Modeling Thick Paint on Japanese Lacquer ware

Shinichi Kinuwaki    Roman Đuriković
The university of Aizu.

Abstract
This paper presents a real time rendering method of beautiful and photo-realistic Japanese lacquer ware partly run on the GPU. We demonstrate depth effect of a thick paint with bump mapping techniques.

1 Introduction

Japanese lacquer is used in many objects such as table, chopstick, cup, trays and container. Lacquer ware is wooden containers with lacquer. Tree sap is used to make clear raw lacquer. In making of lacquer, there are three steps. They are carving a container out of wood, coating it with ware, and drawing pictures on a surface.

In the previous works [1] [2], two stage methods for rendering were proposed. In the first stage, radiance sphere maps were computed for each metallic powder and color pigment. We express lacquer ware(urushi) optical effects using BRDF(bi-directional reflectance distribution function) that is a function describing how the light propagates from incident to outgoing angle. In the second stage, they blended computed sphere maps and textures with controlled alpha channel.

In this work, we focus mostly on second rendering stage. Recently, GPU rendering using programmable shader has been displacing prior fixed function pipeline. To get flexible extension, we propose a urushi rendering technique using programmable shader. Sphere coordinate is calculated on vertex shader, multi texture blending operation is done on pixel shader, and bump effect called bump mapping is added to the surface in pixel shader, too.

1.1 Generating sphere map

When ray from viewer intersects any parts on an object, computing radiance from 4D tabular BRDF is brute force task. We adopted a radiance sphere map [3]. To make sphere map, an orthographic projected sphere whose surface has the same BRDF as target object is used. To render a sphere map, we select a pixel from sphere map, shoot a ray from the camera through this pixel, and find where it hits the sphere. At this point, we fire rays to all light sources and calculate the radiance from the sphere BRDF, the local normal, the viewing and illumination directions as

\[ L(\vec{w}_o) = \sum_k f(\vec{w}_{ik}, \vec{w}_o) I_k(\vec{w}_{ik}) \]

where \( L \) is the radiance in the \( \vec{w}_o \) outgoing direction, \( f \) is the BRDF, \( I_k \) is the incident radiance of the sphere at the point hit by the ray, coming from the \( k \)-th light source, \( \vec{w}_{ik} \) is the direction to the \( k \)-th light source [1].

As stated by Đuriković et al [1], realtime recalculation of a sphere map is possible by using the BRDF representation according to the specular direction with adaptive mesh and the color interpolation between mesh points done in OpenGL with Gouraud Shading.

The result images are shown Figure 1.

1.2 Sphere map coordinate

To access sphere map, we use simple reflected vector as \( R = E - 2N(E \cdot N) \) where \( E \) and \( N \) are eye and surface normal vectors, respectively. When viewing ray intersects object, we compute sphere \((u,v)\) coordinate directly from camera coordinate as illustrated in Figure 2. For every visible vertex, its normal vector is projected into the camera coordinates.
projective plane and translated into the center of normalized sphere map.

![Diagram](image)

Figure 2: We compute sphere coordinate directly from camera coordinate of projected normal vectors.

1.3 Blending of multiple sphere maps

We capture textures with alpha channel taken by digital camera. We blend each sphere map with its corresponding alpha texture. Multiple texture blending is done on pixel shader.

2 Bump mapping

Previous visualization [1] [2] regards all surfaces as flat surfaces. The real surfaces are not so. To add more realism, we use a simple bumpy technique called bump mapping. Bump mapping was introduced by Blinn in 1978 [4]. Bump mapping converts flat surface by varying the into bumpy surface’s one to change object’s normal vector in illumination equation as illustrated in Figure 3.

![Diagram](image)

Figure 3: Left figure is flat surface. Right surface seems not flat after normal is changed by normal map.

A RGB color texture called normal map encodes how large is the roughness of surface. The normal map is generated from given gray scale height field image. The gradient of the height field in each pixel is stored as normal map \( \nabla N = (R, G, B) = (\nabla_x, \nabla_y, 1) \) which will modify the original normal at the object vertex.

2.1 Bump environment mapping

The environment sphere is calculated in camera coordinate system, while the normal map is surface given in tangent space coordinate where the normal without perturbation is always towards Z direction \((0,0,1)\).

Following transformation matrix [5]

\[
M = \begin{pmatrix}
T_x & B_x & N_x \\
T_y & B_y & N_y \\
T_z & B_z & N_z
\end{pmatrix}^{-1}
\]

will convert tangent space into object space, where \( T \) means tangent vector, \( B \) binormal vector, and \( N \) is normal vector in tangent space. Finally, we multiply the result with the camera viewing matrix, \( T_{\text{view}} \), to get the object normal in camera space:

\[
N_{\text{camera}} = T_{\text{view}} M \nabla N.
\]
3 Implementation

We compute texture coordinate, blending multi
textures, and bump mapping with OpenGL and
nVidia Cg1.1 [6]. We use 7 textures including
3 sphere map textures, 3 alpha channel textures,
and a bump map in a single pass. We implement
our visualized program over 60 fps with Pentium4
2.40Ghz, 512MB RAM, and GeForceFX 5900 Ul-
tra. Top of Figure 4 demonstrates the rendering
of thick paint with bump mapping and metallic
sphere maps. Comparing to flat paint shown at
bottom, we got better impression of thickness.

4 Conclusion

Real time rendering visualization of thick paint
was proposed using the bump maps and direct
calculation of texture (u,v) coordinates related to
camera space. The approach is simple and fast.

Figure 4: There are result images. Top thick paint
with bump mapping. Bottom : flat paint without
bump mapping.

References

[1] R. Đuriković, K. Kolchin, S. Ershov, “Ren-
dering of Japanese Artcraft,” Proceedings of
the EUROGRAPHICS short presentations, 131-

"Real-Time Visualization of Japanese Artcraft"
Proceedings of the Computer Graphics Interna-
tional 2003.

[3] Blinn, James,"Simulation of wrinkled sur-
faces,"Computer Graphics(SIGGRAPH 78 Proceed-

computer generated images", Communication of
the ACM 19 542-546, 1976.

Bump-mapping Technique for Today’s GPUs”,
GDC2000: Advanced OpenGL Game Develop-
ment.

[6] "Cg Toolkit User’s Manual A Devel-
oper’s Guide to Programmable Graphics”,
http://www.cgshaders.org/tools/.