Final Exam Review Notes
COMP 575, Fall 2007

Lecture 1 - Course Intro:
• Nothing

Lecture 2 - Rasterization Intro:
• Basics of how a CRT works
• Main differences between CRTs and LCDs
• Differences between raster and vector displays

Lecture 3 - Cameras & Raytracing Info:
• We covered all of this again later in the semester

Lecture 4 - Math Basics:
• What are vectors, matrices, rays, planes, etc.
• Know how to do dot and cross products

Lecture 5 - 2D Transforms:
• What is a vector space?
• How can you use 2 vectors to define a coordinate system in 2D?
• How do you construct translation, rotation, scaling matrices?
• Why do we use homogeneous coordinates?
• What is the other way to implement transforms, if you don't use homogeneous coordinates?
• How does the order of transformations affect the final result?

Lecture 6 - 3D Transforms:
• How do you construct the 3D versions of transform matrices?
• How do you rotate around arbitrary directions, or arbitrary points and directions?
• What is the difference between left-handed and right-handed coordinate systems?

Lecture 7 - OpenGL I:
• How is OpenGL a "state machine"? What are some kinds of information that are stored as OpenGL state?
• How are all objects represented in OpenGL?
• Be able to discuss or create very simple OpenGL commands. For example, given a set of points, what is the difference between drawing it with GL_TRIANGLES vs GL_POLYGON?
• Why might you use GL_TRIANGLE_FAN or GL_TRIANGLE_STRIP instead of just using individual triangles?

Lecture 8 - OpenGL II:
• Nothing

Lecture 9 - Representing Geometry:
- What are the relative advantages and disadvantages to representing geometry procedurally vs. as tessellated polygons?
- What are some reasons why we prefer triangles to other kinds of polygons?

Lecture 10 - Lighting and Shading I:
- How do diffuse and specular surfaces look different?
- What are the characteristics of the surface that cause these differences in appearance?
- What are the different classes of lights we consider in computer graphics? What are some real world examples of the different kinds?
- How are point lights like directional lights? How are they like spot lights?
- What are the 3 terms in the Phong lighting model? How are they computed? (NOTE: You don't necessarily have to know the whole equation including the attenuation terms, but you should know, for example, that the diffuse term is modulated by Lambert's cosine law, or that the specular term is modulated by the similarity of the reflection vector from the light and the eye vector)
- In plain English, what does Lambert's cosine law mean?
- Why do we need the max(0, N•L) term in the diffuse calculation?
- How is the color of a polygon determined in flat, Gouraud, and Phong shading? Which looks the best? What are the artifacts associated with each?
- Which shading models are available in OpenGL?

Lecture 11 - Lighting and Shading II:
- What are some reasons a user might prefer non-photorealistic rendering (NPR)?

Lecture 12 - Vertex Processing I (Viewing):
- What is produced by the vertex processing stage?
- What are the steps involved in vertex processing?
- What is the difference between perspective and orthographic projections? What is the mathematical operation that causes this difference?
- What are some reasons you might prefer an orthographic projection for some applications?
- Why does OpenGL have clipping planes?
- How do you build a projection matrix that does perspective divide? Why does this work?

Lecture 13 - Vertex Processing II (Clipping):
- Why do we bother with clipping?
- How does Cohen-Sutherland line clipping work? Does it extend easily to 3D? How?
- How can you tell whether a point is inside or outside a polygon?
- What are some of the reasons polygon clipping is more complicated than just doing line clipping on each edge?
- Step through the clipping of a polygon using the Sutherland-Hodgman method.
- What is the main difference between the output generated by Sutherland-Hodgman and Weiler-Atherton polygon clipping?

Lecture 14 - Rasterization I (Line Drawing):
- What is the output of the rasterization stage?
- What are some of the difficulties associated with line drawing? What are some of the ways we improved performance from the full screen brute-force method?
- Why is Bresenham's algorithm so fast?
- What is aliasing, and why is it a problem for us?
- How does supersampling antialiasing work? Why does it help solve the aliasing problem?

Lecture 15 - Rasterization II (Polygon Drawing and HSR):
- What is ratio antialiasing? What are the benefits and detriments relative to supersampling?
- How does scanline polygon drawing work? How does this build on line drawing?
- How does floodfill work? What is the main problem with using floodfill naively?
- What is backface culling? How could you implement it efficiently? What does it buy us, in terms of performance?
- What are the two types of depth culling we discussed?
- How does a pipeline that supports Z-Buffering differ from one that uses a painter's algorithm? (Output of the clipping stage, additional memory needed, additional processing, etc.)

Lecture 16 - BSP Trees and Texture Mapping:
- What is a BSP tree? Why is it efficient to use BSP trees to implement a painter's algorithm?
- Under what circumstances will a BSP tree perform poorly?
- In texture mapping, do we map the image into world space coordinates, or do we map the vertices into the image? Why?
- What is mip-mapping, and why do we need to use it?

