Midterm Review

Computer Graphics 15-362 / 15-662

Lecture 2: Linear Algebra and Vector Calculus:

- How can we measure vectors?
- What is a Vector Space?
- Draw a geometric representation of each rule that vectors seem to obey.
- Can a function be a vector? Explain
- Add and scale vectors.
- Add and scale functions
- What is the norm of a vector?
- Associate each property of a norm with a geometric interpretation.
- Compute the Euclidean norm in Cartesian coordinates.
- Compute the L2 norm of a function.
- Associate each property of the inner product with a geometric interpretation.
- Compute the inner product in Cartesian coordinates.

Lecture 2: Linear Algebra and Vector Calculus:

- Use the inner product for operations such as projection.
- Compute the inner product of functions.
- Give properties and an example of a linear map.
- Define span and basis..
- Compute an orthonormal basis from a set of vectors.
- Be able to use Gram-Schmidt orthonormalization.
- Know that orthonormalization of functions can be done by decomposing them into sinusoids.
- Be able to solve a simple system of linear equations, depict it geometrically, and represent it in matrix form.
- Be able to represent a linear map in matrix form.

Lecture 2: Linear Algebra and Vector Calculus:

- Euclidean norm is a notion of length preserved by rotations/translations/ reflections of space. Be able to calculate it for a vector of any dimension.
- Compute the inner product of two n-dimensional vectors. What is the geometric meaning if an orthonormal basis is used?
- Compute the cross product of two three-dimensional vectors. What is the geometric meaning of this cross product? What is the geometric meaning of its magnitude?
- Use the cross product to do a quarter rotation of a vector within a plane.
- Represent the dot product using matrix notation.
- Represent the cross product using matrix notation.
- Understand what the determinant measures. What does the determinant of a linear map tell us? Give an example.
- What is a directional derivative?
- Bonus: Compute the gradient of a function.

Lecture 2: Linear Algebra and Vector Calculus:

- Understand gradient as the best linear approximation and the direction of steepest ascent.
- When is the gradient not defined?
- Bonus: Be able to express gradients of simple matrix expressions
- What is a vector field? Give an example.
- Be able to compute divergence, curl, and the Laplacian of a vector field. Also be able to express the meaning of these terms geometrically, for example by drawing a diagram.
- What is the Hessian? Be able to compute the Hessian of a function.

Lecture 6: Transforms

1. Which of the following operations are linear transforms: scale, rotation, shear, translation, reflection, rotation about a point that is not the origin?
2. Express scale as a linear transform
3. Express rotation as a linear transform
4. Express shear as a linear transform
5. Express reflection as a linear transform
6. Express translation as an affine transform
7. Know what makes a transform linear vs. affine
8. Know how to build transformation matrices from start and end configurations of your object

Lecture 6: Transforms

- Create 2D and 3D transformation matrices to perform specific scale, shear, rotation, reflection, and translation operations
- Compose transformations to achieve compound effects
- Rotate an object about a fixed point
- Rotate an object about a given axis
- Create an orthonormal basis given a single vector
- Understand the equivalence of [x y 1] and [wx wy w] vectors
- Explain/illustrate how translations in 2D (x, y) are a shear operation in the homogeneous coordinate space (x, y, w)

Lectures 7&8: Projection and Rasterization

- Review:
  - Form an orthonormal basis
  - Create a rotation matrix to rotate any coordinate frame to xyz
  - Create the rotation matrix to rotate the xyz coordinate frame to any other frame
  - Know basic facts about rotation matrices / how to recognize a rotation matrix
    - Rows (also columns) are unit vectors
    - Rows (also columns) are orthogonal to one another
    - If our rows (or columns) are u, v, and w, then uXv=w
    - The inverse of a rotation matrix is its transpose
- For any given setup where we place a camera in the environment, pointing down any of the main coordinate axes (x, y, or z), compute a projection of points in the world onto an image plane.
- Create a projection matrix that projects all points onto an image plane at z=1
- Propose a projection matrix that maintains some depth information

Lectures 7&8: Projection and Rasterization

- Understand the motivation behind the projection matrix that projects the view frustum to a unit cube
- Be able to draw / discuss the details of the view frustum
- Prove that a standard projection matrix preserves some information about depth
  Write an algorithm for drawing lines that handles all edge cases (i.e., including edges that are exactly horizontal or vertical).

1. How do we determine coverage for a triangle?
2. Give 2 algorithms for determining triangle coverage
3. Bonus: Be able to use the implicit edge representation to determine if a point is inside a triangle.

Lectures 7&8: Projection and Rasterization

- Textures are used for many things, beyond pasting images onto object surfaces. Be able to list and describe the use cases in this list:
  - Normal maps (create appearance of bumpy object on smooth surface by giving false normal to the lighting equations)
  - Displacement maps (encode offsets in the geometry of a surface, which is difficult to handle in a standard graphics pipeline)
  - (Bonus) Environment maps (store light information in all directions in a scene)
  - (Bonus) Ambient occlusion map (store exposure of geometry to ambient light for better representation of surface appearance with simple lighting models)
  - Can you think of / discover others?
- Know how to interpolate texture coordinates
- Know how to index into a texture and compute a correct color using bilinear interpolation
- Be able to create a mipmap and store it in memory
- Be able to compute color from multiple levels of mipmaps using trilinear interpolation
- What is the logic behind selecting an appropriate level in a mipmap?
- What can happen if we select a level that is too high resolution? too low resolution?

