Overview

- Consider some surface in your environment – a piece of clothing, the carpet, a key, a mirror. What factors determine the color that we actually see at various points on that surface?

- Computer graphics is all about making practical simplifications in order to portray objects in a way that is feasible and that meets the artist's goals as well as possible. For each of the factors you list above, how might you simplify them to make them easy to compute?

- Sketch a simple model of the standard graphics pipeline.

- It is difficult to discuss the graphics pipeline without discussing the use of buffers – the color buffers, depth buffer, accumulation buffer, stencil buffer and so on. Why is there such an emphasis on buffers such as these in the standard graphics pipeline?

- For each of the buffers listed above, describe briefly and at a high level why they are useful.

- The standard graphics pipeline is built around certain assumptions – namely (1) that we want to render a collection of triangles as quickly as possible, and (2) that the processing of each triangle must be kept largely independent of the processing of all other triangles. What real-world effects are difficult to achieve in the standard graphics pipeline, due primarily to these assumptions?

Basic Rendering

- Graphics programming is often described to follow the Model-View-Control (MVC) paradigm. What is meant here by model? by view? by control?

- List all of the information required to specify a wireframe rendering of a simple cube. List model related information and camera related information separately. Draw a sketch to illustrate the various parameters in your lists.

- Show how to use the rule of similar triangles to compute a projection of points on an object onto an image plane. (Give the equations you would use to accomplish the projection.) BONUS: How might we maintain depth information when performing this projection?

- BONUS: Suppose you have a room with a single point light source in the ceiling and having several objects scattered throughout the room. We can compute shadows that are cast on the floor of the room by placing a camera at the position of the light source pointing down toward the floor and projecting all objects in the room onto the floor (i.e., using the floor as the image plane). Sketch such a room, set up a coordinate system, and derive equations to compute this projection. Why is this only an approximation of the true shadows that would be observed in a similar real-world scene?

OpenGL

- Why does OpenGL make heavy use of state variables, for example maintaining a current color and a current transform?

- Explain the difference between the following two sequences of operations. Give a specific example and sketch the expected results to illustrate your point.

```
glRotatef(angle, x, y, z);
glScalef(x, y, z);
drawGeometry( );

glScalef(x, y, z);
glRotatef(angle, x, y, z);
drawGeometry( );
```

- BONUS: Describe the information that passes from the CPU to the GPU. From the GPU back to the CPU. Describe several ways in which you can make the transfer of data from CPU to GPU more efficient.

- What is the difference between using immediate mode rendering (where a display list is not used) and using a display list? Explain your answer in terms of the information transferred from CPU to GPU in each case.

- Why is it useful to have a near clipping plane, even if that clipping plane is located very very close to the camera? Hint: consider the projection equations. What exact situation would lead to a divide by zero error in the projection equations? How does having a near clipping plane avoid this problem?

- Why is it useful to have a far clipping plane as close to the camera as is feasible? Hint: Consider how values in the depth buffer may be interpreted based on the locations of the near and far clipping planes.

Math

- What is the difference between a point and a vector?
- What is a right handed coordinate system?
- How do you compute the dot product of two vectors? Informally, what does the dot product do? Give an example where we need the dot product for Computer Graphics applications.
- How do you compute the projection of one vector onto another?
- What are the implicit and parametric forms of a line? Why are they named implicit and parametric?
- What is the implicit form of a circle centered at (x_c, y_c) with radius r?
- Write an implicit equation for a line given two points.
- Write a parametric equation for a line given two points.
- Suppose you are given a parametric equation for a line. Convert from this representation to an implicit equation for that same line.
- Given an implicit equation for a line, convert it to a parametric representation.
- Given a slope / intercept equation for a line, convert it to parametric and implicit representations.
- Give a situation in a computer graphics application where you would prefer a parametric representation of a line to an implicit representation.
- Give a situation where you would prefer an implicit representation of a line to a parametric one.

