CS348b Final Project Report: Frog Pot

Katherine Liu, Lucy Zhu

June 2019

Figure 1: The final rendered image at 2400 x 1800, 1024 samples per pixel with an integrator depth of 5. Render time was approximately 5 hours.

1 About

Glazed ceramic is a material that consists of multiple layers; one layer sets the hard reflective appearance on the ceramic object's surface, and another layer creates the soft coloration underneath. Like other coated materials, such a surface appearance can be closely simulated with the layered materials technique. Light rays traveling through the layers would bounce back and forth between layers before scattering to create the final appearance. In this project, we attempt to implement a `Ceramic` Material to simulate the surface of glazed ceramic. Our ultimate goal is to modify our `Ceramic` Material to become a general `Layered` Material to simulate materials such as car paint on a metal surface or frosted glass.

2 Inspiration



We both love frogs and plants, and so we wanted to create an image that combines our interests. We combined these three photos into an image from our imagination.

3 Implementation

3.1 Approach

In order to mimic the appearance of a glazed ceramic pot, we decided that having two layers would be sufficient. The top layer is the glaze which would be simulated by a `Glass` Material, since glazes allow transmission and have fairly specular reflections. The bottom layer is the actual clay surface and will be simulated by a `Matte` Material with a diffuse texture. The interaction of the two layers is to be implemented in a new material we named `Ceramic`. To do so, we modified `api.cpp` to accept "ceramic" as the new material and to define the layer materials ("glass" and "matte") as the set-up.

In the `Ceramic` Material class that we created, we take in two materials as parameters. As said earlier, the first is `Glass`, and the second is `Matte`. We account for each layer's properties. We take in the `Glass`'s index of reflection and transmission and the `Matte` material's texture. In our case, that texture would be the `frog.png`, the texture of the frog pot. `Ceramic` calculates the scattering of each material and adds to the BSDF of the surface interaction.

While it would be great if the layers could be treated separately such that we first address one layer before addressing the other, this approach is flawed because the layers do not operate separately. Rays that go through the top layer are able to reflect in the bottom layer back to the top layer. In other words, rays bounce back and forth between the two layers before finally exiting through the bottom layer's base or the top layer's surface. (See Figure 3.)

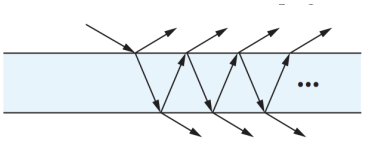


Figure 2: Light rays bouncing through a single layer of a material. Courtesy of A Comprehensive Framework for Rendering Layered Materials

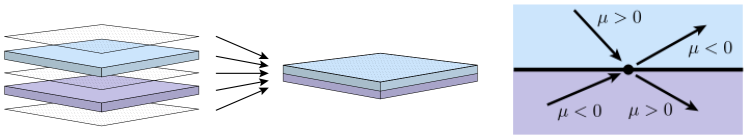


Figure 3: Light rays bouncing within two layers of materials. Courtesy of A Comprehensive Framework for Rendering Layered Materials

We modeled our implementation of the layered materials on the equations below, from A Comprehensive Framework for Rendering Layered Materials:

$$\widetilde{R}^t = R_1^t + T_1^{bt}(I - R_2^t R_1^b)^{-1} R_2^t T_1^{tb}$$
$$\widetilde{R}^b = R_2^b + T_2^{tb}(I - R_1^b R_2^t)^{-1} R_1^b T_2^{bt}$$
$$\widetilde{T}^{tb} = T_2^{tb}(I - R_1^b R_2^t)^{-1} T_1^{tb}$$
$$\widetilde{T}^{bt} = T_1^{bt}(I - R_2^b R_1^t)^{-1} T_2^{bt}$$

where R_1^t represents the reflection matrix transformation for rays reflecting off layer 1's top surface and T_2^{tb} represents the transmission transformation matrix of rays through layer 2 from the top surface to the bottom surface. I, of course, is the identity matrix.

However, this is only a concern if both layers are translucent or transparent. Only the top layer is transparent and so we only have to account for reflections off the bottom layer's top surface (matte material has no transmissions).

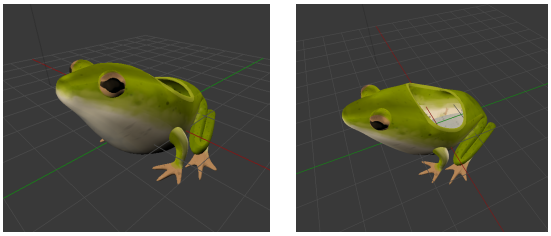
3.2 Challenges and Issues

We ran into trouble trying to implement more than just two layers of materials. The previously described set of equations only works for two layers. With a third layer, we would have to be careful with the order in which we process our layers, because matrix multiplication is not commutative, and thus we have to be careful of the order in which we have the matrix transformations.

4 Modeling

4.1 Frog

Figure 4: The original tree frog model from Google Poly.

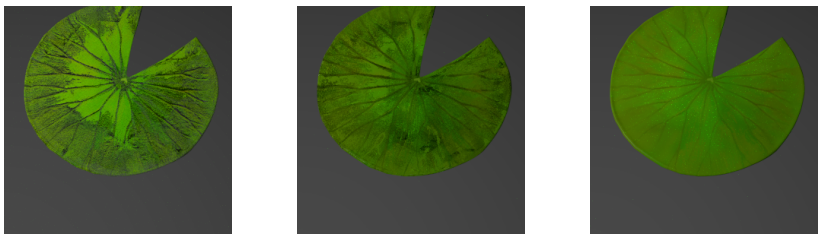


(a) (b)

Figure 5: Our modified frog pot model. Image 5a shows a 3/4 front view of our frog pot. Image 5b shows the frog's back, where one can see the hole in which to place plants.

