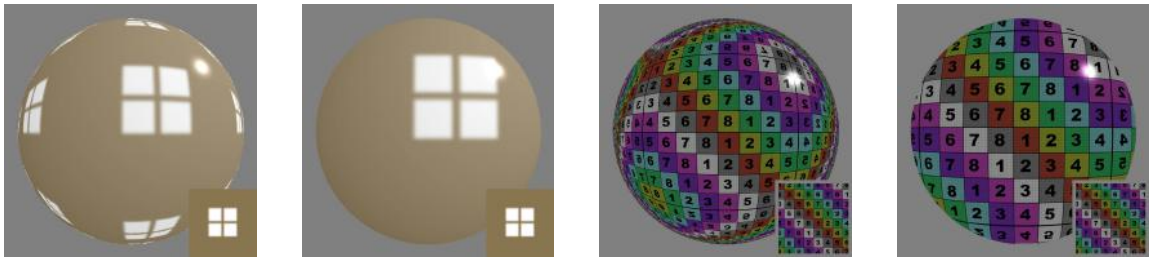
5.1.3.3 Comparison of Two Mapping Methods

As shown in Figure 5.16, Plane mapping method and Hemisphere mapping method have comparatively different results. When we use Plane mapping method, we repeat the pattern if reflect ray traces out of the boundary of Foreground Image, so the result repeats the window pattern as shown in Figure 5.16(a). In addition, the Plane mapping method in the window pattern case looks more distorted, especially on the edge of the shape. Similar with the result in the case of applying a checker as Foreground Image, Plane mapping reflection looks more distorted.

Another example shows the benefits of adopting Hemisphere mapping method. As shown in Figure 5.17, Plane mapping results to much distortion and finally loses the legiable shape of relection image in the eye. However, Hemisphere mapping can properly provide a clear silhouette of buildings which meet the needs of user to represent a meaningful reflection in this case.

In sum, both Plane and Hemisphere mapping methods have pros and cons, we provided a dropdown selection widget in Mock-3D interface for users to choose an

appropriate reflection mapping method in different situations.



(a)Plane mapping (b)Hemisphere mapping (c)Plane mapping (d)Hemisphere mapping

Figure 5.16: Results of two mapping methods on a sphere shape using different Foreground Image as shown in the right bottom thumbnail

(a)Plane mapping (b)Hemisphere mapping

Figure 5.17: Results of two mapping methods on an eye shape using Foreground Image as shown in the right bottom thumbnail

5.1.4 Fresnel

The Fresnel equation describes how to combine reflection and refraction terms when the light moves between media of different refractive indices [38].

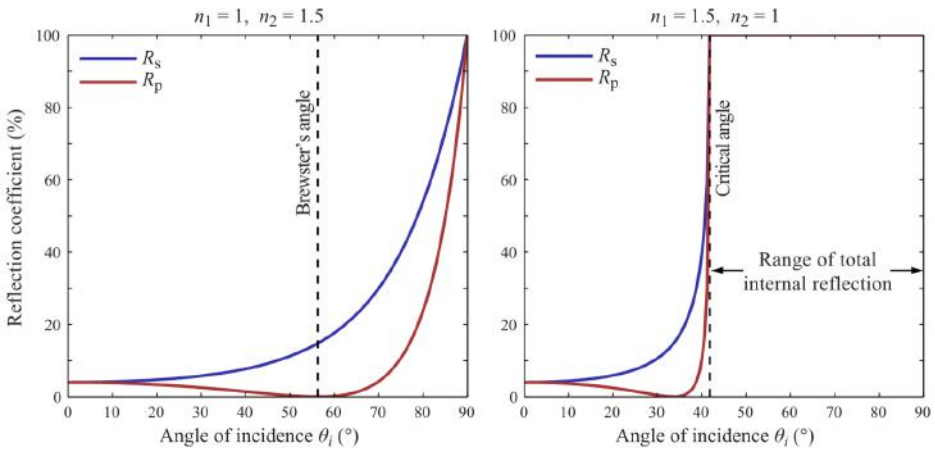


Figure 5.18: Two examples of reflection cooefficient curve of different medium according to the angle of incidence [8]

As shown in Figure 5.18, when light incident transmits from one medium to another medium, the reflection percentage changes according to the angle of incidence. We also notice that different index of refraction boundary has different shape of reflection coefficient curve, but in general, they look visually similar. So, it is possible to use this visual similarity to simplify the Fresnel curve into piecewise linear function as shown in Figure 5.19.

To make the simplification, we selected four points on the curve and piecewise-linearly connected those points as shown in Figure 5.19. Therefore, our reflection coefficient curve becomes three linear equations on three range of the angle of incidence from 0° to 90°. The four points are: when angle of incidence θ equals 0° (θ_a), when θ equals Brewster's angle (θ_b), when θ equals 90° (θ_d), and finally we find a random angle between θ_b and θ_d (θ_c).

Let's call Reflection coefficient in Figure 5.19 as Fresnel value(F) with the range between 0 and 1 on y axis as shown in Figure 5.20, and we use $\cos\theta$ as x axis instead

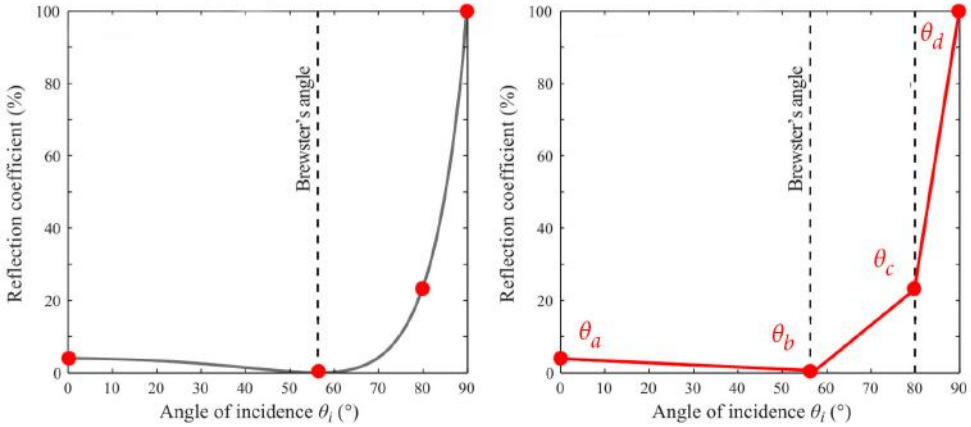


Figure 5.19: Mock-3D reflection cooefficient / Fresnel curve

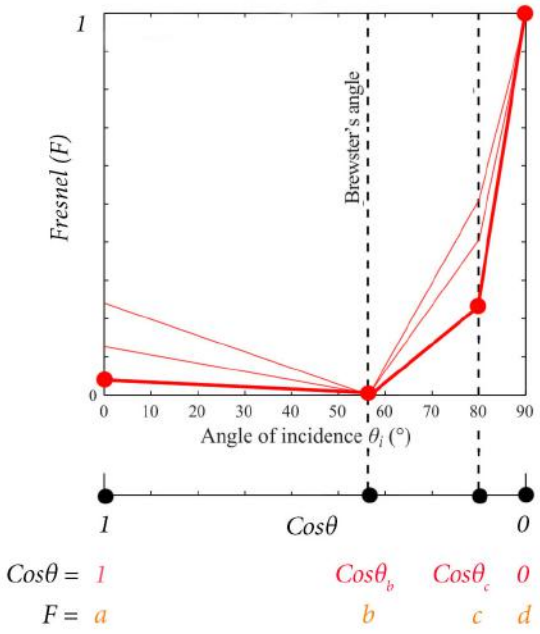


Figure 5.20: Fresnel Curve according to $\cos\theta$

of Angle of incidence. Therefore, the value of $\cos\theta$ of θ_a, θ_b, θ_c, θ_d are 1, $\cos\theta_b$, $\cos\theta_c$ and 0 respectively. The according Fresnel values equal to a, b, c, and d.