More Math

- How do you convert an arbitrary vector to a unit vector?
- How do you compute the cross product of two vectors? Informally, what does the cross product do? Give an example where we need the cross product for Computer Graphics applications.
- What are implicit and parametric expressions for a plane?
- Write an implicit equation for a plane given three points.
- Write a parametric equation for a plane given three points.
- Compute the intersection between two arbitrary lines. What is the best choice for representing each line: a parametric or implicit expression? Justify your answer.
- Compute the intersection between a line and a plane. What is the best choice for representing the line: a parametric or an implicit expression? What is the best choice for representing the plane: a parametric or an implicit expression? Justify your answers.
- Compute the intersection between a line and an arbitrary sphere. Which representation did you choose for representing the line and the sphere (implicit or parametric)? Justify your choice. What are the conditions for this intersection to exist?
- Be prepared to multiply two arbitrary matrices together. What conditions must be satisfied for this operation to be possible?

Meshes

- What are barycentric coordinates? Why are they useful in computer graphics?
- Given a set of 3 coefficients weighting vertices v1, v2, and v3:
    - What properties must be true for these coefficients to be valid barycentric coordinates?
    - What properties must be true for these coefficients to represent a point inside the triangle defined by v1, v2, and v3?
    - What properties must be true for the point to be outside the triangle, on the opposite side of edge v1-v2 from point v3?
- How would we interpolate vertex colors using barycentric coordinates?
- Given an point and a triangle, how do we find the barycentric coordinates of that point?
- Give the algorithm for Loop Subdivision. List some ways in which this algorithm can be improved.
- For what kinds of data representations is the marching cubes algorithm a good approach for generating surface meshes.
- In the marching cubes algorithm, what information do you need to know about each voxel in order to render the subsection of the surface mesh that may be contained within the voxel?

Curves

- Why do we use cubic splines in graphics, as opposed to line segments, quadratic curves, or higher order polynomials?
- How would you derive the expression for a Hermite spline as a function of its control points and tangent vectors? Use what you know about the spline, such as the fact that it passes through the first control point at a parameter value of zero, it passes through its second control point at a parameter value of one, etc.
- Write the matrix form for the Hermite spline.
- What is the geometry matrix (also called the basis matrix) for this spline?
- Suppose you suspected there was an error in your geometry matrix for the Hermite spline. How could you identify and fix that error? (Hint: you must show that the resulting cubic polynomial satisfies the constraints that we know for the Hermite spline. For example, one such constraint is that the spline should pass through the first control point at a parameter value of zero.)
- What are the blend functions?
- Define the term ‘continuity’ as used to describe splines in computer graphics. What are C^0, C^1, C^n, C^∞ continuity? BONUS: What is G^1 continuity? Give an example where G^1 continuity differs from C^1 continuity.
- Describe Catmull-Rom splines and explain how they relate to Hermite splines.
- For Catmull-Rom splines, give an expression for the tangent to the spline at a control point. Explain the reasoning behind this choice of tangent.
- Derive an expression for a Bezier spline, given the knowledge that it interpolates the first and last control points, and that the tangents at those points are: 3(p2-p1) and 3(p4-p3).
- What does it mean to say that a Bezier spline has the convex hull property? For this to be true, what must be true of the blend functions? Why is this property useful?
- Describe the differences between Hermite splines, Catmull-Rom splines, Bezier splines, and B-splines.
- Use the cubic polynomial expression for a B-spline to demonstrate that this spline has C^2 continuity at the join point between two sequential cubic segments.
- Given a description of any spline in matrix form, write out the full polynomial expression for the spline, list its blend functions, state whether it has the convex hull property, and determine whether it interpolates its endpoints.
- BONUS: How can you construct a C^∞ spline that interpolates a given set of control points? What are the advantages and disadvantages of such a spline for use in computer graphics?