Lectures 7&8: Projection and Rasterization

1. How should we choose the correct color for a pixel? There is not an exact right answer. However, you should be able to discuss some of the issues involved.
2. What is aliasing, and what artifacts does it produce in our images and our animations? Give at least 3 different examples.
3. One form of aliasing is where high frequencies masquerade as low frequencies. Give an example of this phenomenon.
4. Suppose we have a single red triangle displayed against a blue background. Does this scene contain high frequencies?
5. (Bonus) What does the Nyqvist-Shannon theorem tell us about how image frequencies relate to required sampling rate?
6. (Bonus) One practical solution on your graphics card for reducing aliasing (i.e., for antialiasing) is to take multiple samples per pixel and average to get pixel color. Try to use sampling theory to explain as precisely as you can why taking multiple samples per pixel can reduce aliasing artifacts

Lecture 9: Depth and Transparency

- Interpolate colors using barycentric coordinates
- Bonus: Compute barycentric coordinates of a point using implicit edge functions
- Compute barycentric coordinates of a point using triangle areas
- Estimate the location of a point inside a triangle given its barycentric coordinates
- Estimate the location of a point outside a triangle given its barycentric coordinates

- Estimate barycentric coordinates of a point from a drawing.
- Show that interpolation in 3D space followed by projection can give a different result from projection followed by interpolation in screen space. In other words, explain why interpolation using barycentric coordinates in screen space may give a result that is incorrect.
- How, then, can we obtain a correct result using interpolation in screen space?

Lecture 9: Depth and Transparency

- What is the depth buffer (Z-buffer) and how is it used for hidden surface removal?
- Where does the depth for each sample / fragment come from? Where is it computed in the graphics pipeline?
- Is the depth represented in the depth buffer the actual distance from the camera? If not, what is it?
- What is the meaning of the alpha parameter in the [R G B a] color representation?
- Be able to use alpha to do compositing with the "Over" operator.
- Is "Over" commutative? If not, create a counterexample.

$$C = \alpha_B B + (1 - \alpha_B)\alpha_A A$$

- What is premultiplied alpha, and how does it work?
- Be able to use premultiplied alpha for "Over" composition.
- Why is premultiplied alpha better? Give 2 examples.
- How do we properly render a scene with mixed opaque and semi-transparent triangles? What is the rendering order we should use? When is the depth buffer updated?
- Draw a rough sketch of the graphics pipeline. Think about transforming triangles into camera space, doing perspective projection, clipping, transforming to screen coordinates, computing colors for samples, computing colors for pixels, the depth test, updating color and depth buffers.

Lecture 9: 3D Rotations

- What is the problem of gimbal lock? Give an example where this problem occurs.
- How does using quaternions solve this problem?
- Know that every rotation can be expressed as rotation by some angle about some axis.
- Know how to go between quaternions and axis-angle format for rotations.
- Know that quaternions are expressed as higher dimensional complex numbers.
- Be able to work out quaternion multiplication from the complex number representation of a quaternion.

Lecture 10: Introduction to Geometry

- List some types of implicit surface representations
- What types of operations are easy with implicit surface representations?
- List some types of explicit surface representations
- What types of operations are easy with explicit surface representations?
- What is a manifold surface?
- Distinguish manifold from non-manifold surfaces
- Can a manifold surface have a boundary? Give an example.
- Describe how to store mesh information in vertex and face tables. Give an example. What are good and bad points of this data structure?
- How would you store a mesh using incidence matrices? What are good and bad points of this data structure?
- What do you need to store in a halfedge data structure? What are good and bad points of this data structure?
- How can you find all vertices in a face with the halfedge data structure?
- How can you find all faces that contain a vertex with the halfedge data structure?
- BONUS: Think of an algorithm to traverse every face in a manifold using this data structure.

Lecture 11: Geometry Processing

- List practical applications that you can relate to for good geometry processing algorithms.
- Give criteria for what makes a good quality mesh. Be sure to state your assumptions (e.g., good quality for what purpose?). Consider triangle size and shape and vertex out-degree.
- Give pseudocode for one iteration of Catmull-Clark subdivision
- Give pseudocode for one iteration of Loop subdivision
- Understand how subdivision algorithms are characterized in terms of their properties (interpolation, continuity, behavior at the boundaries)
- Be prepared to calculate vertex updates in a simple example of Loop or other subdivision. (The vertex weighting masks will be given to you.)
- Understand how the K matrix of the Quadric Error error metric encodes squared distance to a plane. How can it encode the sum of squared distances to many planes? How is this idea used in generating a good error metric for mesh decimation using edge collapse?

Lecture 11: Geometry Processing

- Express distance from a plane, given a point on the plane and a normal vector
- Show how the K matrix (the quadric error matrix) represents squared distance from a plane
- Given K matrices encoding triangles in a mesh, how do we get the K matrix for each vertex?
- If we collapse an edge, what is the K matrix for the new vertex that is added in the edge collapse?
- Given a K matrix and a proposed point, what is the cost (the quadric error)?
- How does this cost represent distance to the original surface?

Lecture 13: Geometric Queries

- Describe some techniques for improving the quality of a mesh to make it more uniform and regular.
- Find the closest point to a point, line, or line segment.
- Give an explicit representation for a ray.
- Give an implicit representation for a plane.
- Compute ray-triangle intersection, including checking whether the ray passed through the inside of the triangle.
- Be prepared to compute ray-primitive intersection for other primitives (e.g., a sphere) and primitive-primitive intersections (e.g., triangle-triangle or line-line)
- In all (most?) of these queries, what is the basic strategy that you use to find the closest point or perform the intersection? Think about how you use explicit and implicit representations of the geometries.

Lecture 13: Geometric Queries

- Be able to distinguish between object-centric (primitive partitioning) acceleration structures and space-centric (space-partitioning) acceleration structures
- Know the difference between these acceleration structures and be able to sketch an example of each type, including the enclosing geometry and the tree data structure that represents it:
  - bounding box and bounding sphere hierarchies
  - KD-trees
  - quadtrees and octrees