Since there exist diverse Fresnel Curves based on index of refraction η, we should use η to interpolate them. Let's give a minium and a maximum reflection precentage values of a and c while b and d stay on value 0, and value 1. Furthermore, in refraction rendering method, we already got $\log_2\eta$ with the range from -1 to 1, now, we normalize it as N:

$$N = \frac{(\log_2\eta + 1)}{2}$$

Therefore, our a, b, c and d values are:

$$a = a_{min}(1 - N) + a_{max}N$$

$$b = 0$$

$$c = c_{min}(1 - N) + c_{max}N$$

$$d = 1$$

So, in order to calculate the Fresnel value of $\cos\theta_i$, we have our pseudo Fresnel equation F as following:

if $1 > \cos\theta_i$ & $\cos\theta_i > \cos\theta_b$ **then**

$t \leftarrow (\cos\theta_i - \cos\theta_b)/(1 - \cos\theta_b)$

$F \leftarrow a * t + b * (1 - t)$

else if $b > \cos\theta_i$ & $\cos\theta_i > \cos\theta_c$ **then**

$t \leftarrow (\cos\theta_i - \cos\theta_c)/(\cos\theta_b - \cos\theta_c)$

$F \leftarrow b * t + c * (1 - t)$

else if $\cos\theta_c > \cos\theta_i$ & $\cos\theta_i > 0$ **then**

$t \leftarrow (\cos\theta_i - 0)/(\cos\theta_c - 0)$

$F \leftarrow c * t + d * (1 - t)$

end if

In Mock-3D system, we provide two parameters, $\sin_b$ and $\sin_c$, for a custom adjustment of Fresnel equation since adapting sin instead of cos can comparatively achieve a linear impact of the interactive result.

We can further prove that using $\sin\theta$ can better demostrate Fresnel Curve and give users a visualized and intuitive control. As shown in 5.21, a top view of getting $\cos\theta$ and $\sin\theta$ values of different position of a cylinder shape, it also show the according results and Fresnel Curve. From the cylinder center to its edge, $\sin\theta$ changes from 0 to 1, $\sin\theta_b$ and $\sin\theta_c$ equals X_b and X_c in the figure, which can visually demonstrate how Fresnel Position slider parameters work.

As shown in Figure 5.22, by adjusting the slidebar of Fresnel Position Control, we can get a variety of Fresnel rendering results.

To provide more various controls of the combination of refraction and reflection, we give a Fresnel Control slidebar to transit Fresnel result to full reflection, or full refraction (see Figure 5.23).

We denote N_F as the interpolate value, then we have our new Fresnel equation:

$$F' = \begin{cases} N_F + F(1 - N_F), & \text{If } \ N_F > 0 \\ F(1 + N_F), & \text{Otherwise} \end{cases}$$

As shown in Figure 5.24, Fresnel Control slidebar provides more flexible fresnel rendering results. With the combined use of both Fresnel Position and Fresnel Control slidebar, we can get nearly all the possibilities of a plausible Fresnel curve.

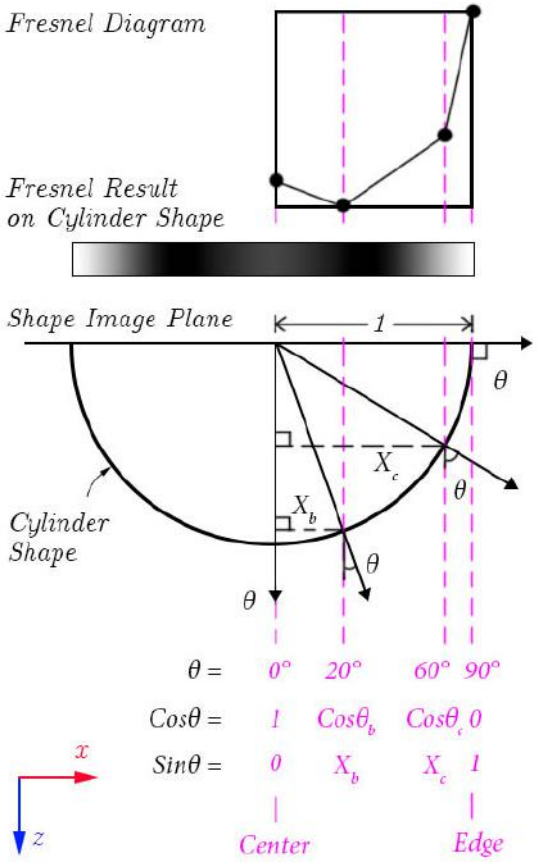


Figure 5.21: From bottom to top: Mock-3D coordinate top view on a cylinder cross section, according Fresnel Result with white as reflection and black as refraction, according Fresnel curve

