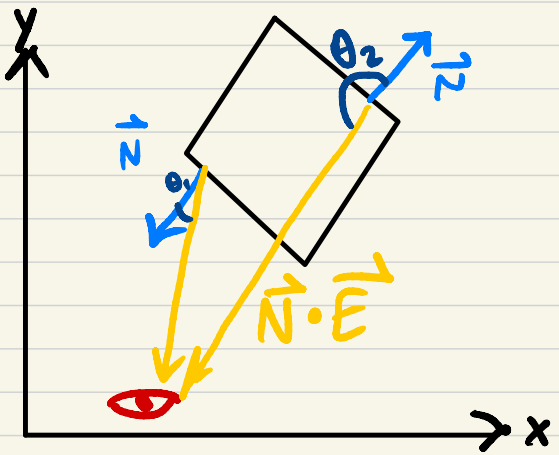
Lecture 10

Backface Culling

- If $\theta > 90$, face is facing away from eye
  - In World Space

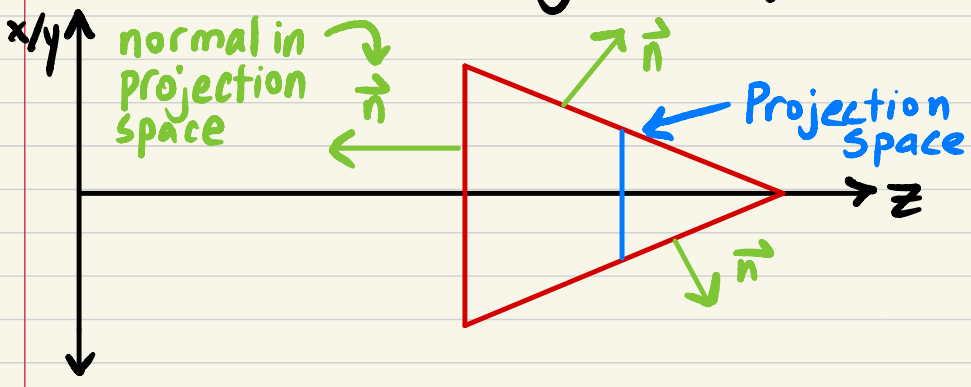
WS



$0 \leq \theta_1 \leq 90$
- Front Face

$\theta_2 > 90$
- Backface

Backface Culling in Projection Space



$+z$ component normal
- backface

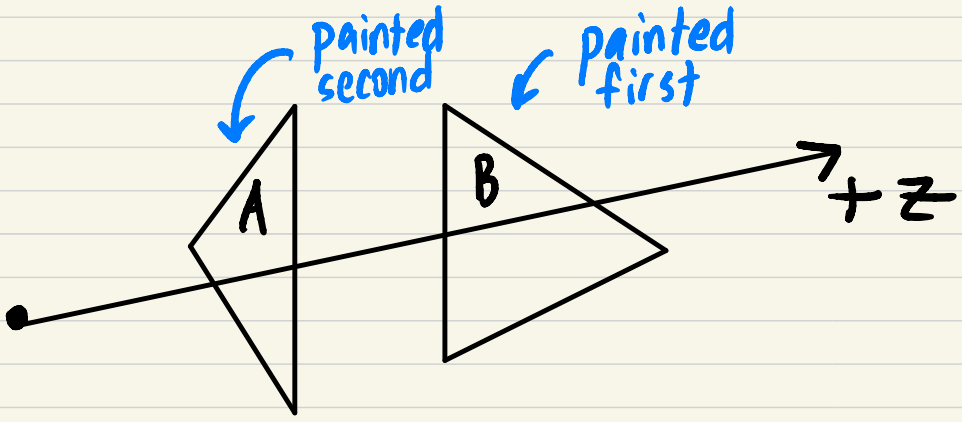
$-z$ component normal
- front face

Painter's Algorithm

- Occurs in Object Space

1. Sort all polygons by z-depth
2. Scan-convert polygons in back-to-front order

- Edge Cases (Painter's Algorithm cannot handle)
  - ↳ Cyclic polygons
  - ↳ Intersecting polygons



Z-buffer Algorithm

- Occurs in image/screen space

2 buffers

↳ z-buffer : Stores depth

↳ color-buffer : Stores background (of each polygon)

Algo:

- For each pixel covered by polygon

↳ Calculate z for polygon at (x, y)

↳ If (z < z-buffer[x][y]):

↳ z-buffer[x][y] = z

↳ color-buffer[x][y] = color of polygon

- Objects do not need to be sorted
- Handles penetrating and cyclic objects
- Aliasing Issues (No weighted color average)

$$z = \frac{Ax + By + D}{C}$$

Scanline z-buffer Algorithm

span: contiguous set of pixels on same scanline

2 buffers: z-buffer, color buffer

Algo:

