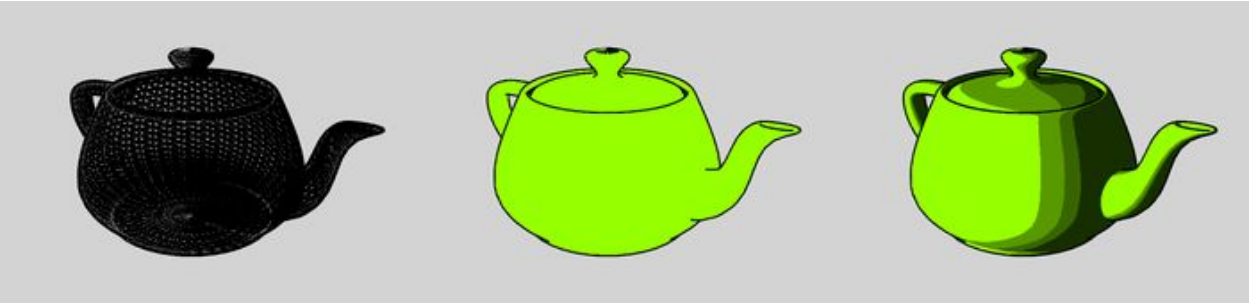
Depth Based Cel Shading & Silhouettes

Niall Miner and Brian Yoon

Our project focused on a technique that is not often implemented into works; distance based cel shading. The challenge was to make different objects at varying distances have different levels of cel shading. This improves performance and keeps the viewer's eyes focused on what is in the foreground of a scene. The solution we found that worked best was to calculate the distance from every object to the camera on each pass and use that number to determine an appropriate shading level. This was then attached to a uniform which was passed to the shaders to apply those correct bins to achieve the desired look. To incorporate silhouettes we were unsuccessful in achieving ridge detection and adjacency lists, but we were able to get them through the using of multi pass rendering where an object is drawn multiple times - the first time being the black outline.

Introduction

The technique in mind is cel shading, which is a form of coloring used in cartoons and anime, but can be applied to video games. On top of cel shading is the use of silhouetting to provide an outline to ridges and creases to give the game an even more accurate cartoon feel, as if the lines were drawn onto the screen instead of being rendered. The idea behind cel shading in computer graphics is to put each fragment's color into one of the predetermined "bins" to clamp it to a range of colors. In the image below you can notice that the gradient is not smooth, there are very obvious regions where colors are bound, giving the typical cel shaded field.



However, the whole point of cel shading is to give off the feel that whatever you are looking at is not realistic. And if you have every object cel shaded throughout a scene, including those that are very far away, the image may look to sharp, when in reality you lose sharpness of objects the further away they are. Even in drawing linearts artists generally make things in the image that are supposedly "closer" to the viewer with thicker lines, and things going into the page thinner. That is our goal for this project: to make objects closer to the camera have more defined regions of color and thicker lines, while objects further away have less regions (making them have less detail overall) and thinner lines.

Problem Statement

Cel shading in itself is not the challenging part of this project. Silhouetting and finding ridges and creases is going to be a little challenging but resources are available. The hardest part will be using the distance to the camera as a variable for cel shading regions. With silhouetting we need to find a way to create them either with adjacency lists or with double rendering for each object. The goal is to give it a more natural realistic feel, while maintaining what makes cel shading look so classic. This will hopefully help us with experience in performance and distance based rendering that we could use in our own projects in the future. It will also give us insight as to how the gaming industry uses this technique and if our distance based cel shading is applicable and efficient.

Related Work

There is tons and tons of information present as to how to calculate cel shading and how to get that cartoon shader feel, but there is little to no information as to applying cel shading with distance as a technique. Some works look like they are using distance based cel shading, but they either don't say that they are or they are applying some other technique to achieve a similar goal.