Lighting, Shading, and Materials

- OpenGL makes use of a simplified illumination model -- *the Phong illumination model* (not to be confused with Phong shading). This model is designed to capture ambient, diffuse, and specular lighting effects. Describe the philosophies behind each of the ambient, diffuse, and specular lighting models used in OpenGL. What phenomena are they meant to represent?
- What information do you need to compute the color at a point due to ambient reflection? Write an expression that performs this color computation.
- What information do you need to compute the color at a point due to diffuse reflection?
- What information is needed to determine color due to specular reflection?
- Why can't we render a mirror correctly using the specular lighting model?
- Suppose you want to render the appearance of a green pepper under a chandelier in an otherwise well-lit room. Give reasonable estimates for all parameters needed to compute the intensity of a point on the surface of the green pepper (e.g., what are reflection coefficients kd, ks, etc.). Justify your choices for parameter values.
- Explain Gouraud shading.
- How do we get normals for Gouraud shading? At what points are the normals calculated?
- Explain Phong shading (again, not to be confused with the Phong illumination model!)
- How do we get normals for Phong shading? At what points are these normals calculated?
- How does Phong shading differ from Gouraud shading? Which is more computationally intensive, and why? Give some examples where you would expect to be able to visually identify the differences.
- What is the BRDF? The BRDF is a function of how many parameters? What are those parameters? Draw a sketch to illustrate your answer.
- Draw BRDF's that capture the basic components of the Phong Illumination Model: ambient, diffuse, and specular reflection.
- Describe a setup you might use to measure the BRDF of a surface.
- What is gloss? What does the BRDF look like for a glossy surface?
- What is the effect of surface roughness on the appearance of lambertian surfaces? (Hint: why does the moon look flat?) Explain the reason for this effect. BONUS: how do we model it?
- BONUS: Describe qualitatively the effect of Fresnel reflectance. What causes this phenomenon? How can we model it in a renderer?

Textures

- Describe conceptually how texture can be applied to a triangle in a scene. Draw a sketch to illustrate your answer.
- Suppose you have texture coordinates (u,v) for each vertex of a triangle. Explain how barycentric coordinates are used to determine the color at each pixel of a textured triangle.
- Texture maps can be used to modify a wide variety of different values that may vary over the surface of an object. List at least five different parameter values that may be controlled using a texture map (i.e., five different ways in which texture maps can be used to control object appearance).
- What is a bump map? a normal map? a displacement map? an environment map?

2D Transforms

- What is a linear transformation? Give an example of a linear transformation in R^2 (i.e., in two dimensions).
- Provide 2x2 matrices to illustrate non-uniform scale, rotation, and shear. Draw before and after sketches to illustrate the effect of your matrices.
- Create a 2x2 shear matrix that shears in the x and y directions simultaneously.
- Create a 2x2 matrix that reflects geometry about the x-axis.
- Create a 2x2 matrix that negates the x and y coordinates of all points.
- Create a 2x2 matrix that projects all points onto the x-axis. ..onto the y-axis.
- Create a 2x2 matrix that projects all points onto the line x=y.
- What is an affine transformation? Give an example of an affine transformation in R^2 (i.e., in two dimensions).
- What are homogeneous coordinates? How do we represent a point in homogeneous coordinates? How do we represent a vector?
- Give a 3x3 matrix representing a translation by (a, b).
- True or false: a linear transformation always maps the zero vector to itself. If this statement is true, how is that consistent with the fact that we can do translations using a 3x3 linear transformation?
- Give a 3x3 matrix that rotates all geometry about a fixed point (a, b).
- How can we tell that a 2x2 matrix in 2D or a 3x3 matrix in 3D is a rotation matrix? What must be true of all rows and all columns? How must the three rows or columns relate to one another? Give examples of matrices that are not rotation matrices and explain why they are not.

3D Transforms

- Give an example of a 4x4 matrix to do non-uniform scaling. Draw before and after illustrations of the results of your transformation.
- Give an example of a 4x4 matrix to do a shear. Draw before and after illustrations of the results of your transformation.
- Give an example of a 4x4 matrix to reflect all points across the x-y plane.
- Give a 4x4 matrix to rotate by angle a about the x-axis, then b about the y-axis, then angle c about the z-axis.
- Give an example of a shear matrix. Transform all points of a cube using this shear matrix. Transform all normal vectors of this cube using the same shear matrix. What is wrong?
- BONUS: If we transform all vertices of a surface using transformation matrix M, what matrix should we use to transform vectors normal to that surface? Note that we want to be sure that the normal vectors remain orthogonal to the surface after the transformation.
- Under what types of transformations is it fine to transform normal vectors by matrix M?
- BONUS: Under what circumstances is the matrix you derived to transform normals identical to M? Test that this is true for some example transformations.
- BONUS: Create a hierarchy of transformations to conveniently represent the geometry of an animated human character. Assume you are given parts of the body (e.g., upper arm) in their own local coordinate frames. The final transformation for each body part should result in a coherent human figure in a particular pose. You want to be sure that logical rotations (e.g. the rotation at the shoulder joint) are easily accessible in your hierarchy so that it is easy to re-pose the character as needed in an animation program, for example.