- For each scanline (y):
  - For each polygon on scanline:
    - Scan-convert/calculate span
    - For each pixel in span:
      - if (z < z-buffer[x])
        - z-buffer[x] = z
        - color-buffer[x] = color
- Multiple Passes through polygon database

Efficiency in Scanline Z-buffer

- Incremental Calculation / Bilinear Interpolation

$Ax + By + Cz + D = 0 \rightarrow z = -\frac{Ax+By+D}{C}$

$z_{x+1} - z_x = -\frac{A}{C}$ ← Can only be applied once per polygon

Lecture 11

Global Illumination

- Light scatters in all directions
  - ↳ Small percentage of light heads in direction of the camera

Types of Lighting

- Ambient : Make out outline of object
- Diffuse : Shape of object via light depth
- Specular : Shiny metallic reflection

Phong = Ambient + Diffuse + Specular
↳ Shiny Plastic

- Reflectivity : Can you see other objects in it
- Shininess : How smooth is it

Ambient Lighting

↳ Does not depend on position of light, object, or eye

Ambient Light reflected off object

$= k_a \cdot I_a$

k_a: ambient reflection coefficient

I_a: intensity of ambient light source

Diffuse Lighting

↳ Reflects light equally in all directions

Diffuse Light off object:

$$k_d \cdot I_p \cdot \cos\theta = k_d \cdot I_p \cdot (\vec{N} \cdot \vec{L})$$

k_d: diffuse reflection coefficient

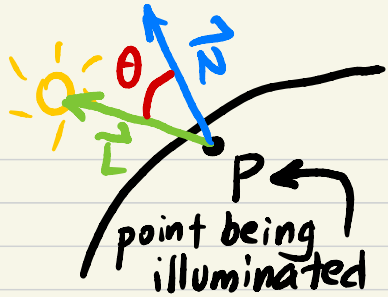
I_p: intensity of point light source

θ: angle between normal and light vector

$\vec{N}$: normal

$\vec{L}$: light vector

Lambert's Law



- Intensity $\propto \cos\theta$
  - $\propto$ is proportional to
  - $\theta = 0 \rightarrow I = 1$ (light aligns with normal)
  - $\theta = 90 \rightarrow I = 0$
  - $\theta > 90 \rightarrow I = \max(0, \cos\theta)$, $\cos\theta = \vec{N} \cdot \vec{L}$

Specular Lighting

- Depends on position of light, object, and eye
- Light reflects unequally in different directions

Specular light reflected off object:

$$f_{att} \cdot k_s \cdot I_p \cdot \cos^n \phi$$

$$= f_{att} \cdot k_s \cdot I_p \cdot (\vec{R} \cdot \vec{V})^n$$

- $\vec{R}$: Reflection Vector
- $\vec{V}$: View / Eye Vector
- n: Material specular reflection component

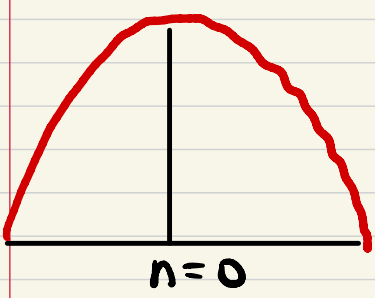
if $n = \infty$ → perfect reflection

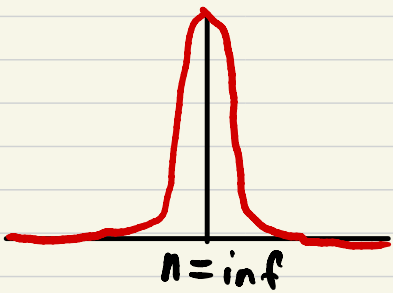
Smoothness Exponent Effect

- In regards to the Specular term
  - If eye aligns with reflection, specular light will be brightest
- $(\cos\varphi)^n$
  - As n increases
    - curve narrows
    - rate of specular decreasing increases