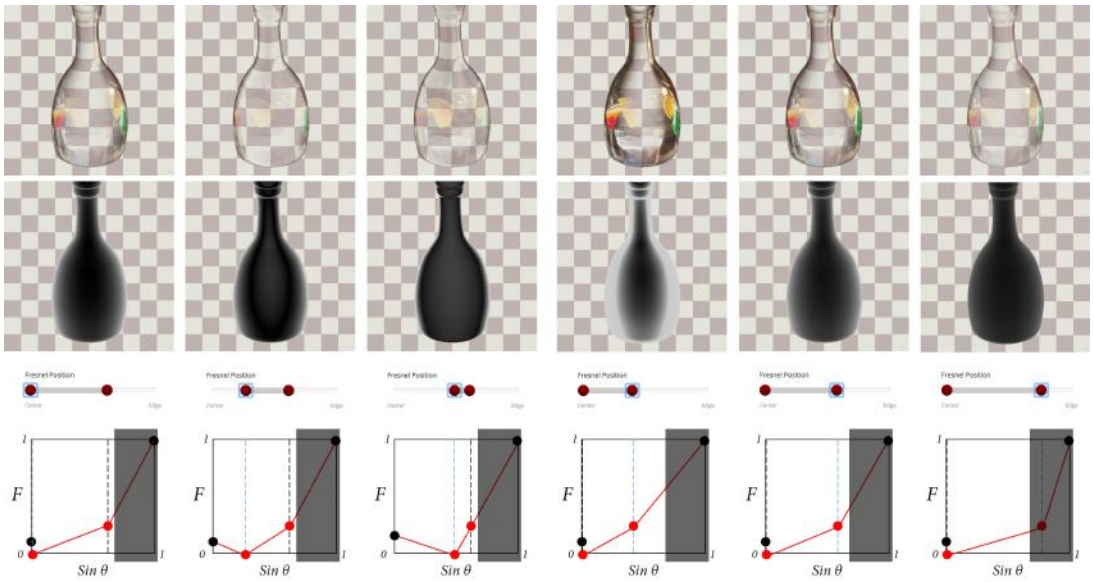
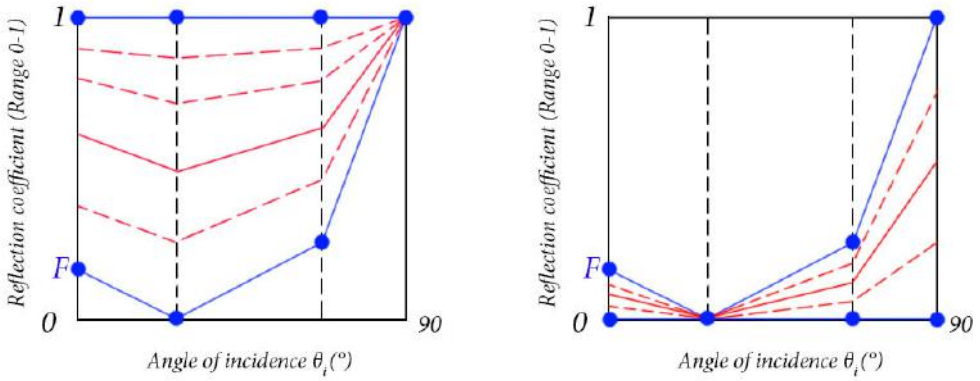


Figure 5.22: Fresnel Results on a bottle shape when user move two handles on Fresnel Position slidebar. The first row shows Fresnel Results. The Second row shows Fresnel matte, in which white part represtent reflection and black part represent refraction. The third row and the last row show the according Fresnel Position slidebar and Fresnel curves.



(a) (b)

Figure 5.23: Interpolations between Fresnel Curve and Full Reflection in (a), between Fresnel Curve and Full Refraction in (b)

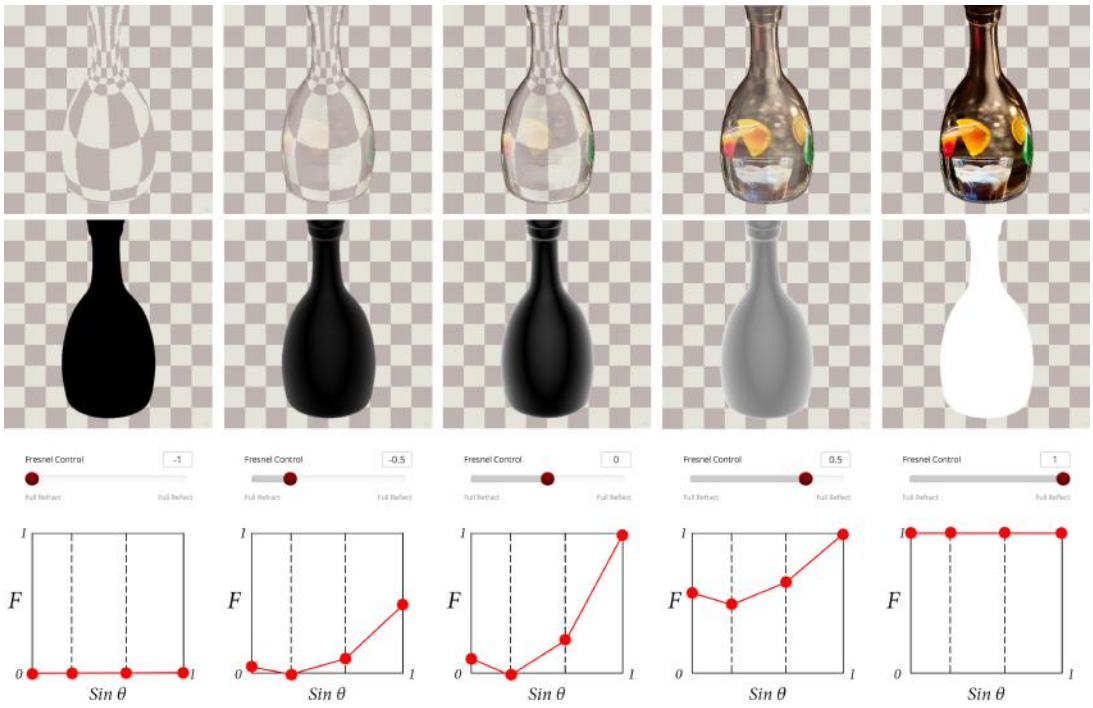


Figure 5.24: Fresnel Results on a bottle shape when user move the handle on Fresnel Control slidebar. The middle result show the default Fresnel curve, the left one is full refraction result and the right one is full reflection result. Others are the interpolation results.

5.2 Compositing

After we get the result of differenct material properties, the final process is to composite them together. In this section, we mainly talk about the compositing of Diffuse and Fresnel result using Diffuse alpha channel to create our Appearance result, which is then, composite with Background Image. After that, we will talk about additional diffuse alpha control.

5.2.1 Compositing Equation Using Diffuse Alpha Channel

There are two compositing processes in the workflow of Mock-3D system (see Figure 5.1 at the beginning of this chapter). (1) The compositing of Diffuse result and Fresnel Result and (2) The compositing of Apprearence result and Backgroung Image.

The parameter we use to composite them are Diffuse alpha channel in (1) and Shape Image alpha channel in (2).

The Diffuse alpha channel is derived from both Bright Image and Dark Image, which is transparent in some area to represent a partially transparent object. Let a_D denote the Diffuse alpha. We already got Fresnel result I_F, which mixes the reflection and refraction results, in previous sections. So our appearance result I_A in compositing (1) is:

$$I_A = I_F(1 - a_D) + I_D a_D$$

After the calcuation of the apprearence equation, in compositing (2), we composite our appearance with the Background Image I_{BG} using the alpha channel from Shape Image a_S. Then, our final compositing image I_C will be:

$$I_C = I_{BG}(1 - a_S) + I_A a_S$$

5.2.2 Additional Diffuse Alpha Control

To have a better control of the transparent part of Mock-3D objects for users, we propose an additional Diffuse Alpha control section.

If users want to activate this function, they need to provide a Diffuse Alpha Image in color. By setting different values of transparency in Red, Green and Blue in slidebars, users can control different areas with different transparenty respectively, which can be up to three areas (R,G,B).