Camera / Projection

- Give a complete list of parameters required to specify how a given scene will be projected into an image (a complete list of camera setup parameters). Assume perspective projection and a view frustum centered along the line running from the camera position along the "Look Vector".
- Perspective projection can be accomplished through the following sequence of steps. You should be able to derive the transforms involved in any one of these steps.
    - Transform view frustum to canonical camera view.
    - Transform canonical camera view to canonical orthographic view volume.
    - Do perspective division.
    - Project into 2D.
    - Do a windowing transform to convert coordinates to pixel values.

GLSL

- As we discussed in class, when we write shaders, we are primarily concerned with changing the functionality of two parts of the graphics pipeline. Which parts of the graphics pipeline are they?
- Give at least three examples of effects you can create within the OpenGL framework only by writing your own shader.
- For each example, explain at a high level the vertex shader and fragment shader code that would have to be written to create a shader for such an effect.
- A toon shader takes the full range of color values that would be output for a diffuse shading of an object, for example, and replaces this range of shades with one or a few discrete colors (e.g., a green object may be shaded with light green, dark green, and black). Which part of the graphics pipeline would you replace to create this effect. Give pseudocode for a toon shader.

Color and Light

- The actual color of an object can be described as a distribution of light intensity over wavelength, as might be measured, for example, by a spectroradiometer. Why then, is it considered sufficient to represent color by a vector containing magnitudes of three light components such as red, green, and blue?
- What are metamers? Give details about what causes this effect.
- Suppose we have chosen particular shades of red, green, and blue light for a display. Describe an experiment that could be used to determine exactly what R,G, and B values should be used to portray a color with broad spectral components in such a way that it would appear identical on the display and in nature? This is the technique that was used in early color studies.
- What is the CIE color standard and how does it differ from results of those early color experiments?
- What are the three axes of the Munsell color space, which was designed to be a perceptually uniform color space?
- Our perception of color is affected not only by the frequency of light, but by the context in which that color is viewed. Give some examples of how context can affect perception of color.
- Why do we use red, green, and blue in our displays but cyan, magenta, and yellow in our printers? Use principles of additive and subtractive color to give your answer.

- The form of the Rendering Equation from the 1986 SIGGRAPH paper by Jim Kajiya is given below. Explain each term in this equation. Use a sketch to illustrate

$$I(x,x') = g(x,x')\left[\epsilon(x,x') + \int_S \rho(x,x',x'')\, I(x',x'')dx''\right]$$

- How are light sources represented in this equation?
- Where are the reflective properties of a material (e.g., its BRDF) captured?
- Where is falloff of light with distance captured?
- How are transparent and translucent objects handled?

Ray Tracing

- What was the rendering equation? Motivate & list the terms.
- Relate the rendering equation to forward ray tracing. Why is forward ray tracing not good for image formation?
- What is the difference between ray-"casting" and ray-"tracing". When would you only need raycasting?
- Given incoming ray (s,d) hit surface point p (with normal 'n') describe secondary rays you would emit to compute:
    - direct lighting from light at point 'l'.
    - simple (mirror-like) reflection.
    - glossy reflection.
    - refraction.
    - glossy refraction.
    - full indirect illumination.
- How would you implement scattering media in a raytracer?
- Review: Describe how to perform ray intersection tests against:
    - Plane.
    - Sphere.
    - Triangle.
    - Mesh.
    - BONUS: CSG object (assume you have lists of ray intersections with the object primitives that are used to form the final CSG object already).
- Given a camera at (0,0,0) facing (1,0,0) with up-vector (0,0,1), horizontal field-of-view 60 degrees, square pixels, and a 640x480 image, what ray passes through pixel (10,40)? (NOTE: assume the top-left pixel is (0,0).)
- Write pseudo-code for a backward ray-tracer that uses a recursive function.
- Write the contribution of light intensity at a point from various sources, including direct lighting (ambient, diffuse, and specular), a single reflection ray, and a single transmission ray.
- Draw a simple scene and sketch all rays that will be traced from the eye through one pixel in the following cases:
    - Ray casting
    - Ray tracing with one bounce