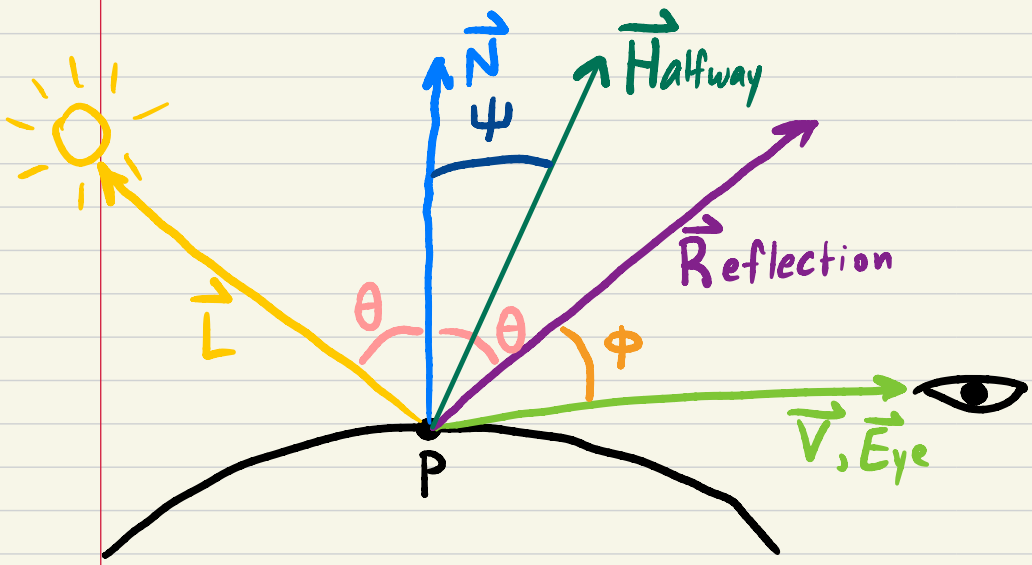
$n = \infty$

- Mirror, change 1°, cannot see reflection anymore





Specular Lighting



$$\cos\Phi = \vec{R} \cdot \vec{V}$$

$$\vec{H} = \text{normal}(\vec{L} + \vec{V}) = \frac{\vec{L} + \vec{V}}{\|\vec{L} + \vec{V}\|}$$

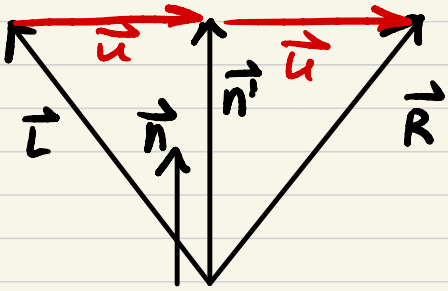
Blinn Phong:

★? Replace $(\vec{R} \cdot \vec{V})^n$ with $(\vec{H} \cdot \vec{N})^n = \cos^n \Psi$

$\Psi = \Phi/2$

Calculating $\vec{R}$ vector

$\vec{n}'$ is projection of $\vec{L}$ onto $\vec{n}$



$$\vec{n}' = (\vec{N} \cdot \vec{L})\vec{N}$$

↳ if $\|\vec{N}\|^2 = 1$

$$\vec{u} = \vec{n}' - \vec{L}$$

$$\vec{R} = \vec{L} + 2(\vec{n}' - \vec{L}) = 2(\vec{n} \cdot \vec{L})\vec{n} - \vec{L}$$

* Lighting Misc Improvements
  ↳ Spot Lights

Lecture 12

Flat Shading

- Also called constant or faceted shading
  - → No interpolation
- Occurs in World Space or Eye Space
- Find illumination at center, apply same color to all points inside polygon

Mach Banding

- Exaggerates contrast between edges of slightly variated shades of same color

Gouraud Smooth Shading

- In Eye Space:
  - → Illuminate each vertice
- In Screen Space:
  - → Interpolate color across polygon face
- Issues
  - → Thin Polygons
  - → Specular Reflection with Large Polygons

Phong Smooth Shading

- Calculate illumination at each pixel
- Interpolate normals similar to others, then normalize

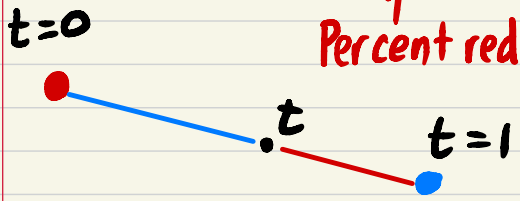
Issues with Smooth Shading

- 3-D polygon smoothness depends on degree of tessalation (not related to shading)
- Orientation dependence

Baycentric Interpolation

Value at t: $(1-t)x_1 + t\,x_2$

Percent red: $(1-t)$ — Percent blue: t



Percent red: $\frac{\text{Area of red triangle}}{\text{Total Area}}$

Percent blue: $\frac{\text{Area of blue triangle}}{\text{Total Area}}$

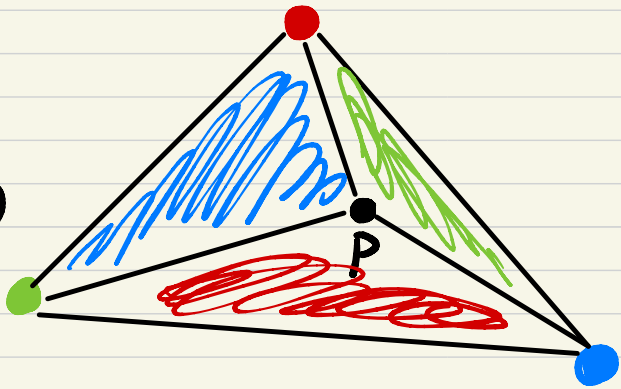
Percent green: $\frac{\text{Area of green triangle}}{\text{Total Area}}$

