TRANSFORMS AND TEXTURE MAPPING 2

CS184: COMPUTER GRAPHICS AND IMAGING

Feb 1-2, 2021

1 Sampling

1. What is the connection between applying a box blur to an image and supersampling each pixel within the image?

   Solution: Application of a 1-pixel box blur with dimensions x by x is equivalent to x by x supersampling on each pixel, as both compute the average value of each pixel by subdividing it and computing the average of those subpixels.

2. We have two wheels that rotate at different speeds with different numbers of spokes that we want to record with our camera.

   - **Wheel A** has 4 spokes and rotates at a rate of 6 rotations per second.
   - **Wheel B** has 6 spokes and rotates at a rate of 5 rotations per second.

   What is the lowest integer frame rate (in frames per second) we need on our camera to avoid any aliasing effects?

   Solution: In 1 rotation, a wheel with n spokes shows the same "image" n times, so if that rotation took 1 second, it would have a frequency of n hz. Knowing that:
   Wheel A has a signal frequency of $\frac{4}{rot} * 6\frac{rot}{s} = 24hz$.
   Wheel B has a signal frequency of $\frac{6}{rot} * 5\frac{rot}{s} = 30hz$.
   To avoid aliasing, using the Nyquist theorem, we need to sample at a rate more than double the highest signal frequency - B's in this case, so we need a rate faster than 60 fps, giving us 61 fps.

2 Basic Transforms

1. Which of the following transforms preserves the distances between every pair of points on an object? (Rotation, Translation, Scaling, Reflection, Shearing)

Solution: Rotations, translations, and reflections (and their combinations) all fulfill this property. They are what we call isometries or rigid transformations (as if we were manipulating an object that cannot be bent or broken).

2. Write the transformation matrix for a $30°$ counterclockwise rotation (assuming homogenous coordinates in 2D).

Solution:

$$\begin{bmatrix} \cos\frac{\pi}{6} & -\sin\frac{\pi}{6} & 0 \\ \sin\frac{\pi}{6} & \cos\frac{\pi}{6} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. Write the transformation matrix for a 2D object that is reflected across the y-axis, then translated up by 1 unit, to the right by 3 units, then rotated around the origin by $90°$ counterclockwise. (Leaving it as a product of matrices is fine).

Solution:

An important thing to remember here is that the transformation matrices are multiplied right to left. Keeping the order correct is important!

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -1 \\ -1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

3 Texture Mapping

3.1 Barycentric Coordinates

Since triangles form the basic building blocks of most 3D models, we use a special coordinate system to locate points within a triangle relative to its vertices.

The first step in mapping a texture onto a triangle is to convert the screen pixel to barycentric coordinates. These coordinates (α, β, γ) can be thought of as the weights being as-

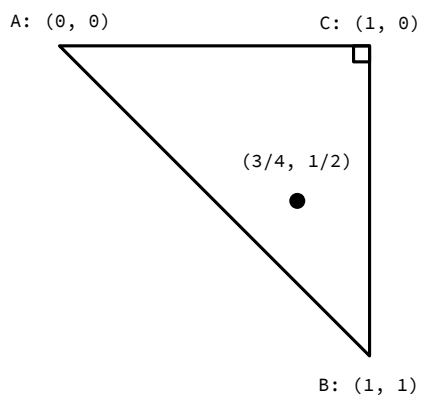
signed to each vertex such that the weighted average of the vertices forms a screen-space coordinate. That is,

$$(x, y) = \alpha A + \beta B + \gamma C$$

1. What happens if α, β, or γ is less than zero?

Solution: If any of the coordinates is less than zero, then the point will lie outside the triangle. As long as the coordinates all sum to one, however, the point should still lie in the same plane as that formed by the three barycentric coordinates.

2. What is the barycentric coordinate of the point corresponding to the screen-space coordinates $(x, y) = (\frac{3}{4}, \frac{1}{2})$ in the following diagram? (Hint: You don't have to use the closed-form solution, you can figure it out from just the definition of barycentric coordinates).



Solution: We can form a system of equations to solve for our unknowns (α, β, γ), based on our constraints that they sum to 1, and the x and y coordinates of the given point:

$$\alpha + \beta + \gamma = 1$$
$$x : 0\alpha + 1\beta + 1\gamma = \frac{3}{4} \Rightarrow \beta + \gamma = \frac{3}{4}$$
$$y : 0\alpha + 1\beta + 0\gamma = \frac{1}{2} \Rightarrow \beta = \frac{1}{2}$$

Plugging in the last constraint into the second gives us $\frac{1}{2} + \gamma = \frac{3}{4} \Rightarrow \gamma = \frac{1}{4}$, and plugging that into the first equation gives $\alpha + \frac{1}{2} + \frac{1}{4} = 1 \Rightarrow \alpha = \frac{1}{4}$, so $\alpha, \beta, \gamma = \frac{1}{4}, \frac{1}{2}, \frac{1}{4}$.

3. Suppose we specify colors at each of the vertices in order to color the triangle. If the

RGB values of (A, B, C) are (`(1, 0, 0),(0, 1, 0),(0, 0, 1)`), then what is the interpolated color at the selected point above?

Solution: `(0.25, 0.5, 0.25)`

3.2 Texture Coordinates and Mipmaps

Now that we have the barycentric coordinates (α, β, γ), we can convert them to texture-space coordinates (u, v). To do this, we use the same interpolation trick as we did with the colors above - we specify coordinates A_{uv}, B_{uv}, C_{uv} on the texture itself, then use our barycentric weights to compute $(u, v) = \alpha A_{uv} + \beta B_{uv} + \gamma C_{uv}$.

Sometimes pixels lie in-between texels, and a pixel may cover an area much larger or smaller than a single texel. When this occurs, aliasing artifacts are often visible when textures are actually rendered. One way of dealing with this issue is to filter while rasterizing, but that can often be computationally intensive. Instead, we often store prefiltered textures in the form of a mipmap to approximate the correct result and reduce aliasing.

1. The resolution of a mipmap at level 3 is what fraction of the size of the original texture?

   Solution: If the original texture has a size of $(128, 128)$, then the mipmap at level 3 will have a size of $(16, 16)$. This is $8^2 = 64$ times smaller.

2. A mipmap takes up what fraction of the original's size in memory?

   Solution: The storage overhead of the entire mipmap (including the original texture of size x) is $\frac{4}{3}x$. To see why, look at this slide.

3. How do we determine which level of a mipmap to use for a particular screen-space pixel (x, y)?

   Solution: We start by taking the pixel in question (x, y) and mapping it into texture coordinates $(u, v)_{00}$. We also map $(x + 1, y)$ into $(u, v)_{10}$ and $(x, y + 1)$ into $(u, v)_{01}$. To compute the mipmap level, we find:

   $$\left(\frac{du}{dx}, \frac{dv}{dx}\right) = (u, v)_{10} - (u, v)_{00} \qquad \left(\frac{du}{dy}, \frac{dv}{dy}\right) = (u, v)_{01} - (u, v)_{00}$$

   Whichever delta is larger will dominate the mipmap selection, so to find the larger squared distance we do:

$$L = max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

$$D = \log_2 L$$

4. How can we combine values from two neighboring mipmap levels and why would we do that?

Solution: We can interpolate between levels - an example of this is Trilinear Filtering, where we first interpolate between pixels of a single level and then between levels. This is done to make the transition of mipmap levels smoother.