This additional alpha control will add upon on the original Diffuse alpha after the Diffuse alpha is calculated by Bright Image and Dark Image. Let's denote a_{D_0} as the calculated alpha value of Diffuse result. Then our new alpha value of Diffuse a'_D will be:

$$a_{D_1} = a_{D_0}(1 - r_I) + a_{D_0} N_R \, r_I$$

$$a_{D_2} = a_{D_1}(1 - g_I) + a_{D_1} N_G \, g_I$$

$$a'_D = a_{D_2}(1 - b_I) + a_{D_2} N_B \, b_I$$

Where r_I, g_I, b_I stand for the Red, Green and Blue channel of Alpha Control Image respectively, and N_R, N_G, N_B are parameters (range from 0 to 1) which get values from Red, Green and Blue slidebars.

6. IMPLEMENTATION AND RESULTS

6.1 Implementation

The languages we used in web-based Mock-3D project are: WebGL, GLSL, HTML, Javascript, JQuery, CSS and SVG. In this section, we will introduce a basic WebGL pipeline model we used and then, talk about the framework tools we use to assist our responsive front-end development.

6.1.1 WebGL Application Pipeline Model

WebGL is a javascript implementation of OpenGL ES 2.0, which is becoming increasingly more popular because it is supported by all browsers except Internet Explorer (and even that appears to be changing). Besides the advantage of being able to run without recompilation across platforms, it can easily be integrated with other Web applications and make use of a variety of portable packages available over the Web.

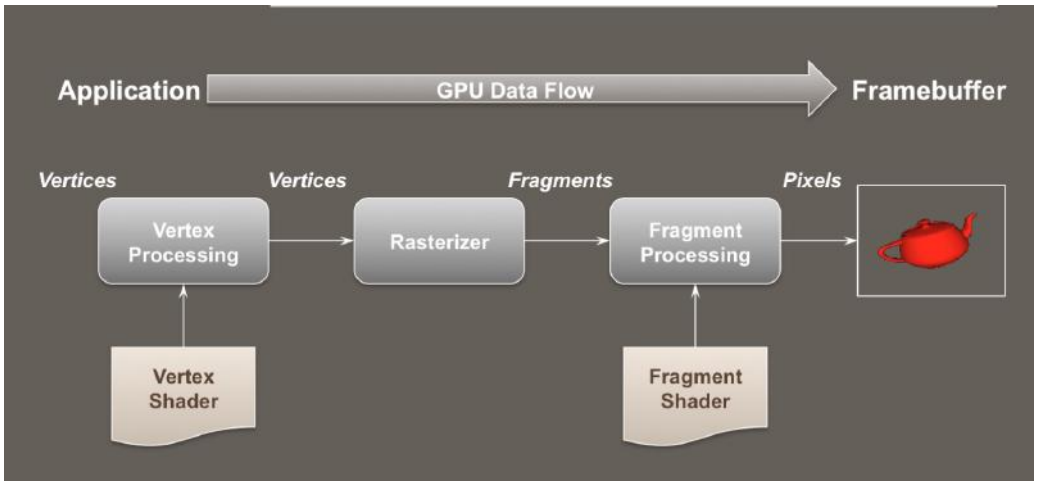


Figure 6.1: WebGL application simplified pipeline model [9]

Figure 6.1 is a simplified WebGL application pipeline model. Data flows from the application through the GPU to generate an image in the frame buffer. The application will provide vertices, which are collections of data that are composed to form geometric objects, to the OpenGL pipeline. The vertex processing stage uses a vertex shader to process each vertex, doing any computations necessary to determine where in the frame buffer each piece of geometry should go.

After all the vertices for a piece of geometry are processed, the rasterizer determines which pixels in the frame buffer are affected by the geometry, and for each pixel, the fragment processing stage is employed, where the fragment shader runs to determine the final color of the pixel.

In a WebGL program, we must do the following tasks:

- Set up canvas to render into - HTML5 Canvas element
- Generate data in application
- Create shader programs
- Create buffer objects and load data into them
- "Connect" data locations with shader variables
- Render

In our project, we setup an HTML file, and the application in a separate Javascript file. HTML file includes shaders, and it reads in utilities and application.

6.1.2 Front-end Development

We use Bootstrap as the framework to develop responsive projects. In addition, it provides a comparatively clean, refined and elegant interface, which also includes

source code to customize style, font and widgets through a LESS compiler to generate minified CSS and Javascript.

Left and right columns use a sidebar component, in which the left sidebar is set as “offcanvas” style when the width of the window of a device is smaller than 768px. All of the three views - left column, right column, and main center area - are in fixed position style in CSS and “overflow” style is set to “hidden” to avoid scrolling. However, both left column and right column “overflow” in y-direction are set to “auto”, in order to see content when one section is fully expanded.

We have three Accordion components in the HTML: one is for the left sections, another is for the right sections, the third is in multiple lights. We restrict to expand one item in one accordion list each time to achieve a clear navigation. This restriction also works in multiple lights to imply that only the expanded light can be edited and only its light position is influenced by current mouse position.

Since WebGL needs to work with canvas element in HTML, our result is displayed on a canvas. On the top of canvas, we have a hidden SVG layer to demonstrate the lights'positions, which can be displayed by checking on the “Show lights position” checkbox in the Light section.

6.2 Results

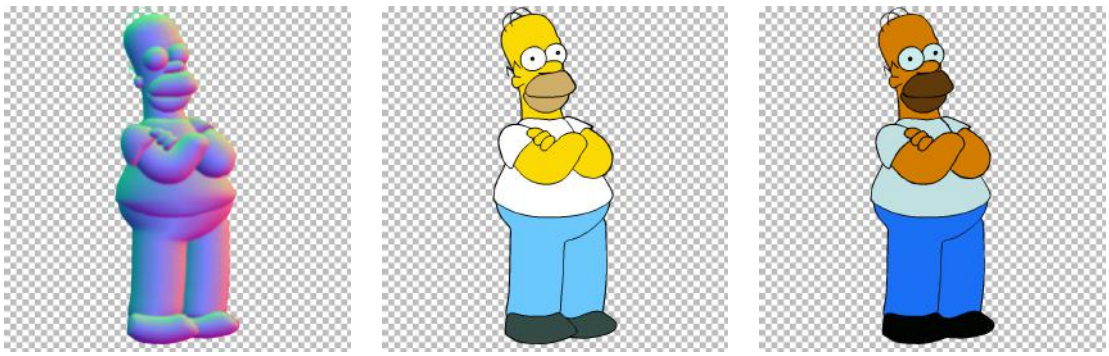
In this section, we will give some results based on using different controls in Mock-3D application to achieve interactive lighting, shading and rendering results. We will give the results of different shading styles, the use of refraction, half reflection and full reflection, then fresnel application, impossible shape examples and the result of using a artwork directly as a Shape Image.

6.2.1 Style

Here we will give some examples of using Mock-3D to achieve differenct styles: cartoon shading, patterns merging and artistic shading.

6.2.1.1 Cartoon Shading

As shown in Figure 6.2, the style control slider can be used to control the transition boundary from our Bright Image to Dark Image.