Figure 1: Gameplay of *Legend of Zelda: Wind Waker* by Nintendo
http://www.zeldadungeon.net/Zelda09/Walkthrough/03/045.png

Looking at this image you can see in the distance there are two objects that are greyed out. Now this could very well be done with distance based cel shading because those objects are far away from the camera, but they could also just as well be shading anything that is not in the immediate vicinity of the player. After doing a lot of research into how The Legend of Zelda: Wind Waker was actually made, all that is mentioned is the fact that cel shading is used, but no information on depth based shading was present. Let's say for the sake of discussion that the aforementioned photo does not actually use distance based cel shading, let's say that they use more passes to filter and blur objects in the distance. This approach would be much slower and more computationally expensive compared to our technique. With our technique you would be doing the most amount of work for only the objects in the foreground, the background objects would actually be computed and rendered a lot faster because of the lower levels of shading needed. In the approach above the background objects may actually be taking longer to computer because of their blur and darker shading applied.

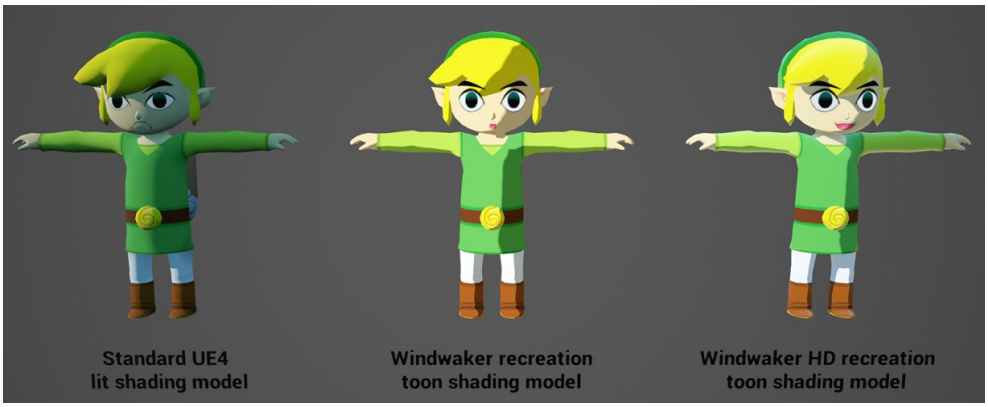


Figure 2: Renderings of *Legend of Zelda: Wind Waker* by Nintendo

The image above is again from *Legend of Zelda: Wind Waker.* It shows the process of toon shading from a 3D model to the classic game's shaded model and finally to the model of the HD release of the game which added spectral highlighting to the classic toon shader. We picked Wind Waker as one of our main sources of inspiration because of how popular and well known it is. This meant that there were many examples we could find that are related to our work.

Problem Solution

Every time an object is being drawn to the screen it needs to first know how many "bins" of shading it is going to need. The way that we calculate this is my having a function that determines the distance each object's origin is from the camera. After the VAO for the object has been specified we call this function and return the a constant divided by this result. This ensures that closer objects have a higher number of shading levels than those further from the camera. This is then attached to a uniform for every shader that we want to use cel shading. Obviously things like the wireframe and control points do not need to be cel shaded at all so this uniform is never attached to them. Our cel shader takes place inside of the blinn-phong shader

we have. This may seem counter intuitive at first because blinn-phong shading is used for more realistic shading whereas cel shading is intentionally used for a cartoon look. However, we use blinn-phong to calculate shading like normal - which gives us the light as well - and then clamp all of those values to the desired bin. This way we are placed shadows and the lights in the correct place, while still achieving our goal. Additionally, having the specular highlight is useful to create another dimension. This is generally a stylistic choice and looks best with only one level of shading via cel shading. An example is shown below where the smallest dot on each teapot is the specular highlight.