Value at p

(%.red)(red)

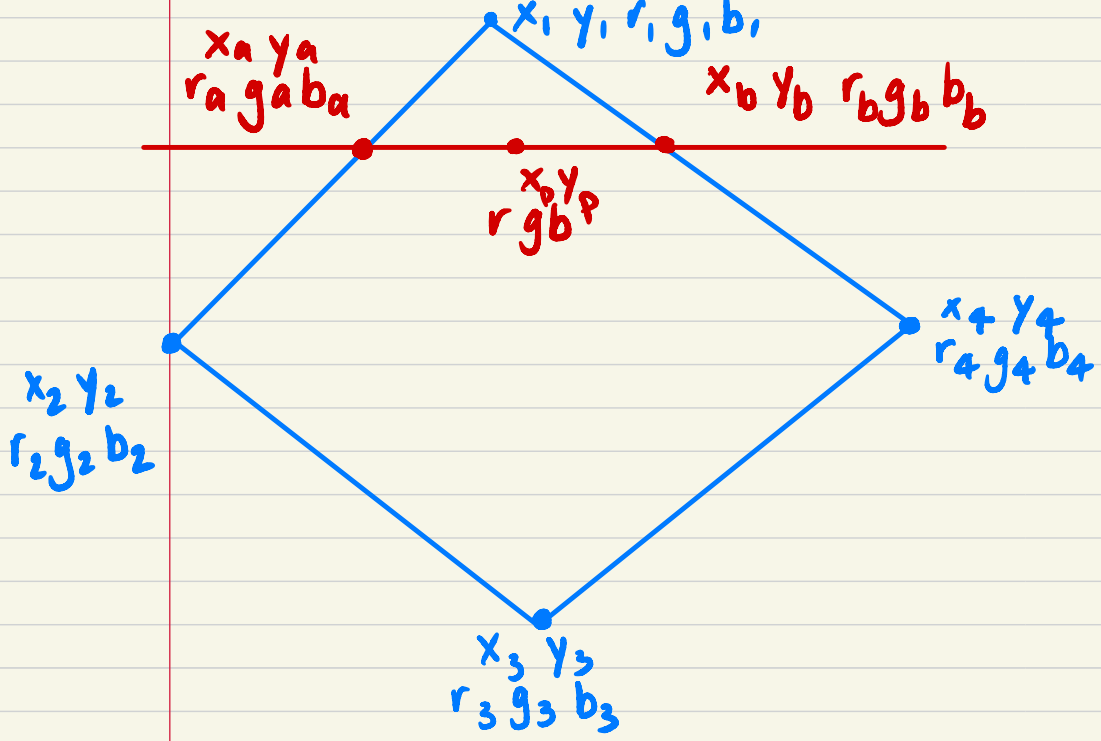
+(%.green)(green)

+(%.blue)(blue)



Bilinear Interpolation

- Interpolate along 2 edges
  - ↳ Interpolate along the scanline
    - ↳ Incremental Calculations



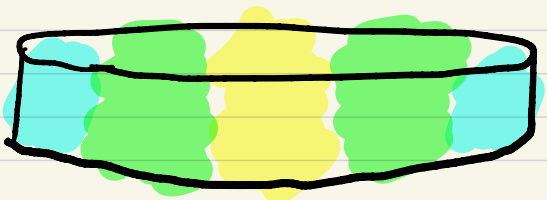
$y_a = y_b = y \longrightarrow \frac{r_a - r_2}{r_1 - r_2} = \frac{y_a - y_2}{y_1 - y_2}$

$$\frac{r - r_a}{r_b - r_a} = \frac{x - x_a}{x_b - x_a}$$

Lecture 13

Non-Photorealistic Shading

- Angle between $\vec{N}$ and $\vec{L}$ variates color
  - To silhouette: $(\vec{N} \cdot \vec{V} < 0.01)$



Radiosity

- Based on light energy conservation
  - Calculate how much light energy must be used to illuminate each polygon
  - Replaces background/ambient light in equation

