COS426 Final Exam Q&A

1. **Rendering Equation [Kajiya 86]**

- compute radiance in outgoing direction by integrating reflections over all incoming direction
- $L_0(x', w') = L_e(x', w') + \int_\Omega f_r(x, w, w')(w \cdot n) L_i(x', w) dw$
- $\cos\theta = \vec{w} \cdot \vec{n}$
- applied recursively
- questions: explain each term, what they are, know incoming/outgoing directions; can have different parametrizations;
- rendering equation (2) with visibility term and attenuation with distance;

2. **Radiosity**

- an integral of radiance; light leaving all directions combined;
- a particular way to model light when all the surfaces are diffuse; attenuation with outgoing distance is constant; derived with rendering equation (2)
- all energy leaving a surface; integral over all area
- can be approximated by discretizing the surfaces into elements; B_i is the amount of radiosity leaving i; ρ is the reflectance coefficient;
- need to explain what the equation means, what are the terms, and how it leads to system of equation;

3. **Phong model**

- K_A, I_A is the terms that we assume represent light bouncing around;

4. **Aliasing**

- visual artifacts due to under-sampling; you introduce a pattern that's not in the original data;
- solution: use a low-pass filter to eliminate frequencies higher than 2x your sampling rate, so you don't have super high frequencies; such as Gaussian functions;
- use **mip-map** as the data structure to accelerate; pre-computed, optimized sequences of textures, each of which is a progressively lower resolution representation of the same image. The height and width of each image, or level, in the **mipmap** is a power of two smaller than the previous level.

5. **Shading Models**

- flat: one lighting calculation for every polygon
- Phong: linear interpolation of vertex normals for every pixel within
- Gouroud: linear interpolation of intensities

6. **Dither**

- you are going to use spatial resolutions of your eyes, to trade off for color/intensity resolution; and depends on what visual artifacts I want

- ordered dither: trying to make sure things don't clump
- Floyd-Steinberg: error diffuses

7. **3D Rendering Pipeline/Polygon Pipeline**

- sequence of transformations that happens when we go from a point to the screen;
- viewing transformation takes to camera view;
- projection matrix puts it on the screen;
- viewport transformation turns that into pixels;

8. **Curves and Surfaces**

- computational ways to describe curves with desired properties: smoothness, intersect, represent
- **Bezier** curve: control points, segments defined by 4 control points each; positions of control points + base function define all shapes;
- reason about continuity by looking at the control points
- they are splines, which means they are broken into segments; they are usually cubic, meaning blending functions are cubic; all the differences boil down to blending functions
- **B-Spline**: doesn't interpolate the control points, all within; share more than one control point for neighboring segments, blending functions are designed that so that joints are guaranteed to have C^2 continuity; (cd: Bezier curve only has one point shared between neighboring segments)