Honors Senior Project Report

Reuben Wattenhofer

April 26, 2019

For my senior project, I had proposed to create a hair model, including the hair modeling, rendering, and even animating, time permitting. However, the scope narrowed as the size of the task became apparent, and I instead spent the semester working on the hair rendering only. Fortunately, the ICT provided me with several ribbon-patch hair models with accompanying textures, so I was able to immediately begin on the rendering techniques. In this report, I will discuss the three main techniques I implemented, including their advantages, drawbacks, and bugs. The project code itself is available at my Github repository:

https://github.com/ReubenWattenhofer/hair-renderer



**Figure 1.**

A basic shader, left, compared with my shader on the right.

**Kajiya-Kay**

The most important technique I implemented was the Kajiya-Kay lighting model. Hair has special properties which make the specular reflections band-shaped rather than circular or oval. To achieve this effect, I found a Kajiya-Kay implementation in a SIGGRAPH presentation (Scheuermann, 2004) and integrated it with my Unity shader. One of the few modifications I made was related to self-shadowing: the SIGGRAPH implementation used ambient occlusion textures to simulate self-shadowing, and I removed this since I implemented a more realistic technique. In addition, I added ambient lighting within the Kajiya-Kay function, which I think is important for simulating most lighting conditions (there aren’t many environments without ambient lighting). Ambient lighting is a potentially complex topic, but I simply used trial-and-error to determine the best place to insert the ambient light value.



**Figure 2.**

Kajiya-Kay lighting creates a specular band, mimicking real hair.



**Figure 3.**

Without ambient lighting, the hair facing away from the light would be entirely dark.

Kajiya-Kay is not perfect. If the camera and light source are pointing directly at each other from opposite directions, the resulting lighting is somewhat unrealistic. However, the effect is not too jarring, and can be circumvented by avoiding steep horizontal lighting angles.

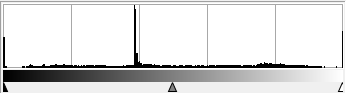
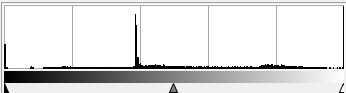


**Figure 4.**

Specular highlight when the head is between the camera and the light.

**Deep Opaciy Maps**

Another technique I implemented was deep opacity maps. Developed by Cem Yuksel and John Keyser (2008), the purpose of deep opacity maps is to provide realistic self-shadowing in hair. Since hair is semi-transparent, light can transfer through a strand of hair and into the neighboring strands. However, light diminishes as it travels through the volume of hair, and without taking this into account the hair will end up too bright.

**Figure 5.**

Deep opacity maps, disabled on left and enabled on right, with accompanying histograms.

As we can see in Figure 4, the effects are none too obvious – there is some slight shading moving in the right direction. I implemented the technique correctly as far as I could tell, so I am not sure why the results are so lackluster.

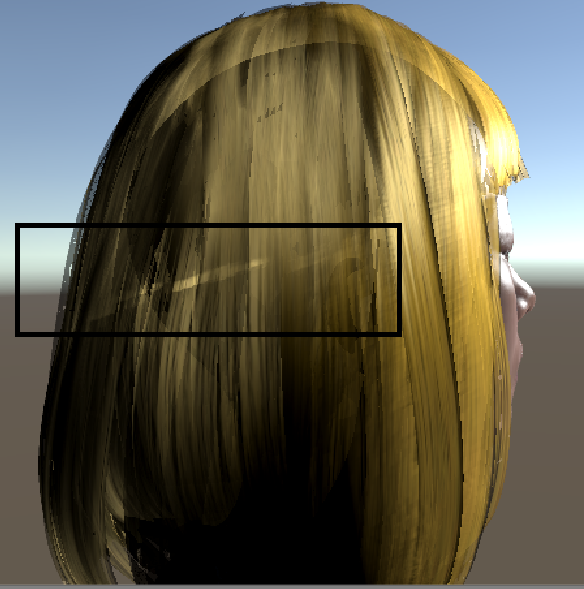
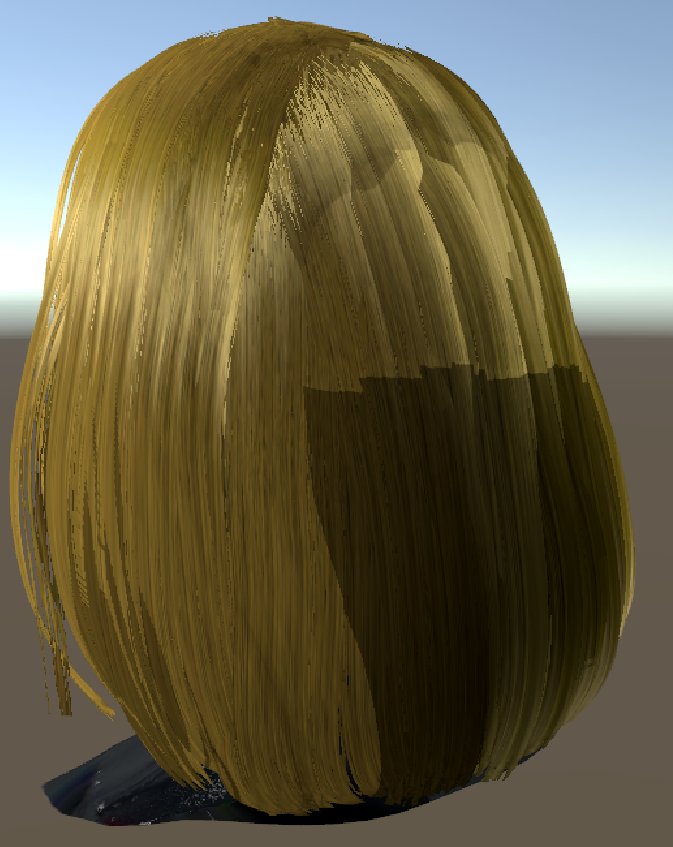