Texture Mapping

- Map digitized image onto polygon face
- Texture coordinates

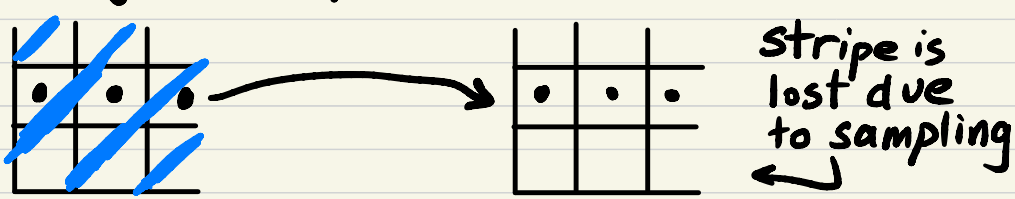
u, v [0,1]
coordinates of polygon

s, t
coordinates of pattern

Texel → Texture element
↳ Use texel color as diffuse color

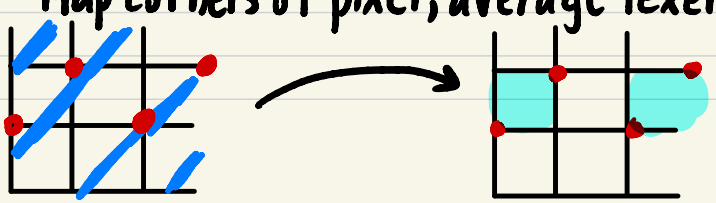
Aliasing in Texture Maps

- In high frequency patterns, some detail is lost



Area Sampling

- Map corners of pixel, average texel colors in pixel color's range

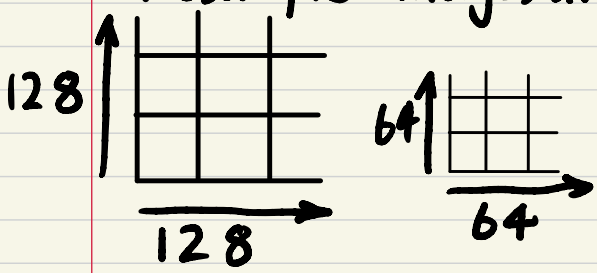


Supersampling

- Illuminate each pixel multiple times independently
  - ↳ Increases rendering time

Mipmapping

- Resample images at smaller sizes to reduce rendering time

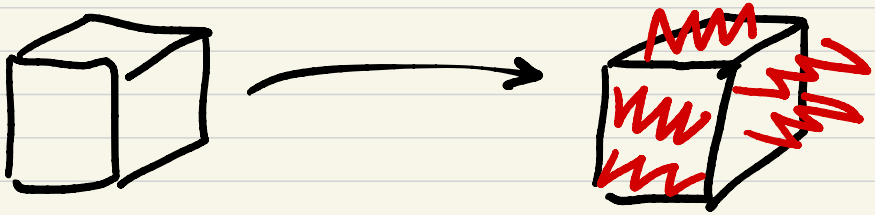


Multi-Texturing

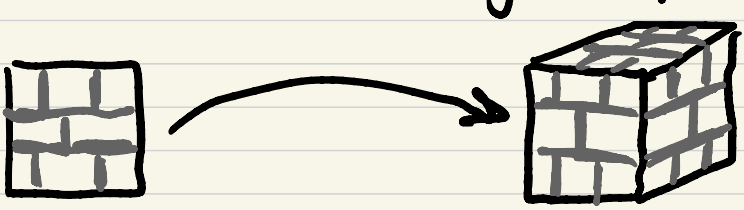
- Apply multiple textures onto object by blending

Creating Bumpy Textures

- Displacement Mapping
  - ↳ Add texture by physically adding more geometry (change shape of object)
  - ↳ Requires hidden surface removal



- Bump Mapping
  - ↳ Change direction of normal and variate in several directions at every point
  - ↳ Bump Maps are grayscale
  - ↳ Does not change shape of object



