Class Notes: Texture Mapping

Ben Davis, John Lopez, Kevin Patterson

May 7, 2019

1 A Brief Review of Illumination

Recall that if we were to mathematically define the illumination portion of the rendering pipeline, for a given color, c, it would look like the following:

$$I_c = C_c(a_c + d_c(\overrightarrow{n} \cdot \overrightarrow{l}) + s_c(\overrightarrow{n} \cdot \overrightarrow{h})^m + E \text{ Where,}$$

- C_c represents a base color
- a_c represents the ambience value
- d_c represents the distortion value
- $(\overrightarrow{n} \cdot \overrightarrow{l})$ represents reflection
- $s_c(\overrightarrow{n} \cdot \overrightarrow{h})^m$ represents specular light
- E represents the values from the environment

What if we already had an image to project onto an object, instead of having the rendering pipeline create it for us? This is the purpose of **texture mapping**.

2 What Are Textures, Really?

Textures may best be explained as a pattern of colors that create a desired image. The pattern may start off on a flat plane and be applied to a 3 dimensional shape, or vice-versa.

Figure 1: A label is an example of a 2D texture applied to a 3D surface.

Texture mapping is a technique used to create the images seen on billboards. Because the resolution of the desired image is thousands of times smaller than the final product, texture mapping is crucial to ensure the desired image.

Figure 2: Some textures must be applied to higher-resolution surfaces.

1. We define the significant points in our image as a series of **texture coordinates**. By abiding by these texture coordinates, we can replicate the desired image with accuracy.
2. When the texture coordinates are carried over to the desired image, the desired colors need to be looked up as they are rendered.
3. **Bilinear interpolation**, which was discussed in the notes about shading, is used to carry the texture coordinates over to the final image.

3 Challenges of Texturing

Suppose we are attempting to texture a flat image using a spherical map, as depicted in the image below:

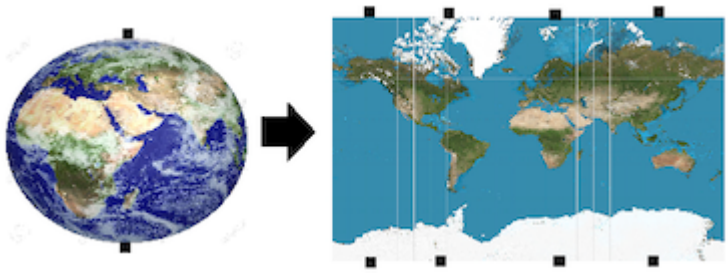

Figure 3: It isn't trivial to map a 2D texture onto a 3D warped object such as a sphere.

Although our surface may be textured easily, how do we accurately map the image surrounding the poles? In a single frame, this issue might not be that prevalent, but in an animated object (such as a globe), multiple points might be projected.

Mathematically speaking, how do we properly ensure that all of the x, y, and (possibly) z values transfer over from one model to another?

$$f(X_T, Y_T) = f(x, y, z)$$

We have three potential solutions.

- We pick a single point to represent multiple points.
- We take the average of our given points for a desired area:
    1. We "zoom out" of our desired area so that everything appears smooth.
    2. We take the values of our colors at given points.
    3. We apply these values as textures.
- We perform **anti-aliasing** (see the class notes on Scan Conversion for more details).

4 Texturing Pipeline

These are the main steps used to texture a surface:

1. Define points in space (**x**, **y**, **z**) that we want to map the texture to.
2. Place corresponding coordinates on the texture and image.
3. t_x, t_y, t_z get projected to image space, meaning that the result won't be flat.

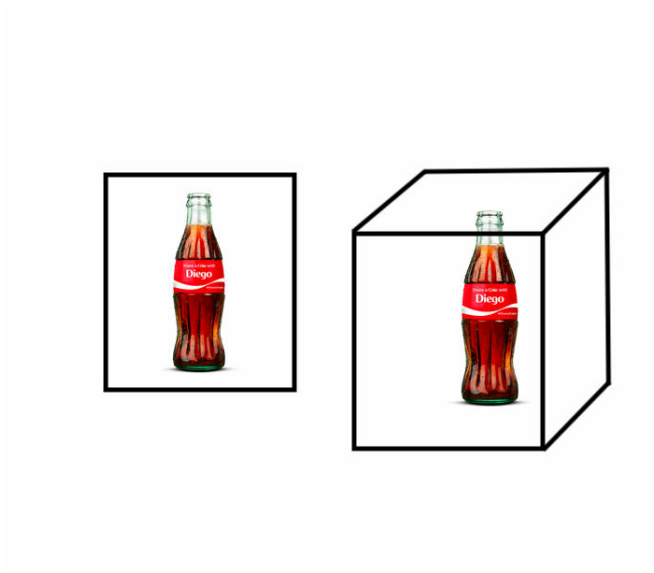

Figure 4: The letters on the label as they appear in image space will pop out a bit because of the added dimension that allows for movement in the z-direction.