**Figure 6.**

Another example, with deep opacity disabled on the left and enabled on the right. Notice the slight darkening on the left side of the righthand image.

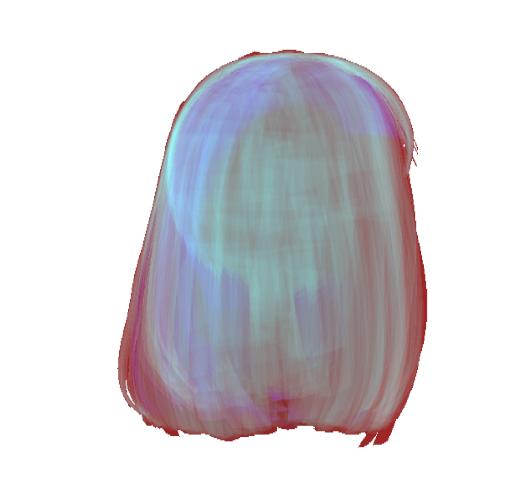
Deep opacity maps are created in two passes. The first pass generates a depth map of the hair. The second pass creates three opacity layers, using the depth map to conform the layers to the shape of the hair. This conformation is what makes deep opacity maps superior to previous self-shadowing techniques, since the interpolation between opacity layers will almost always occur inside of the hair volume, where errors are less likely to be noticed.

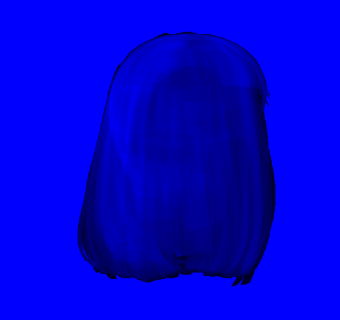
In addition to these two passes, I also perform another depth pass for the head, and use the head depth in addition to the hair depth when establishing base depth. This is because the hair behind the head will otherwise have very low opacity compared to the surrounding hair, resulting in an unnatural spike in brightness.