Lecture 17 - Mapping II:
- Understand the mip-mapping example.
- How do the different texture mappings work? What sort of artifacts appear if you attempt to use an "incompatible" mapping, like a cylindrical mapping on a sphere?
- What is environment mapping, and why do we implement it with cube maps? What visual effects can we fake in OpenGL with this technique?
- What is bump mapping? What is displacement mapping? How do they differ? How can you tell the difference between a bump-mapped and a displacement-mapped sphere?

Lecture 18 - Cameras and Light Transport:
- Why does the image appear upside-down with a pinhole camera?
- An ideal pinhole camera is always perfectly in focus. Why isn't this true with real cameras? (Ignore lens effects at this point; I'm just asking about the aperture)
- Given that an ideal pinhole camera is so perfect, why can't we just use that?
- Why do you see depth of field effects with real cameras?
- Why do you see motion blurring with real cameras?
- Why, in ray tracing, do we shoot rays from the camera instead of from the lights?

Lecture 19 - Ray Casting I:
- Why do we use rays in parametric form [R(t) = P + tD] in our ray caster/tracer?
- What about our ray tracer lets us only do the full ray computation once (for the V0 ray), and then just do one matrix multiply to find any other ray?
- Why doesn't it cost us anything extra to do Phong shading in our ray tracer?

Lecture 20 - Ray Casting II:
- What are some examples of camera intrinsics? Camera extrinsics?
- What are the 3 vectors that are used to create the camera matrix? What values do you need to know in order to find these vectors?

Lecture 21 - Ray Tracing I:
- What causes a point to be in shadow?
- In a ray tracer, how can you test if a point is in shadow?
- What are the special issues you face when testing for shadows against the different types of lights (point, directional, spot, area)?
- What is the difference between the equation used for the "reflection" vector in the specular lighting calculation, and that used for computing reflection vectors?
- In your ray tracer, when would you stop generating reflection rays and return a color? (NOTE: There are 4 cases that I can think of)

Lecture 22 - Ray Tracing II:
- What are the inverse transforms for rotation, translation, and scaling?
- What are some reasons we wanted to use inverse transforms on the rays instead of regular transforms on the objects?
- Why shouldn't you normalize the direction vector in your new ray, after you apply the inverse of any object transforms to it?
- Why does doing things this way result in the wrong normal? What can we do to fix it?
- What causes the "reintersection" problem, and how can we fix it?

Lecture 23 - Advanced Ray Tracing:
- What are some ways we can make our ray tracer faster? Specifically, how can we speed up object intersections?
- What is the basic idea behind bounding volumes? How about spatial data structures? Is there any reason we can't do both?
- What is the basic idea of distribution ray tracing? What are some of the different effects we can get from this basic method?
- Of these new effects, which require new components to be added to the ray tracer? What needs to be added?

Lecture 24 - Radiosity:
- Understand Heckbert's light transport path notation. For example, what kind of renderer does LD*E refer to? L(D|S)*E?
- What effects can we simulate with radiosity that we could not handle before?
- What can we do with ray tracing that we cannot model with radiosity?
- Why is ambient light a "hack"? Why don't we need it anymore if we use radiosity?
- Radiosity treats every surface patch as a potential emitter. Why does it do this?
- Radiosity solutions can take a really long time to compute. Is this a problem when viewing environments that use radiosity for lighting?

Lecture 25 - Path Tracing and Photon Mapping:
- What effects can these techniques model that neither the ray tracer nor radiosity could handle?
- What is the basic idea of path tracing? Why does it work?
- Why are bi-directional path tracing and metropolis light transport improvements? What kinds of scenes do they help the most? (Or, alternatively, what kinds of scenes does path tracing have the hardest time with?)
- Photon mapping is a multi-pass algorithm. What happens (very roughly) in each pass?

Lecture 26 - Color Spaces and High Dynamic Range:
- What colors look the brightest to the human eye? Which are the least bright?
- How many types of cones do humans have? Roughly, what colors do they correspond to? How does this relate to the design of, say, TVs?
- Name at least one limitation of the RGB color space.
- What is an example of an additive color space? A subtractive?
- How much dynamic range does an average computer monitor have? How much can real world scenes have? (Just an orders of magnitude answer here is fine)
- How can we capture the dynamic range of a real world with a normal (limited range) digital camera?
- How can we display high dynamic range images on a regular display?

Lecture 27 - Image Based Rendering:
- Nothing

Lecture 28 - Grab Bag:
- Name an image processing function that can be implemented as convolution. What kernel could you use to implement this function. Why does this work; that is, why does convolution with this kernel produce the effect you describe?
- A blurry image only has detail at (high or low; choose one) frequencies. How can you combine this image with the original one to get an image only containing details at the other end of the frequency spectrum?
- Basically, how do particle systems work?
- What was a really cool extension of particle systems that I discussed in class?