CSC 240 Computer Graphics
Day 18: More Mapping

Nick Howe
Smith College
Lab 10 Review

How do we set up the $(u, v)$ coordinates for the pyramid?
• Adjust coordinates for triangular faces (some outside 0-1 range)
• Split bottom face diagonally
Lab 10 Review

Order of vertex listings must match between `pyramidGeom.faces` and `pyramidGeom.faceVertexUvs`.

ABC is not the same as ACB, BCA, CBA, etc.
Your Questions

Q. I don’t understand why we need to normalize 1, u, and v by z — can you elaborate on that?

A. Texture should appear smaller when it is farther away, and larger when it is close. If you don’t normalize by 1/z, then the texture gets drawn at the same scale everywhere in the polygon – too big for the farthest parts, and too small for the nearest.
Perspective Correct Texture Mapping

This span is smaller

This span is bigger

Camera: (0,0,0)

Equally spaced texture receding in space
Q. How do we get the affine view? What is that a result of?
A. The affine view is what you get when you apply the texture at equal scale everywhere regardless of z value.

Q. In question 2, what exactly are u1, z1, etc. referring to? Are these coordinates of a polygon?
A. They are the coordinates of the triangle corner – u₁ and z₁ refer to the first corner, u₂ and z₂ to the second, etc.
Q. I'm not entirely clear on the difference between a normal map and a bump map. And how do we apply both at the same time?
A. You don’t want to apply both at the same time. A bump map represents surface height; the normal is inferred by taking a derivative. For the normal map, the normal vector is explicit.

Q. Why are bump maps less flexible than normal maps?
A. They have fewer degrees of freedom.
Q. For shadow mapping: can you reexplain the step where you convert the coordinates to the light’s frame of reference (step 5 on slide 14)?

A. We know both the camera and light coordinates in terms of world coordinates. Thus we can compute a transformation to convert.

Q. Also, why is this step completed after recording the coordinates from the camera’s point of view (step 4) instead of being executed after step 2 (when the z is recorded at each point from the light’s point of view)?

A. We’re converting camera coordinates to light’s frame. We don’t have those coordinates until step 4.
Shadow Mapping Example

1. Render scene from light.
2. Record z at each point
3. Render scene from camera
4. Record (x,z) at each point
5. Convert (x,z) to light’s frame of reference
6. Compare to stored z values

2D example (XZ plane) orthographic projection
Q. Can you explain in more detail why the different kinds of lights require different projections?

A. For directional light, the rays are parallel, as are sight lines in orthographic projection. For a point light, the rays come together at the light source, like the sight lines in perspective.
Q. I don't understand the projection diagram for perspective -correct. Can you explain why the span is bigger/smaller? Can you explain the reason and logics behind the interpolation and the math?

A. The span changes in size for the same reason that railroad ties decrease in size with distance from the camera.
Q. Why perspective makes x&y shrink, but not z? Why the affine image looks like that?(The image in the middle)

A. After perspective projection, there is no z. The affine image computes the texture mapping wrong; this is the point.

Q. I don't quite understand how we get that formula for \( u^* \) and \( v^* \). Would you please explain it in more detail?

A. As part of the perspective projection, we normalize the homogeneous coefficient. This introduces a \( 1/z \) factor in all the XY coordinates.

\[
P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} xf/z \\ yf/z \end{bmatrix}
\]

\[
u = \frac{\alpha u_1 + \beta u_2 + \gamma u_3}{\alpha + \beta + \gamma}, \quad v = \frac{\alpha v_1 + \beta v_2 + \gamma v_3}{\alpha + \beta + \gamma}
\]

\[
u^* = \frac{u_1}{z_1} + \frac{u_2}{z_2} + \frac{u_3}{z_3}, \quad v^* = \frac{v_1}{z_1} + \frac{v_2}{z_2} + \frac{v_3}{z_3}
\]
Your Questions

Q. Can you show visually what you mean when a polygon has the same z values? Is it just a 2D shape?

A. If all the z values are the same, the polygon is fronto-parallel to the camera. There is no perspective correction; all distances are equal.
Your Questions

Q. Why is the RGB value range for blue (128-255) half that of red and green (0-255)?

A. Think of the origin at (128,128,128). Numbers less than 128 represent negatives. Since blue represents the z component, a negative value would be pointing away from the camera.
Your Questions

Q. When should we better use normal mapping? When should we better use bump mapping?

A. They are pretty interchangeable, so use whichever is convenient. Normal mapping might give you slightly more realistic results.

Q. My eyes are kind of deceiving me and I can't tell if the first black and white normal map is raised out or pushed in. Also can we create a normal map with pressed in texture if z can only be greater than 0?

A. White is out, black is in. All that matters is the height relative to neighboring points, since that gives the normal vector. Remember: everything is rendered on the flat polygon; we’re just shading.
Q. How does the bump map know then where each bump is if it doesn't have x or y?
A. The bumps/normal are mapped onto the polygon using (u,v) coordinates just like ordinary texture.

Q. How did you calculate the normal vector for the last question aka what is 127.5?
A. I used the equations from slide 6.

Q. I enjoyed the coding section of this video
A. I’m glad!

\[
x = \frac{r}{127.5} - 1 \\
y = \frac{g}{127.5} - 1 \\
z = \frac{b}{127.5} - 1
\]
Q. What is scanline interpolation?
A. During a half-triangle fill, we work on one line at a time. Instead of the full barycentric calculation, it is more efficient to compute the two endpoints and do linear interpolation in between.

Q. Why does changing the geometry alter the shading of a pixel?
A. If we change the geometry, then the normal vector may change.

Q. Can you explain how the normal mapping gets from high-poly render to normal map to rendered surface?
A. Here’s a picture.
Your Questions

Q. Where do you find matching images (the png and jpg) like you had for the Earth object in your demo?

A. Search for “free 3d textures”

Q. Can you go over finding the normal vector, from a normal image?

A. Yes. (Keep in mind this is done automatically by software.)
For each component, divide by 127.5 and subtract 1.

Q. Is there a way we can have a small demo doing the shadow mapping?

A. Yes. [link]https://repl.it/@nhowe/CSC-240-Shadow-Demo
Your Questions

Q. Can you explain the difference in using the perspective and orthographic projections for computing shadows? How much is a significant difference for comparing camera-visible points and light-visible points? And does it depend on canvas size?

A. For DirectionalLight, rays are parallel and we use an orthographic projection. For PointLight, rays radiate and we use perspective.
Q. Please explain more about "compare depth of camera-visible point to light-visible point" on the slides.

A. That’s what I was demonstrating with the 2D example. You render the scene from both the camera and the light, and compute 3D coordinates of the visible points. Where they are the same, the pixel is lit; where they are different, the pixel is unlit.
Q. How to convert camera frame to light frame?
A. There are 3D transformation matrices that convert world coordinates into camera or light frame coordinates.

\[ P_L = T_L P_W \quad \text{and} \quad P_C = T_C P_W \]

\[ P_L = T_L T_C^{-1} P_C \]

Q. How does the program render shadows with different darkness levels when the shadow map seems to only tell whether there is a shadow?
A. There may be other lights on the surface (ambient, etc.)
Activities

• Handout 6: shading calculations
• Handout 7: texture
• Lab 11 (complete on your own): install Blender

Icebreaker: If you had a boat, what would you name it (and why)?