Bump Map

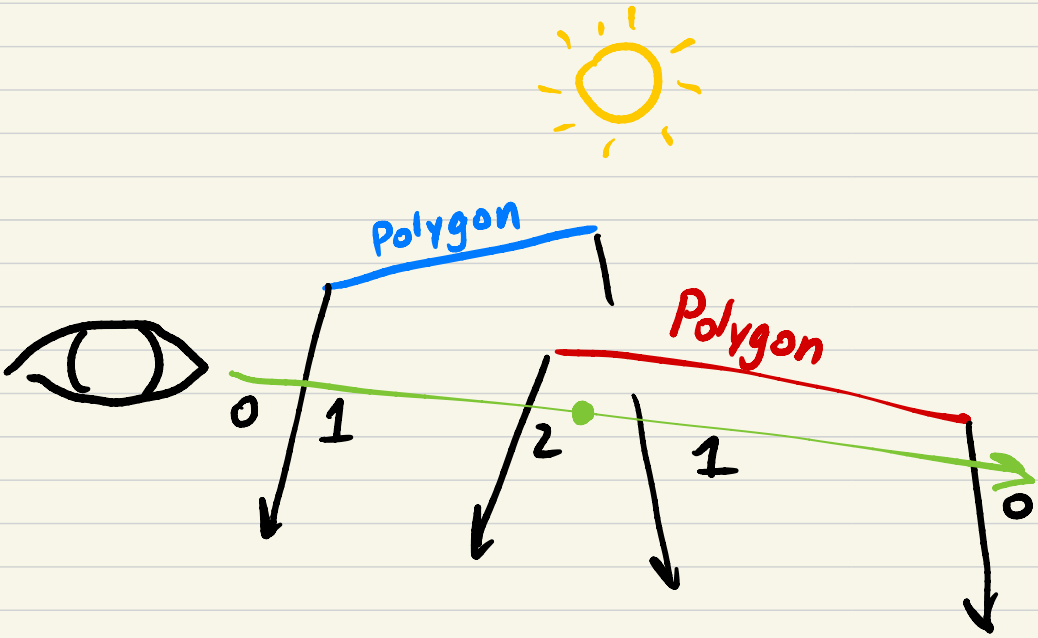
Environmental/Reflection Mapping

- Use reflection with view vector
  - Use polar/spherical coordinates to map color from reflection map to point
    - Reflection map defined on 6 sides of box or sphere

Shadow Algorithms

Shadow Volume Method

1. Create shadow volume for each front facing polygon
2. Store shadow volume
3. Apply parity test to check if visible point is in shadow
   - ↳ If parity > 0, point is in shadow

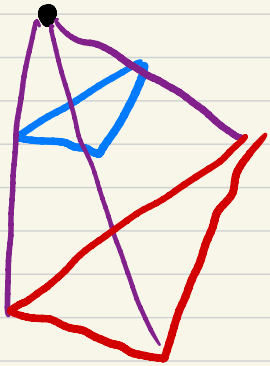


2-Pass Z-Buffer

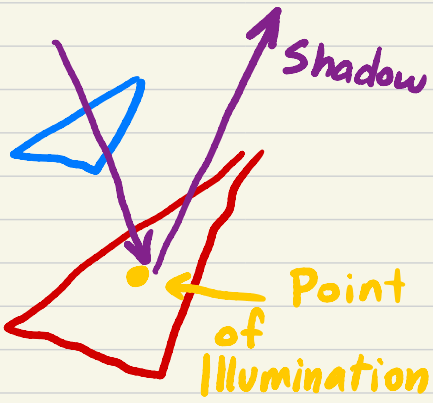
- If point is not visible by light source, it is in shadows

1. Do z-buffer from light position
2. Do z-buffer from eye position

1.



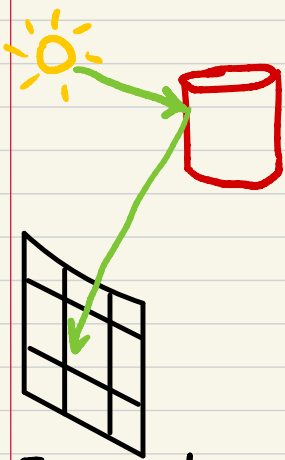
2.