**Figure 7.**

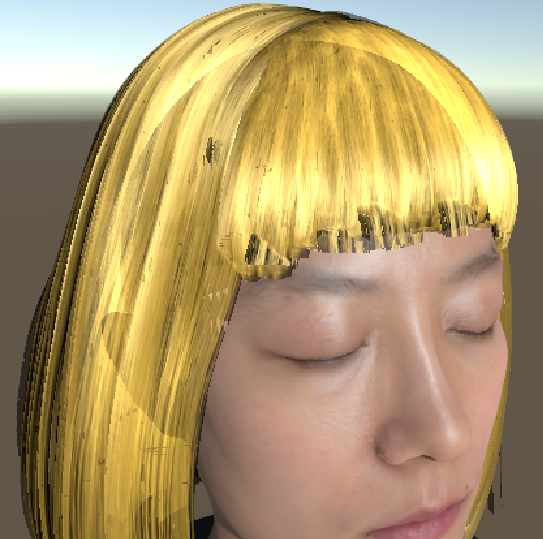
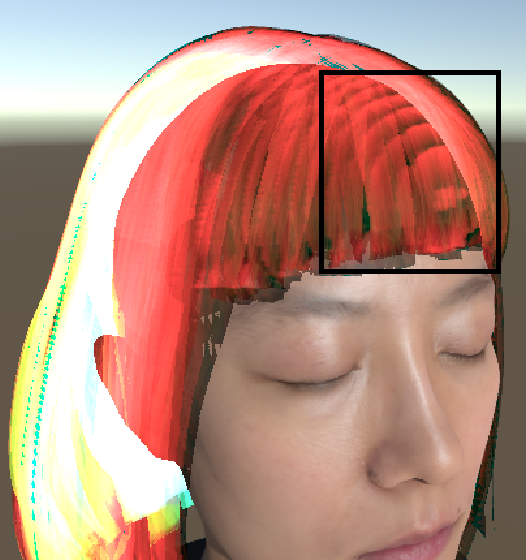
The sudden change in brightness in the left image is occurring because the head is not being factored in when calculating base depth. On the right, we can see that a lighting artifact still occurs even with the supplementation of the head depth, though it is not as egregious as it might otherwise be.





**Figure 8.**

A deep opacity map. The red channel is the depth map, while the green, blue, and alpha channels are reserved for the opacity layers.



**Figure 9.**

A deep opacity map projected onto the hair volume; each of the three colors represents an opacity layer. In the middle image, notice the “rings” of opacity, and how they do not affect the rendered image on the right because they are hidden inside of the hair volume.



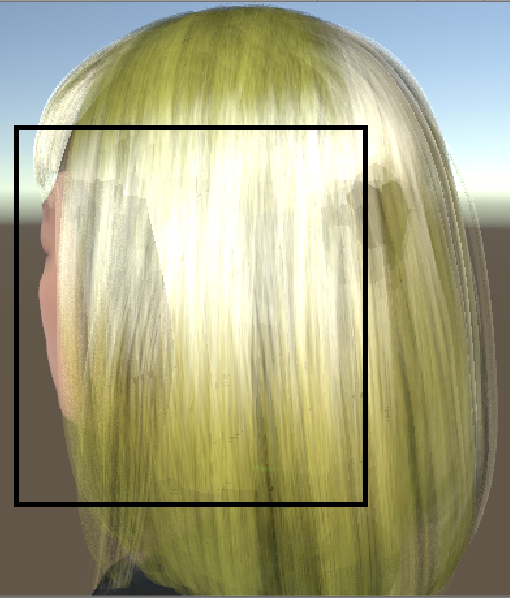
**Figure 10.**

The background of the deep opacity map must be white to prevent artifacts similar to the above from appearing (thought the artifacts will actually be dark, not light). This occurs due to imperfect texture sampling.

**Transparency**

Transparency is something that I attempted to implement several times over the course of the semester. The order in which semi-transparent fragments are rendered affects the outcome of the image – fragments must be rendered back to front, ending with the closest fragment to the camera. I first attempted to sort each ribbon patch in the hair mesh, using a CPU algorithm. However, I was not able to get it working, and also wanted a more general technique that didn’t depend on a ribbon-patch hair model.

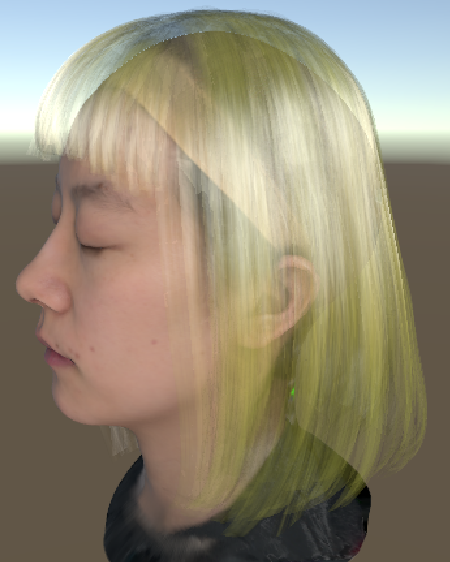
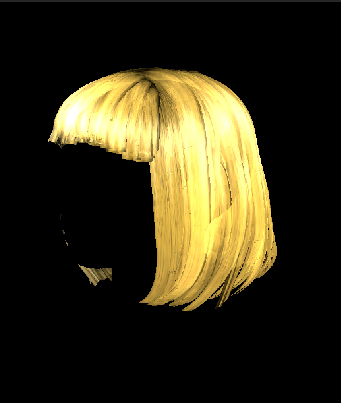
Eventually, I discovered a paper written by Erik Sintorn and Ulf Assarsson (2009) which described an order-independent transparency algorithm they developed. They use a technique they named “occupancy maps”, which function somewhat similarly to deep opacity maps: in fact, they claim the algorithm can be used as a better replacement for deep opacity maps. I won’t go into all of the details of the algorithm, partly because I wasn’t able to fully implement it. The main idea I used from the paper was a formula they described, where a fragment’s color is inversely proportional to its depth in a volume. By scaling each fragment’s color based on its depth, the results can be additively blended for a realistic look. However, the fragments’ depths must be known, or at least intelligently guessed. To keep track of the fragment depths, I implemented a “slab map” as described in the paper, which records the number of fragments that fall in each texel. In addition, I used a depth map of the head to determine if a hair fragment is out of sight; if so I discard the fragment without recording it. Without this, the transparency effect doesn’t work properly.