We started our modeling process with the frog, the main subject of our image. We were unable to find any frog pot models online, so we opted to take an existing frog model and make it into a pot. We obtained this tree frog from Google Poly, and we adjusted its general shape to attain a bowl-like roundness. We created a hole in the frog's back and established thickness with the shell method. We then painted the frog in Photoshop and in Blender using the texture paint function to give it a speckled, painted appearance.

4.2 Lily Pads

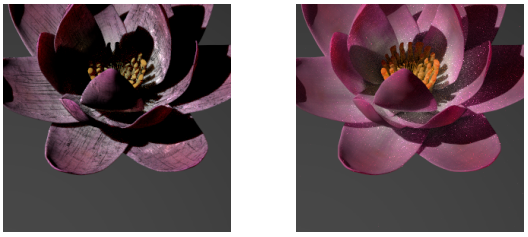


(a) (b) (c)

Figure 6: Progression of our lily pad model.

We modeled one lily pad for our scene and reused it three times. We started by creating a very thin cylinder, cutting out a triangular shape, and slightly raising the edges to emulate the imperfect shape of a real lily pad. We painted the lily pad in Photoshop. We had some trouble with the bump map, as seen in 6a and 6b, but we adjusted the bump map values until we were satisfied with the lily pad in 6c.

4.3 Water Lily



(a) (b)

Figure 7: Image 7a shows our render of the water lily with the `KdSubsurface` Material and bump map. Image 7b shows our render with the `Translucent` Material.

We obtained the lily pad model from Turbosquid. We first tried rendering it using a `KdSubsurface` Material, but found that the petals looked too rigid and desaturated. The `Translucent` Material worked much better; the petals of the flower now look delicate and saturated. We also applied the same material to the lily pad model to attain the same result.

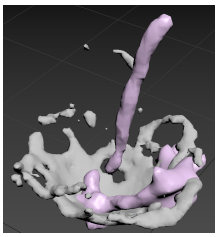


Figure 8: A splash of water from the watering can.

4.4 Moving Water

We modeled the moving water with 3DS Max using fluid simulation. We created the splash and the stream of water in two separate simulations.

In the first simulation, we created a sphere to obtain the desired bowl-shaped splash. After the particles collided with the sphere, the resulting model was flipped over.

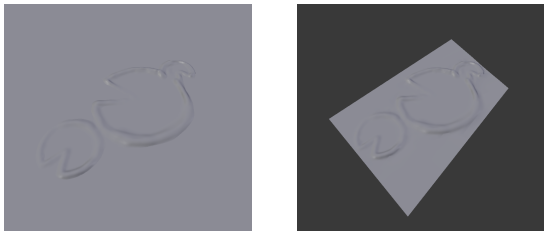
In the second simulation, we first added an initial velocity to generate the curve of the water flowing out of the watering can's spout and tweaked the kernel, viscosity, and the dimensions until the fluid appearance was to our liking.

When rendering, we applied a `Glass` Material to the water.

4.5 Watering Can

We obtained the watering can model from TurboSquid and modified it slightly to have a thin spout with a single hole instead of a wide head with multiple holesm as was originally in our inspiration photo. We decided upon this for the sake of the aesthetic value that a single stream of water would have, as opposed to the aesthetics of many droplets of water raining down. We wanted the water to be clearly seen against the dark background and limited lighting.

4.6 Pond



(a) (b)

Figure 9: Image 9a shows both layers of the pond model. Image 9b shows only the top layer.

The pond consists two planes layered on top of each other. The bottom layer is a single large plane that provides a flat, undisturbed surface. The top layer is smaller, establishing the water tension around the lily pads. In order to obtain the same glossy look that the pond water had in our inspiration photo, we set the water's material to be `Glass`. The pond water appears black because it is transparent and the space beneath the water is empty, but the black color works to our advantage.

4.7 Reeds

We obtained an image of reeds from publicdomainpictures. We created a large plane behind the scene and UV mapped the image onto the plane. We applied a Gaussian blur on the background image to simulate depth of field.

5 Conclusion

We were hoping to be able to extend our `Ceramic` Material to a general `Layered` Material for all kinds of layered materials. We also wanted to add support for more than two layers. Unfortunately we were only able to correctly simulate a `Ceramic` Material with a layer for the glass-like glaze and a layer for the matte paint. If we had more time, we would have wanted to use our general `Layered` Material to apply a painted metal look to our watering can, which currently looks unremarkable.

6 Contributions

The final project was a collaboration between two team members. Lucy worked on geometry modifications of the frog and the can, modeled the splash and fluid, worked with the materials, and implemented the code to create the ceramic

material. Katherine modeled the lily pads and pond, created the texture of all objects, and read papers about layered materials.

7 Bibliography

Laurent Belcour. "Efficient Rendering of Layered Materials using an Atomic Decomposition with Statistical Operators." ACM Transactions on Graphics, Association for Computing Machinery, 2018, 37 (4), pp.1. ff10.1145/3197517.3201289ff. ffhal-01785457v3f

Pharr, Matt, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. http://www.pbr-book.org/.

Wenzel, Jakob, Eugene d'Eon, Otto Jakob, et al. "A Comprehensive Framework for Rendering Materials." *Cornell.Edu*. http://www.cs.cornell.edu/projects/layered-sg14/layered.pdf.

Wenzel, Jakob. "layerlab: A Computational Toolbox for Layered Materials." http://www.mitsuba-renderer.org/ wenzel/papers/layerlab-sg15.pdf.