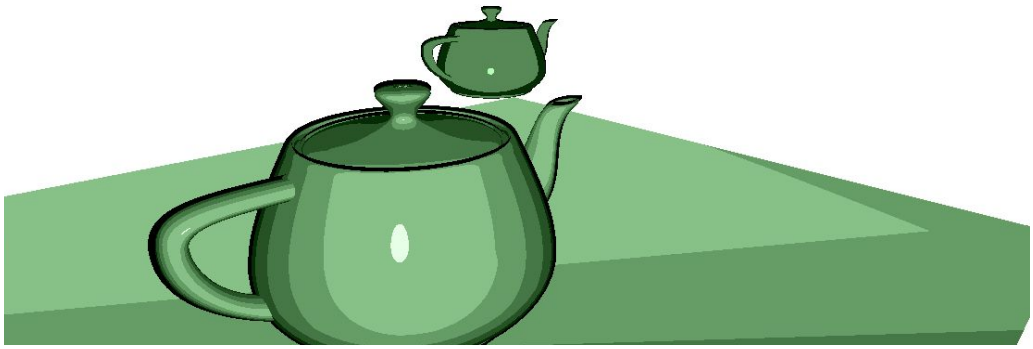


Figure 3: Demonstration of teapot cel-shading based on distance to camera

The silhouette shading was done in a different way as described in the OpenGL 4.0 Shading Language Cookbook. The method mentioned uses triangle adjacencies to compute which edges have opposite facing triangles attached and then extrudes a quad from that edge. The method implemented instead is a two pass technique. This is because the teapot model is based on bezier patches and thus, the triangles adjacency does not work as intended. The two pass technique is as follows:

1. Render the model with front face culling enabled and with black as the only fragment out color. The model is also scaled by 1.02.
2. Render the model as normal (also unscaled).

This method works well enough, but there is one major issue. In the following image, next to the spout of the teapot, there is a gap between the silhouette and the teapot itself. This completely breaks the illusion of drawn lines. A solution would be to, instead of scaling the image in the source code, draw quads that are extruding from the teapot in the geometry shader along the normal. This way the silhouettes always originate at the teapot. This also eliminates the need for a two pass shader as this extrusion can be done in a single pass. In other words, if the vertex is back-facing then extrude a black quad from that vertex.

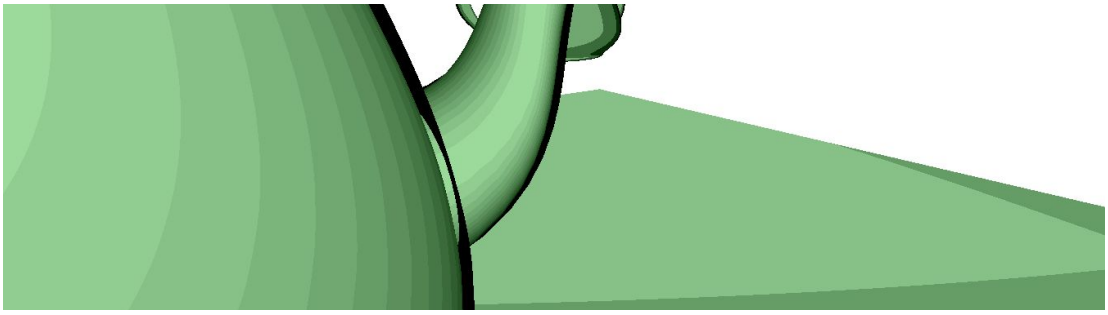


Figure 4: Demonstration of the silhouette separated from the teapot

Conclusion

In conclusion we were very successful in our implementation. We were able to achieve our goal of having distance based cel shading, and we know that we got it to work because we placed two objects into our scene, and when you move the camera close to one of them , you can see a high level of detail present in the foreground, but the background object has low details and bins. We were unsuccessful in getting silhouettes to work with adjacency lists but we did find a solution through multipass rendering. We render both objects as purely black and slightly larger than normal, then we render them again normally and with normal coloring, which produces a black outline around them. In the future it would be best if we used models that did not involve bezier surfaces, but since we were working with previous assignments we had to do workarounds for silhouettes. If we were to use other objects it would make rendering look better and more efficient.

Works Cited

"How to Outline a 3D Object with a Cartoon Style Stroke?" *Shaders - How to Outline a 3D Object with a Cartoon Style Stroke? - Game Development Stack Exchange*. Web. 10 May 2017.

"Igor Dykhta." *Sun and Black Cat - Computer Grafics & More*. Web. 10 May 2017.

Kinkley @yelknik, Jonathan. "The." *The A.V. Club*. 16 Dec. 2014. Web. 10 May 2017.

Wolff, David. *OpenGL 4.0 Shading Language Cookbook: Over 60 Highly Focused, Practical Recipes to Maximize Your Use of the OpenGL Shading Language*. Bimingham, UK: Packt, 2011. Print.

Hutchins, Adam. *Advanced Real--Time Cel Shading Techniques in OpenGL* (n.d.): n. Pag. Rensselaer Polytechnic Institute (RPI). Web.