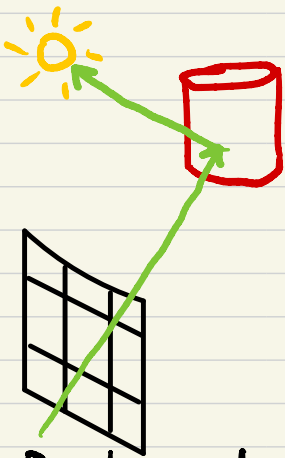
Lecture 14

Ray Casting Algorithm

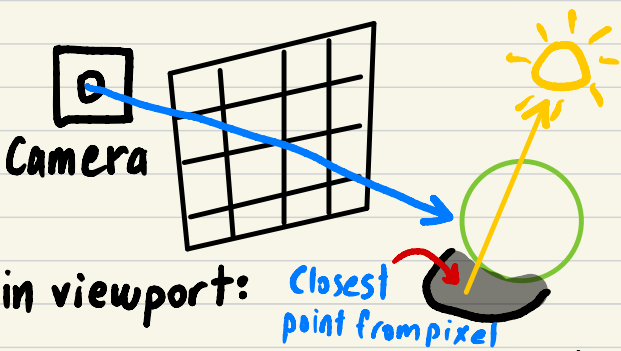
- Occurs in Eye Space or World Space



Forward



Backward

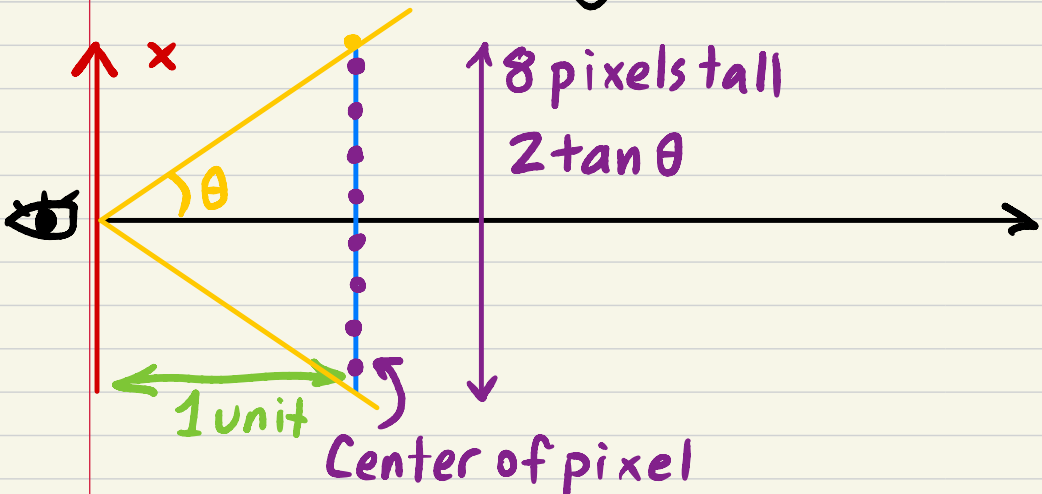


Algorithm:

For each pixel in viewport:

1. Shoot ray from eye through each pixel
2. Find intersection between ray and objects
3. Pick intersection point closest to eye
4. Illuminate closest intersection point
5. Plot pixel with illuminated color of closest object

Ray Tracing in Eye Space



$$\text{width of pixel} = \frac{2\tan\theta}{x_{res}} = \frac{2\tan\theta}{8}$$

x-coordinate of center of pixel (i,j):

$-\tan\theta + (i+0.5)w_p$

← leftmost edge

y-coordinate of center of pixel (i,j):

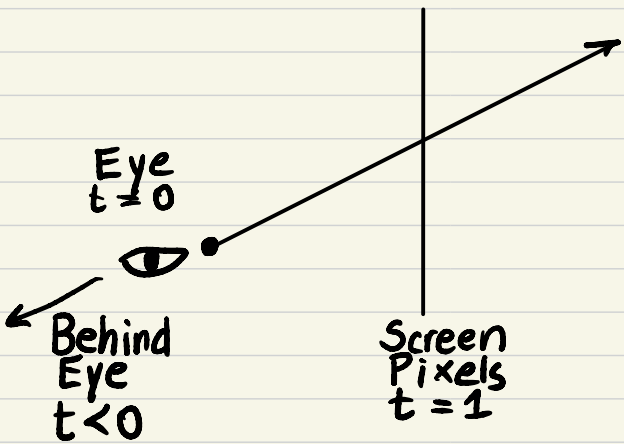
$$\frac{1}{A_r}\left[\tan\theta - (i+0.5)\frac{2\tan\theta}{y_{res}}\right]$$

← topmost edge

Parametric Form of Ray from Eye

- Assume everything in world is a sphere
- Eye Space:

$$x = t x_p$$
$$y = t y_p$$
$$z = t z_p$$



Collision with Ray & Sphere

Sphere: $(x-a)^2 + (y-b)^2 + (z-c)^2 - R^2 = 0$

↳ $(t x_p - a)^2 + (t y_p - b)^2 + (t z_p - c)^2 - R^2 = 0$

↳ $At^2 + Bt + C = 0$

$A = x_p^2 + y_p^2 + 1$

$B = -2(x_p + y_p + 1)$

$C = a^2 + b^2 + c^2 - R^2$

$\vec{N} = P - \text{Center}$

$$t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

$B^2 - 4AC < 0$ ↳ no intersection

$B^2 - 4AC = 0$ ↳ touch edge

$B^2 - 4AC > 0$ ↳ full collision

Collision with Polygon & Sphere

Plane: $Ax + By + Cz + D = 0$

↳ $Ax_p t + By_p t + Ct + D = 0$

$$t = -\frac{D}{Ax_p + By_p + C}$$

↳ Denominator = 0

↳ No intersection

$\vec{N} = \langle A, B, C \rangle$

Check if intersection point P in polygon:

↳ Project P onto 2-D plane

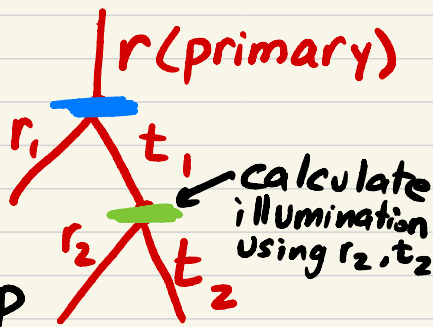
↳ ignore axis of max(A, B, C) coord

↳ Apply 2-D test

Ray Tracing

- Primary Ray : Ray from eye into world
- Secondary Ray : Reflection, Refraction, Shadow