(a) Shape Image (b) Bright Image (c) Dark Image

(d) Results of different style control

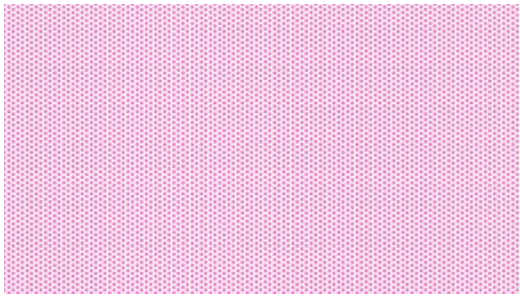
(d) Results of different style control

Figure 6.2: An example of cartoon shading application: Homer Simpson

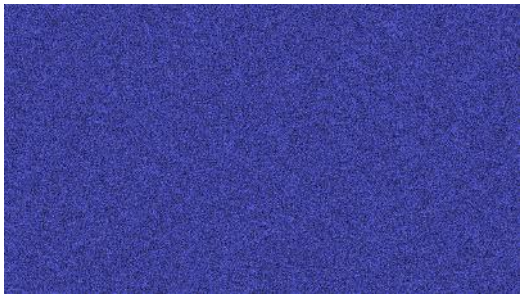
6.2.1.2 Pattern Merging

Instead of turning an artwork into two different shades to use as Dark Image and Bright Image in Mock-3D, we can apply any two different images/patterns to create an interactive artwork (see Figure 6.3).

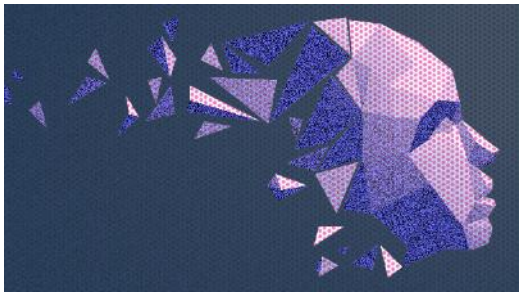
(a) Shape Image



(b) Bright Image



(c) Dark Image



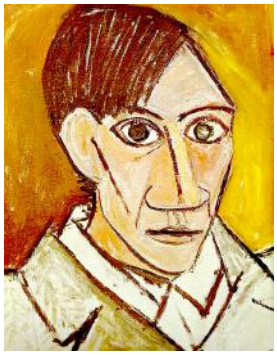
(d) Results of different light position

Figure 6.3: An example of pattern merging application.

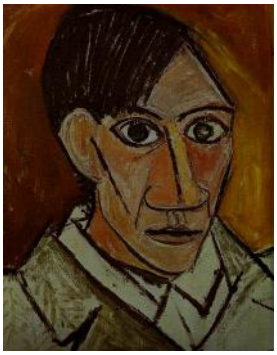
6.2.1.3 Artistic Shading

As show in Figure 6.4, the shape map image we used here is a manual painting Shape Image. In the results, we can see clear artistic strokes representing the boundary of light and dark. After we placed a light source in different positions, lights created different interesting mood scenarios.

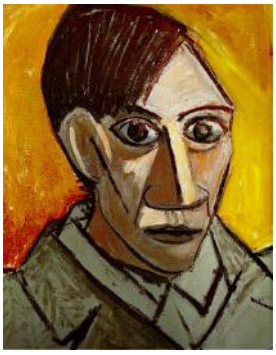
(a) Shape Image

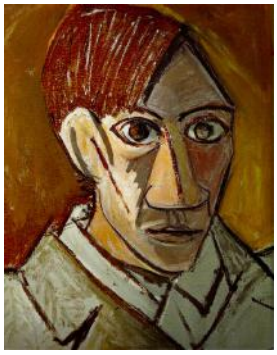


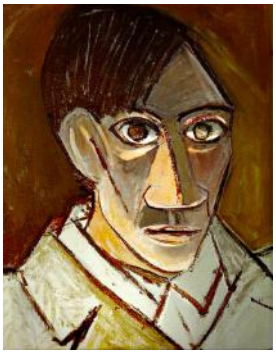
(b) Bright Image



(c) Dark Image







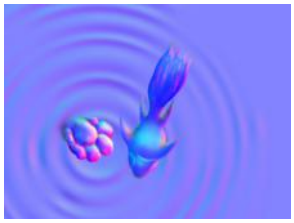
(d) Results of different light position

Figure 6.4: An example of artistic shading result. The artwork is modified by the painting "Self-portrait" from Pablo Picasso.

6.2.2 Refraction

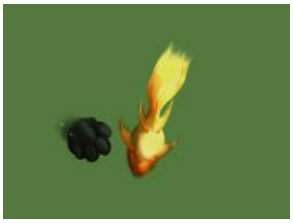
In the application of Background Image to get refraction results, we use Diffuse Alpha Control Image to make the diffuse result transparent. Then, the Background Image can be visible and be distorted according to the Shape Image and refraction control.

As shown in Figure 6.5, we assign different refractive values to the transparent part - ripple, to get interactive ripple results. The cat in the background is distorted in an interesting way after the refraction of the ripple.



(a) Shape Image

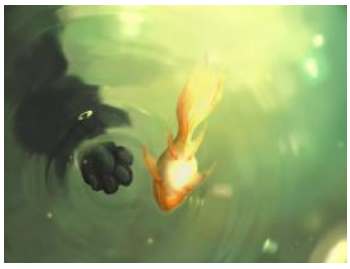
(b) Bright Image

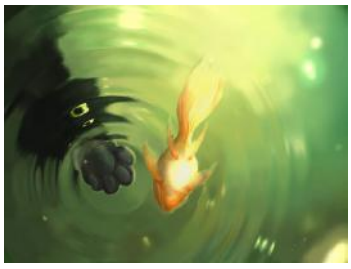


(c) Dark Image

(d) Background Image

(e) Alpha Image





(f) Results of different Index of Refraction

Figure 6.5: An example of a rendering of water refraction. The artwork is modified by the painting "Goldfish" from Jenni Ulrich [10].

6.2.3 Reflection

In this section, we will give some results of using a Foreground Image to be reflected according to the Shape Image to get full reflection results. In Mock-3D, Foreground Image can be scaled and shifted, and it will be shifted also according to the default light/mouse position. We will first give an example of full reflection on a specific area of the image. Then, we will give a result of the combination of diffuse and pure reflection.

6.2.3.1 Full Reflection

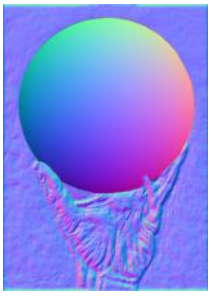
As shown in Figure 6.6, we had the ball in hand fully reflected by the Foreground Image mapped on plane. The famous artwork "Hand with Sphere" from M.C. Escher is interactive represented according to the cursor movement.