**Figure 11.**

Transparency without using the head depth map. Notice how the shape of the hair behind the head is visible.

One of the biggest hurdles with transparency is blending with the background. To get around this, Sintorn et al suggested rendering the hair to a separate buffer. Again, I used the head depth map to cull fragments behind the head prior to rendering. Then, the background color is reduced based on how many hair fragments overlay it (requiring another slab map reading), and the hair buffer is rendered on top of the background.



**Figure 12.**

Hair buffer on left, and the middle image shows it blended with the background. On the right, with high transparency used, we can see that blending occurs seamlessly.

As I mentioned earlier, I was not able to completely implement the technique proposed by Sintorn et al (it involves using a higher-resolution map in addition to the slab map to achieve realistic transparency). Perhaps because of this, there are two major flaws with my implementation. The first is that the head is visible underneath the hair when light hair colors are used. The second problem is a visual glitch, somewhat resembling static, that occurs on the surface of the hair volume. I presume both problems are related to my somewhat-slipshod method of estimating fragment depth order -- I only use the slab map, which isn’t very high-fidelity. Despite this, I am quite pleased with the current transparency and how it adds to the realism of the hair.



**Figure 13.**

No transparency on left, compared with transparency on right. Notice how the head is distinctly visible under the hair. Perhaps the model does not have enough layers of hair?

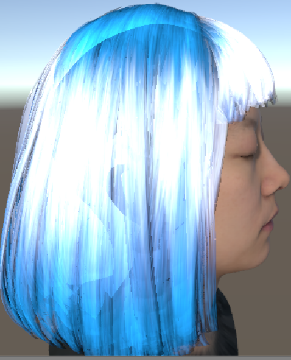
**Limitations**

In addition to the bugs and limitation I outlined throughout the report, this renderer only works with a single directional light. It might be possible to support multiple directional lights with a few changes, but point lights are not supported at all, and would require significant refactoring and design changes.

Regarding the capabilities of this renderer in real-time, I did not perform any real tests beyond observing framerate. After I implemented hair transparency, I noticed that the framerate will drop from 70 fps to 20 (i7 3rd gen CPU, GTX 960 4GB ram) if the camera is zoomed into the hair volume such that it takes up the entire screen. This is likely due to the many screenspace buffers that are required for the transparency effects. I’m sure that the framerate could be brought up, either by using fewer passes/buffers or by compressing the data more effectively – I used 32bit RGBA render textures for all of the maps used in transparency.

**Acknowledgements**

I would like to thank Dr. Hans Dulimarta and Kalle Bladin for the invaluable assistance they provided throughout this project. Without their help and support, I could not have achieved what I did.



**Figure 14.**

Thank you Jun Xing for suffering through many bad hair days on my behalf. Wasn’t it worth it?

**References**

Scheuermann, T. (2004). Practical real-time hair rendering and shading. *ACM SIGGRAPH 2004 Sketches on - SIGGRAPH 04*. doi:10.1145/1186223.1186408

Sintorn, E., & Assarsson, U. (2009). Hair self shadowing and transparency depth ordering using occupancy maps. Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games - I3D 09. doi:10.1145/1507149.1507160

Yuksel, C., & Keyser, J. (2008). Deep Opacity Maps. Computer Graphics Forum, 27(2), 675-680. doi:10.1111/j.1467-8659.2008.01165.x