    - Ray tracing with two bounces

- Consider a partially transparent object that is illuminated with two lights, one visible from each side of the object. Start with a ray from the eye that hits one side of this object at approximately a 45 degree angle from the surface normal and sketch all of the rays that will be traced to determine the color of that intersection point.
- Give Snell's Law and use it to derive an expression for the refraction ray in terms of the surface normal, the incoming direction, and the indices of refraction of the two media.
- Under certain circumstances, there will not be a solution to the equation used to compute the refraction ray. What are these circumstances? What happens to the light ray? This phenomenon is called total internal reflection.
- In ray tracing we choose to trace very few rays through the scene in order to keep computation times reasonable. List and sketch some of the paths that light can travel that are not captured by ray tracing.
- Consider the following effects: mirror reflection, refraction, caustics, and color bleeding. Which of these can be captured through ray tracing? For which does ray tracing not perform well?
- We can define a view frustum with the following inputs:
    - **e** = eye point
    - **v** = up vector
    - **-w** = look vector
    - l = distance of left side of image from the look vector
    - r = distance of right side of image from the look vector
    - t = distance of top of image from look vector
    - b = distance of bottom of image from the look vector
    - n = distance of near clipping plane along look vector
    - f = distance of far clipping plane along look vector
    - nx = number of pixels in a row of the image
    - ny = number of pixels in a column of the image

  Here we assume that the image plane is the near clipping plane. Sketch a diagram showing all of these input variables. Give an expression for the ray cast from the eye point through pixel (i, j) based on these input parameters.
- Show how to compute ray triangle intersections, including identification of barycentric coordinates for use in interpolation of normals and texture coordinates.

- Distributed ray tracing can be used to capture a variety of effects and create less sharp and more realistic images. Explain how to use distributed ray tracing to capture each of the following effects:
    - antialiasing
    - gloss
    - translucency
    - soft shadows
    - depth of field
    - motion blur
- BONUS: If we use distributed ray tracing for all of these effects at once, how can we keep the number of rays that are traced in a scene to some reasonable number and still get good even sampling of each of these effects?

Spatial Data Structures

- Why do we want to build a hierarchical data structure such as a bounding box hierarchy?
- Describe the main difference between the following two approaches: (1) construct a bounding volume hierarchy (e.g. bounding boxes or bounding spheres), (2) construct a hierarchy of splitting planes (e.g. KD trees or BSP trees).
- Describe a technique for constructing a bounding box hierarchy. What data structure will you use to store this hierarchy?
- Is your technique top-down or bottom up? Give an algorithm for the alternative approach (i.e., if your first approach was top-down, give a bottom-up approach to constructing a bounding box hierarchy).
- How do we do intersection testing between a ray and a bounding box? Note that we do not need to find the point of intersection, but we only need to determine whether or not the ray intersects the bounding box.
- How do we do intersection testing between a ray and a bounding sphere? Again, we only need to know whether the ray intersects the sphere. We do not need to find the specific intersection point.
- Explain in detail how to use a bounding box hierarchy to identify the intersection between a ray and the closest object in the environment. Your algorithm should, of course, be more efficient (in the general case) than the brute force process of checking the ray for intersection with all objects.
- How would you update the bounding box hierarchy if there are moving objects in the environment? Is there a way to do this efficiently, i.e., without rebuilding the entire tree?
- An alternative to the bounding box or bounding sphere hierarchy is to use splitting planes to divide space. Octrees, KD trees, and BSP trees are all splitting plane algorithms. Describe the differences between these approaches.
- Describe how to construct a KD tree. Describe how to perform ray-object intersection using this data structure. Your ray-object intersection algorithm should check regions from front to back order, so that the search for an intersection may be halted when an intersection point is found.
- In the KD tree scenario, what problem is introduced when the splitting planes intersect some of the objects? Assume that you do not want to use the plane to split the objects (e.g., splitting a sphere into two parts may not be desirable for ray tracing). How can you solve or work around this problem?
- Describe how to construct a BSP tree and how to do ray-object intersection testing in front to back order with this data structure.