6.2.3.2 Diffuse with Full Reflection

In the case of representing a material which only have the reflection but no refraction, such as metal or waxed floor. We use an Alpha Image to control the mix of diffuse and the result of fresnel. As shown in Figure 6.7, we set 0.5 to the value of "alphaA" slider, and 1 to Fresnel control slider. These parameters stand for a half diffuse and half pure reflection appearance.

6.2.4 Fresnel

In a Fresnel application as shown in Figure 6.8, we use two different oil painting arts as Foregound Image and Background Image. After Fresnel calculation, we can see a distorted refractive cake background and a reflective room foreground at the same time. They blended naturely according to the shape of the bottle - most reflection on the edge or center of the bottle, other areas show more refractive results. The blending method can be intuitively controlled by users via a Fresnel Control

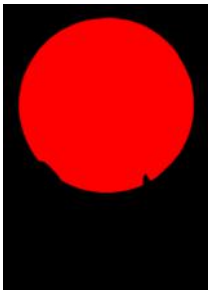


(a) Shape Image

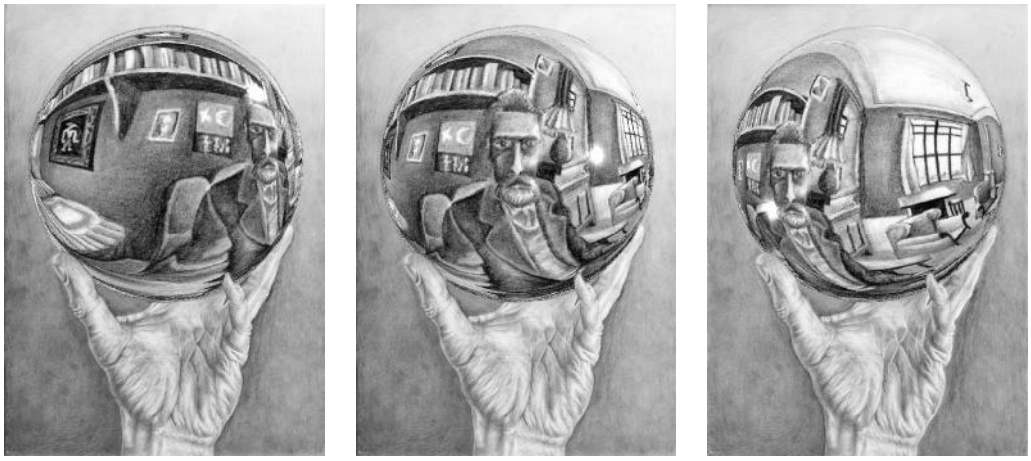
(b) Bright Image

(c) Dark Image

(d) Foreground Image

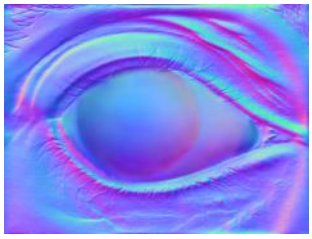


(e) Alpha Image



(f) Results of reflection foreground shift in x or y direction, using "map as Plane" method

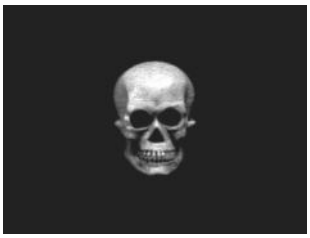
Figure 6.6: An example of interactive full reflection rendering. The artwork is modified by the drawing "Hand With Sphere" from M.C. Escher



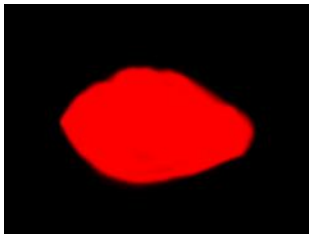
(a) Shape Image

(b) Bright Image

(c) Dark Image



(d) Foreground Image



(e) Alpha Image

(f) Results of reflection foreground shift in x direction,
using "map as Hemisphere" method

Figure 6.7: An example of the rendering of half diffuse half reflection. The artwork is modified by the drawing "Eye of Death" from M.C. Escher

slidebar and a Fresnel Position slidebar.

6.2.5 Impossible Shape Illumination

Figure 6.9, 6.10 and 6.11 show the possiblities of illuminating impossible shapes and impossible artwork. Three different color values are given to Shape Image by considering what precentage of the face aim to the right, top and towards camera individually. The shape Image of the third example (see Figure 6.11) is created by assigning a gradient map with 3 calculated color on the light, grey and dark shade of the original image by M.C. Escher.

6.2.6 Others: Artwork as Shape Image

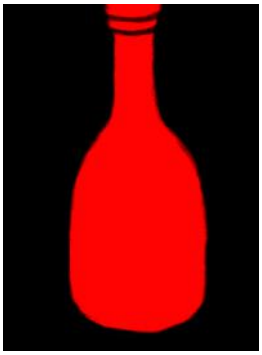
Figure 6.13 and Figure 6.14 show an example of using the original artwork (Figure 6.12) as Shape Image or Bright or Dark Image, instead of creating a Shape Image to explore more interesting results.