Ray Tree



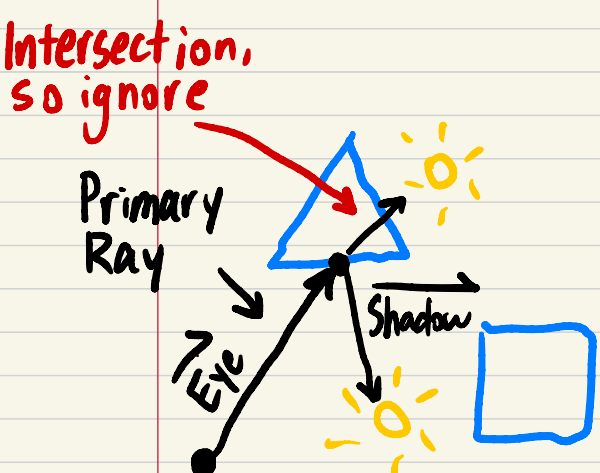
Illumination of Ray Tree

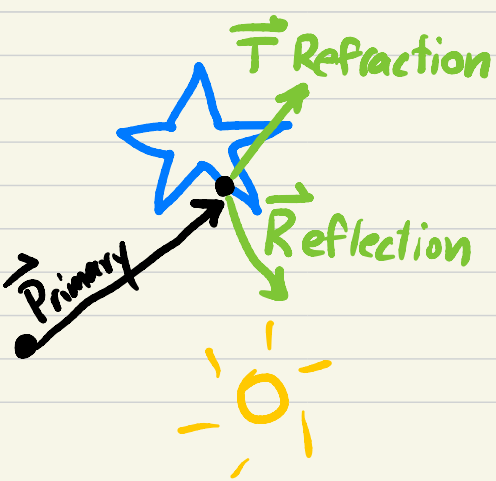
↳ Calculated bottom-up

$I = \text{ambient} + \text{diffuse} + \text{specular} + k_r I_r + k_t I_t$

k_r: reflection coefficient

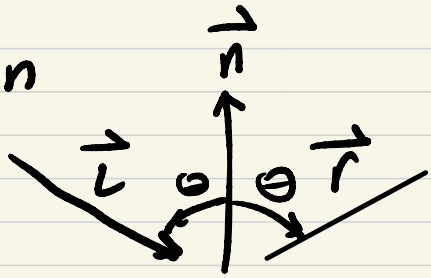
k_t: refraction coefficient





Reflected Ray Direction

$$\vec{r} = \vec{i} - 2(\vec{i} \cdot \vec{n})\vec{n}$$



Acne

↳ Slide along normal to fix

Use Lecture 15 notes

Lecture 16

Transparency / Opacity

rgbα ← Opacity [0,1]

0 Opacity ⟶ Complete Transparent

1 Opacity ⟶ Complete Opaque

Transparency = 1 − Opacity

Composing Opacity:

Straight Alpha: RGB not multiplied by α

$$C_o = \frac{C_f \alpha_f + C_b \alpha_b (1 - \alpha_f)}{\alpha_f + \alpha_b (1 - \alpha_f)}$$

color at point?

Pre-Multiplied: RGB multiplied by α

$$C_o = C_f + C_b (1 - \alpha_f)$$

$$\alpha_o = \alpha_f + \alpha_b (1 - \alpha_f)$$

- Stops when composited α reaches 1
- Requires order: Either FTB or BTF

Voxels (Volume Rendering)

Voxel: 3-D pixel, lengths $\Delta x, \Delta y, \Delta z$

↳ Each grid point has scalar value $f(x,y,z)$

↳ $\Delta x = \Delta y = \Delta z$ → Structured volume dataset

Voxelize solids into cube volume

↳ Voxel grid order depends on viewer location

Transfer Function

Maps discrete densities into color gradient

Marching Cubes (Volume Rendering)

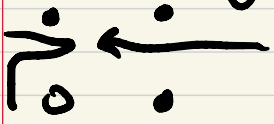
- Generates isosurface
- Define $f_{ijk} = c$

↳ Color vertices white → $f_{ijk} < c$

black → else

- Within marching surface, isosurface passes through contrasts

Aliasing



Not accounted for

↳ Anti-Aliasing: Sample more! (Super sampling)

Splatting (Volume Rendering)

- Project voxels onto screen
- Splats are rendered and composited as disks on screen
  - ↳ Splatted in FTB or BTF order in respect to viewer

V-Buffer (Volume Rendering)

- No creation of intermediate volume set
- Ray cast through volume
  - ↳ Step through ray by incrementing t
  - ↳ Increasing t increment increases step size

Ray parametric: $p + td$

p: pixel location; d: direction

2 pipelines:

Color : $c = c_1 + c_2(1 - \alpha_1)$

Opacity : $\alpha = \alpha_1 + \alpha_2(1 - \alpha_1)$