table of contents

Contents

1 Chapter 1 Introduction

2 Chapter 2 Misc Math

1. (trig identities)
2. **(dot product)**
$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\phi \qquad proj_{\mathbf{a}}(\mathbf{b}) = \|\mathbf{a}\| \cos(\phi) = \frac{a \cdot b}{\|\mathbf{b}\|}$$
3. **(cross product)**
$$\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \sin(\phi)\mathbf{n} = \det \begin{pmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{pmatrix}$$
where $\mathbf{n}$ is unit vector perpendicular to plane formed by $\mathbf{a}, \mathbf{b}$, determined by the right hand rule
4. **(coordinate frames)**
   - the **global coordinate system** is one formed by cartesian canonical orthonormal basis and the canonical origin that is not stored explicitly.
   - Other coordinate system is called a **frame of reference** or **coordinate frame** are stored explicitly (the origin, and 3 orthonormal vectors as a function of the canonical basis) and is called a **local coordinate system**
   - If $\mathbf{b}$ is in canonical $\mathbf{x} - \mathbf{y} - \mathbf{z}$ coordinate and want to transform to a local $\mathbf{u} - \mathbf{v} - \mathbf{w}$ coordinate. then the transformed vecter can be written as
$$\sum_{\mathbf{q} \in \{\mathbf{u}, \mathbf{v}, \mathbf{w}\}} \langle \mathbf{q}, \mathbf{b} \rangle \, \mathbf{q}$$
5. **(construct basis)** by GramSchmidt process
6. (linear interpolation) over 2 points or a set of points where the function is piece-wise linear
7. (triangles)
   - barycentric coordinates is a nonorthogonal makes interpolation straight-forward over a triangle. Say $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ forms a triangle, then the vertices $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ act as basis vectors and form the barycentric coordinates. Any point $\mathbf{p}$ is
$$\mathbf{p}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c} \qquad \alpha + \beta + \gamma = 1$$
   (α, β, γ) is the barycentric coordinate of $\mathbf{p}$ w.r.t. $\mathbf{a}, \mathbf{b}, \mathbf{c}$. Given $\mathbf{p}$ in cartesian coordinate, we can find its baarycentric coordinate by solving
$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}) \quad \iff \quad \begin{pmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{pmatrix} \begin{pmatrix} \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} x_p - x_a \\ y_p - y_a \end{pmatrix}$$
   Geometrically, α, β, γ can be computed using triangle area
$$\alpha = A_\alpha / A \quad \beta = A_\beta / A \quad \gamma = A_\gamma / A$$
   Properties
   - can detect pointson edge/verties easily
   - mixes coordinates of vertices smoothly
   - can detect insidedness of points easily, i.e. $\alpha, \beta, \gamma > 0$

3 Chapter 3 Raster Images

1. **(raster display)** show images as rectangular arrays of pixels.
2. **(raster images)** a 2D array that stores pixel value for each pixel that is device-independent
3. **(vector images)** description of shapes that is resolution independent but needs to be rasterized before display
4. **(point sample)** the reflectance, or fraction of light reflected as a function of position on a piece of paper

$$I(x, y) : R \subset \mathbb{R}^2 \to V$$

where V is the set of possible pixel values, i.e. $V = \mathbb{R}^+$ for grayscale images. Pixel in raster images are local average of the color of the image, called a point sample of the image

5. **(pixel values)** as tradeoff between memory and resolution. Lowered bits introduce artifacts such as clipping and banding
6. **(RGB color)**
   (a) (pixel coverage) α is the fraction of pixerl covered by the foreground layer.
   (b) (composite pixel) Let $\mathbf{c}_f, \mathbf{c}_b$ be foreground and background colors respectively, then $\mathbf{c} = \alpha\mathbf{c}_f + (1 - \alpha)\mathbf{c}_b$
   (c) **(alpha/transparency mask)** α values for all pixels, stored as a separate grayscale image; or stored as a 4th channel, called the alpha channel

4 Chapter 4 Ray Tracing

1. **(rendering)** take a scene, a model, composed of many geometric objects arranged in 3D space and produce a 2D image that shows the objects as viewed from a particular viewpoint
   - **(object-order rendering)** for each object, update pixels that the object influences
   - **(image-order rendering)** for each pixel, set pixels based on the objects that influence it
2. **(ray-tracing)** image-order algorithm for making renderings of 3D scenes
   (a) **(ray generation)** compute origin and direction of each pixel's viewing ray based on the camera geometry
   (b) **(ray intersection)** finds closest object intersecting the viewing ray
   (c) **(shading)** computes the pixel color based on the reseults of ray intersection
3. **(linear perspective)** 3D objects projected to image plane in such a way that stright lines in the scene become straight lines in the image
4. **(parallel projection)** 3D points are mapped to 2D by moving them along a projection direction until they hit the image plane
   - **(orthographic/oblique projection)** if image plane is perpendicular to view direction, the projection is orthogrpahic; otherwise, it is called oblique
5. **(camera frame)** let $\mathbf{e}$ be the viewpoint, and $\mathbf{u}, \mathbf{v}, \mathbf{w}$ be the three basis vectors. $\mathbf{u}$ points upward (from camera's view), $\mathbf{v}$ points upward, and $\mathbf{w}$ points backward. $-\mathbf{w}$ is called the view direction.
6. **(computing viewing ray)** represent ray with 3D parametric line

$$\mathbf{p}(t) = \mathbf{e} + t\mathbf{d}$$

idea is to find $\mathbf{e}, \mathbf{d}$
   (a) **(orthographic view)** where $\mathbf{e}$ the viewpoint is placed on the image plane