(a) Shape Image

(b) Bright/Dark Image

(c) Foreground Image

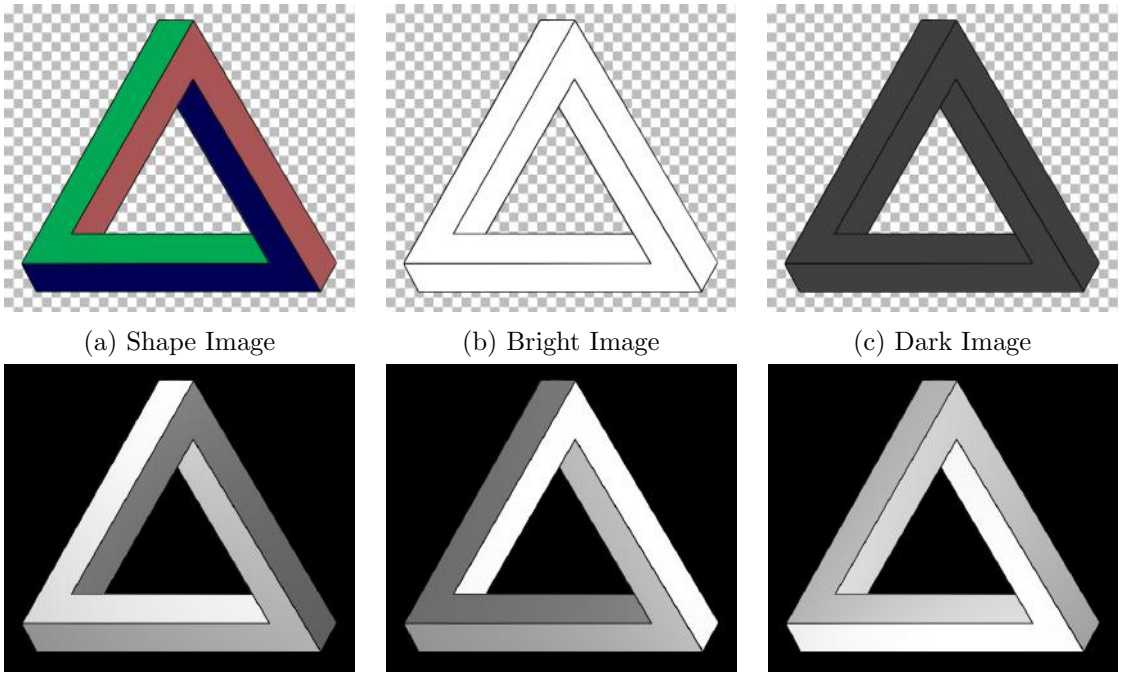
(d) Background Image



(e) Alpha Image

(f) Results of tweaking fresnel control slider and fresnel position slider

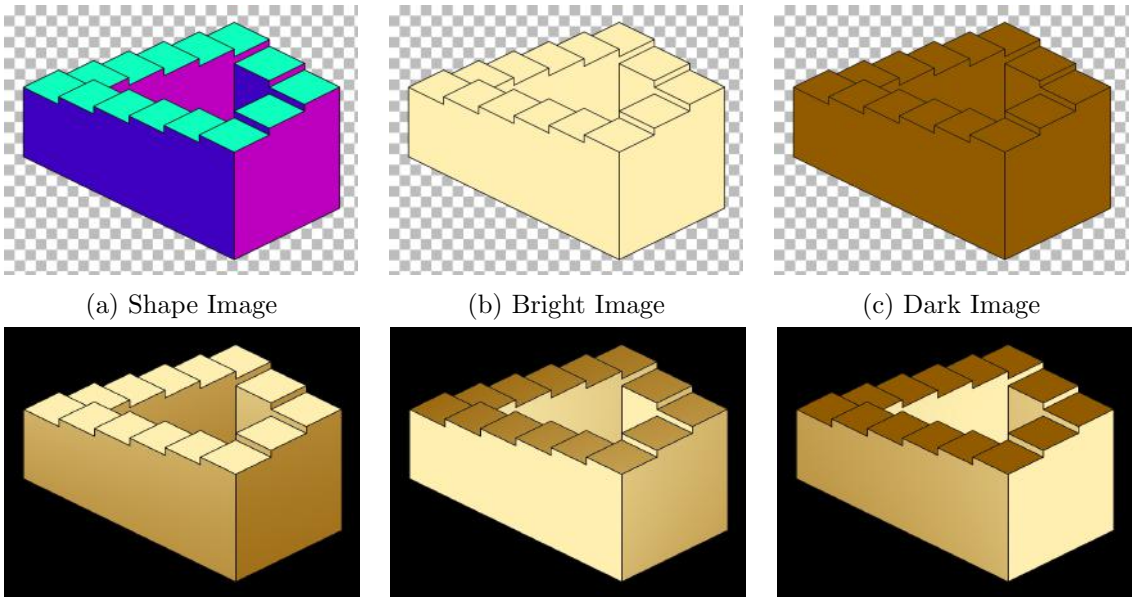
Figure 6.8: An example of interactive Fresnel control on the shape of a transparent bottle. Bright/Dark and Background Image modified by an artwork from Alison Mackay [7]. Foreground Image cropped and modified by an artwork from Cecilia Rosslee [11].



(a) Shape Image (b) Bright Image (c) Dark Image

(d) Results of different light position with a black background

Figure 6.9: An example of illuminating impossible shape: Triangle



(a) Shape Image (b) Bright Image (c) Dark Image

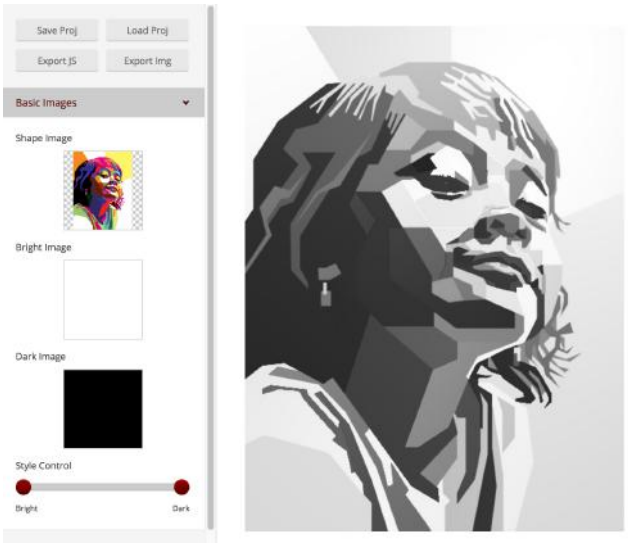
(d) Results of different light position with a black background

Figure 6.10: An example of illuminating impossible shape: Penrose Stairs

(a) Shape Image (b) Results of different light position

Figure 6.11: An example of illuminating impossible shape: M.C. Escher "Concave and Convex". The result using (a) as shape image, a solid white as Bright Image and solid black as Dark Image

Figure 6.12: An original artwork by Wedha Abdul Rasyid [12]

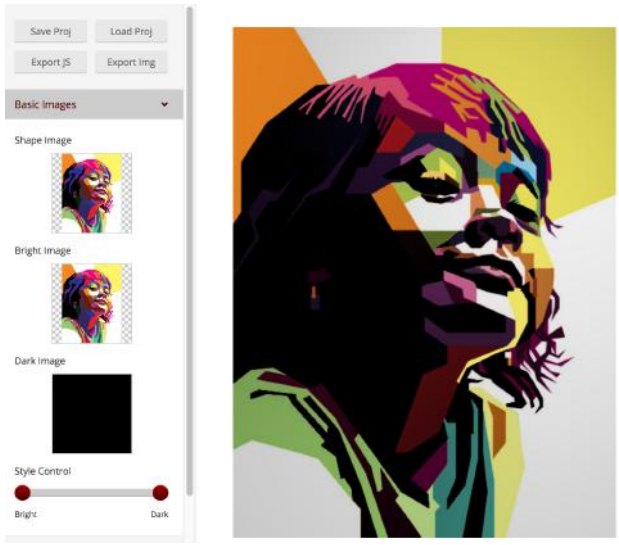


(a) Interface 1

(b) Results from different light position and style control

(c) Multiple lights results

Figure 6.13: Examples of using an original artwork (see Figure 6.12) as Shape Image to achieve a variety of results. (a) shows the interface with the input of Shape Image, Bright Image and Dark Image. The artwork is modified by a painting from Alison Mackay. The Foreground Image is cropped and modified by a painting from Atelier Cecilia Rosslee



(a) Interface 2

(b) Results from different light porsition

Figure 6.14: An example of using an original artwork (see Figure 6.12) as Shape Image and Bright Image to Achieve various Results. (a) shows the interface with the input of Shape Image, Bright Image and Dark Image.

7. CONCLUSION AND FUTURE WORK

The goal of this project is to let artists achieve a variety of qualitative physics inspirations and shading test with their personal artistic style. We try to provide an easy-to-use interface with complete control for the properties of alpha, lights, shadows, reflection, refraction, ambient occlusion and rendering qualities by uploading paintable control images and adjusting parameters. By using this tool, artists can easily turn their traditional 2D artwork into an interactive 3D-looking file with qualitatively convincing physically correct details. The major contribution of this tool is providing a flexible paintable method to light and render the impossible / incoherent / inconsistent shapes, and at the same time, the intuitive Fresnel Control method provides a new way to composite the reflection and refraction result in the artwork.

REFERENCES

[1] A. Gleizes and J. Metzinger, *Du" cubisme".* RG Fischer, 1993.

[2] D. Robbins, "Jean metzinger: At the center of cubism," *Joann Moser, Jean Metzinger in Retrospect. Iowa City: University of Iowa Museum of Art*, pp. 9–23, 1985.

[3] K. Jonathan, "Using local information for compositing cg into traditional art," Master's thesis, Texas A&M University, 2009.

[4] S. F. Johnston, "Lumo: illumination for cel animation," in *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, NPAR '02, pp. 45–52, 2002.

[5] C. Shao, A. Bousseau, A. Sheffer, and K. Singh, "Crossshade: shading concept sketches using cross-section curves," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 45:1–45:11, 2012.

[6] M. T. Bui, J. Kim, and Y. Lee, "3d-look shading from contours and hatching strokes," *Computers & Graphics*, vol. 51, pp. 167 – 176, 2015. International Conference Shape Modeling International.

[7] A. Mackay, "Cakes with glass bottle," 2013. Oil on Board.

[8] E. Hecht, "Optics, 4th," *International edition, Addison-Wesley, San Francisco*, vol. 3, 2002.

[9] D. S. Ed Angel, "An introduction to webgl," 2014. Siggraph2014 Course.

[10] J. Ulrich, "Goldfish," 2012. Mizu-no-Akiras Deviant Art page.

[11] C. Rosslee, "Baron de rode," 2013. Oil on Canvas.

[12] W. A. Rasyid, "How to create a geometric, wpap vector portrait in adobe illustrator," 2013. Online tutorial.

[13] J. Hart. John Hart, (2013) private conversion: According to a market research firm 3D Graphics is only 8% of the whole graphics market. 2D graphics such as vector, image and video is 90% of the graphics market.

[14] Y. Wang, *Qualitative Global Illumination of Mock-3D Scenes*. PhD thesis, Texas A&M University, 2014.

[15] Y. Wang, O. Gonen, and E. Akleman, "Global illumination for 2d artworks with vector field rendering," in *ACM SIGGRAPH 2014 Posters*, p. 95, ACM, 2014.

[16] J. Sun, L. Liang, F. Wen, and H. Shum, "Image vectorization using optimized gradient meshes," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 11, pp. 11:1–11:7, 2007.

[17] A. Orzan, A. Bousseau, H. Winnemoller, P. Barla, J. Thollot, and D. Salesin, "Diffusion curves: A vector representation for smooth-shaded images," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, pp. 92:1–92:8, 2008.

[18] D. Sýkora, J. Dingliana, and S. Collins, "Lazy- brush: Flexible painting tool for hand-drawn cartoons," *Computer Graphics Forum*, vol. 28, no. 2, pp. 599–608, 2009.

[19] M. Finch, J. Snyder, and H. Hoppe, "Freeform vector graphics with controlled thin-plate splines.," *ACM Transactions on Graphics (TOG)*, vol. 30, pp. 166:1–166:10, 2011.

[20] T.-P. Wu, C.-K. Tang, M. S. Brown, and H.-Y. Shum, "Shapepalettes: interactive normal transfer via sketching," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, pp. 44.1–44.6, 2007.

[21] R. Vergne, P. Barla, R. W. Fleming, and X. Granier, "Surface flows for image-based shading design," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 94, pp. 94:1–94:9, 2012.

[22] D. Sýkora, L. Kavan, M. Čadík, O. Jamriška, A. Jacobson, B. Whited, M. Simmons, and O. Sorkine-Hornung, "Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters," *ACM Transactions on Graphics (TOG)*, vol. 33, 2014(to appear).

[23] A. Gooch, B. Gooch, P. Shirley, and E. Cohen, "A non-photorealistic lighting model for automatic technical illustration," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 447–452, ACM, 1998.

[24] B. K. Horn, "Hill shading and the reflectance map," *Proceedings of the IEEE*, vol. 69, no. 1, pp. 14–47, 1981.

[25] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller, "Curvature-based transfer functions for direct volume rendering: Methods and applications," in *Visualization, 2003. VIS 2003. IEEE*, pp. 513–520, IEEE, 2003.

[26] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi, "Efficiently combining positions and normals for precise 3d geometry," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 536–543, 2005.

[27] K. ichi Anjyo, S. Wemler, and W. Baxter, "Tweakable light and shade for cartoon animation," in *Symposium on Non-photorealistic animation and rendering*, NPAR '06, pp. 133–139, 2006.

[28] J. a. P. Gois, B. A. D. Marques, and H. C. Batagelo, "Interactive shading of 2.5d models," in *Proceedings of the 41st Graphics Interface Conference*, GI '15,

(Toronto, Ont., Canada, Canada), pp. 89–96, Canadian Information Processing Society, 2015.

[29] C. Toler-Franklin, A. Finkelstein, and S. Rusinkiewicz, "Illustration of complex real-world objects using images with normals," in *Symposium on Non-photorealistic animation and rendering*, NPAR '07, pp. 111–119, 2007.

[30] T. Ritschel, M. Okabe, T. Thormlen, H. peter Seidel, and M. Informatik, "Interactive reflection editing," *ACM Transactions on Graphics (TOG) (Proc. SIGGRAPH Asia)*, vol. 28, no. 5, pp. 129:1–129:7, 2009.

[31] T. Ritschel, T. Thormlen, C. Dachsbacher, J. Kautz, and H. Seidel, "Interactive on-surface signal deformation," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, pp. 36:1–36:8, 2010.

[32] E. Akleman, D. House, and S. Liu, "Barrycentric shaders: Art directed shading using control images," in *Expressive 2016: Computational Aesthetics Conferences*, p. accepted, Eurographics, 2016.

[33] C. Abras, D. Maloney-Krichmar, and J. Preece, "User-centered design," *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, vol. 37, no. 4, pp. 445–456, 2004.

[34] K. Holtzblatt and H. R. Beyer, "Contextual design." Retrieved 20 April 2014.

[35] A. Cooper *et al.*, *The inmates are running the asylum:[Why high-tech products drive us crazy and how to restore the sanity]*, vol. 261. Sams Indianapolis, 1999.

[36] A. Cooper, R. Reimann, D. Cronin, and C. Noessel, *About face: The essentials of interaction design*. John Wiley & Sons, 2014.

[37] D. A. Norman, *Emotional design: Why we love (or hate) everyday things*. Basic books, 2005.

[38] J. F. Hughes, A. Van Dam, J. D. Foley, and S. K. Feiner, *Computer graphics: principles and practice*. Pearson Education, 2013.