- compute coordinate (u, v) of each pixel (i, j) on the image plane of size $(r - l) \times (t - b)$

$$u = l + (i + 0.5)\frac{r - l}{n_x}$$
$$v = b + (j + 0.5)\frac{t - b}{n_y}$$

- set ray's origin to be $\mathbf{e} + u\mathbf{u} + v\mathbf{v}$ and direction to be $-\mathbf{w}$

(b) **(perspective views)** project along lines that pass through a single point, the viewpoint, rather than along parallel lines. $\mathbf{e}$ the viewpoint is placed some distance d in front of $\mathbf{e}$, call this distance the **image plane distance / focal length**

- compute coordinate (u, v) of each pixel (i, j) on the image plane using previous formula
- set ray's origin to be $\mathbf{e}$ and direction to be $-d\mathbf{w} + u\mathbf{u} + v\mathbf{v}$

from eye $\mathbf{e}$ to a point $\mathbf{s}$ on the image plane.

7. **(ray-sphere intersection)** Solve a quadratic formula satisfying the 2 condition, that the intersecting point is both on the ray and on the sphere.

8. **(ray-triangle intersection)** Want to find the first intersection between the ray $\mathbf{e} + t\mathbf{d}$ and a surface that occurs at a $t \in [t_0, t_1]$. Given parametric surfaces , we can solve for

$$\mathbf{e} + t\mathbf{d} = \mathbf{f}(u, v)$$

for 3 unknowns, u, u, v. If the surface is a plane in barycentric coordinate, then solve for

$$\mathbf{e} + t\mathbf{d} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

for some t, β, γ. The intersection is inside the triangle if and only if $\beta > 0$,$\gamma > 0$, and $\beta + \gamma < 1$. There is no solution if either the triangle is degenerate or the ray is parallel to the plane containing the triangle. This equation can be solved analytically with Cramer's rule

9. **(ray-polygon intersection)** Given a planar polygon with vertices $\{\mathbf{p}_1, \cdots, \mathbf{p}_m\}$ and surface normal $\mathbf{n}$, we can compute intersection point between ray and plane containing polygon $\mathbf{p}$ with

$$(\mathbf{p} - \mathbf{p}_1) \cdot n = 0 \qquad t = \frac{(\mathbf{p}_1 - \mathbf{e}) \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}$$

then we check if $\mathbf{p}$ is inside the polygon by sending 2D ray out from $\mathbf{p}$ and count the number of intersections between the ray and boundary of the polygon: if the number of intersections is odd, then the point is inside the polygon.

10. **(ray-scene-intersection)** To intersect a ray with a group of objects, i.e. the scene, simply intersect ray with the objects in the group and return the intersection with the smallest t value.

11. **(shading model)** is designed to capture the process of light reflection, whereby surfaces are illuminated by light sources and reflect part of the light to the camera

- (light direction) $\mathbf{l}$ is unit vector pointing toward light source
- (view direction) $\mathbf{v}$ is the unit vector pointing toward camera
- (surface normal) $\mathbf{n}$ is unit surface normal at point of reflection
- (properties of the surface), i.e. color, shininess

12. **(Lambertian Shading)** For each color channel, compute pixel color L with

$$L = k_d I \max(0, \mathbf{n} \cdot \mathbf{l})$$

where k_d is the diffuse coefficient or the surface color, and I is intensity of light source, and $n \cdot l = \cos(\theta)$ where θ is angle between surface normal and light source. Lambertian shading is **view independent**, i.e. color of the surface does not depend on the viewing direction $\mathbf{v}$, leading to matte, chalky appearance. Higher k_d, more contrasty

13. **(Blinn-Phong Shading)** To account for shininess, or **specular reflection**, a **specular component** is added to the **diffuse component** to account for highlights. Blinn-Phong accounts for specular reflection by generating brightest reflection when $\mathbf{v}$ and $\mathbf{l}$ are symemtrically positioned across the surface normal, i.e. mirror reflection; the reflection decreases smoothly as the vectors move away from the mirror configuration. Idea is to compare bisector of $\mathbf{v}$ and $\mathbf{l}$ to the surface normal $\mathbf{n}$

$$L = k_d I \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s I \max(0, \mathbf{n} \cdot \mathbf{h})^p \quad \mathbf{h} = \frac{\mathbf{v} + \mathbf{l}}{\|\mathbf{v} + \mathbf{l}\|}$$

where k_s is the specular coefficient of the surface and p is Phong exponent where larger value indicate shinier/glossier surface

14. **(Ambient Shading)** To avoid rendering completely black pixels for surfaces that receive no illumination, add a constant illumination to surfaces, with no dependence on surface geometry

$$L = k_a I_a + k_d I \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s I \max(0, \mathbf{n} \cdot \mathbf{h})^n$$

where k_a is surface specific ambient coefficient and I_a is the ambient light intensity.