CSCD18

Lec 01: Image Formation

10 September

What does it take to make an image?

Light source

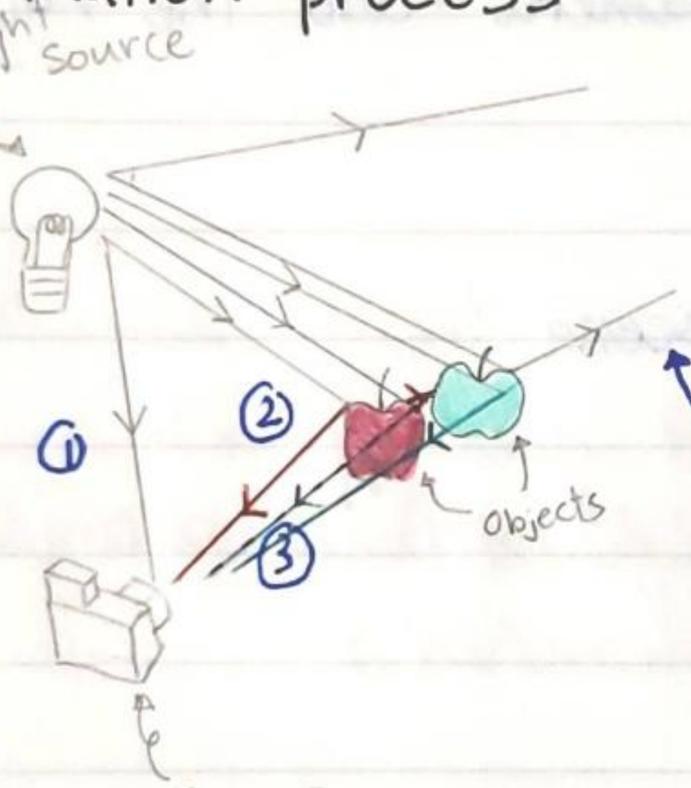
- Sun, LED, fluorescent, chemical, heat, lightning, etc.

Sensor

- film, CCD, CMOS, retina (digital camera)

Objects

Image Formation process

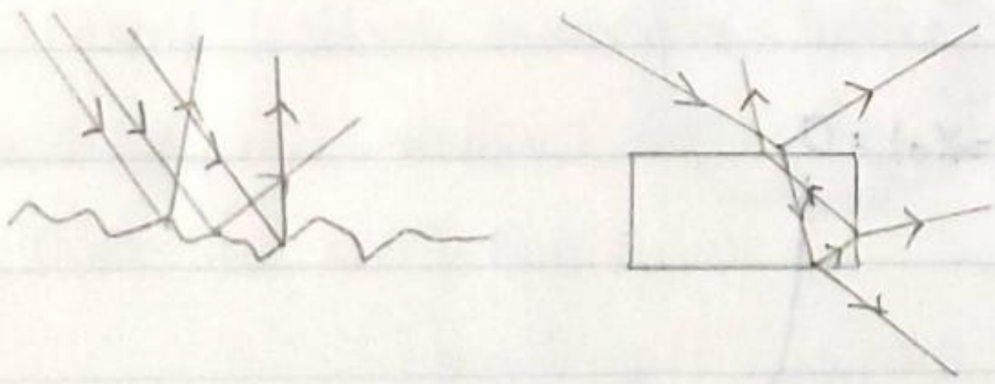


Light can ① go straight to the sensor, ② bounce on an object to reach the sensor, bounce ③ multiple times on objects to reach the sensor, or leave the scene (never reach sensor).

When light hits an object, the colour changes (gets absorbed) and reflects to a different angle

(Computation heavy because many bounces and many rays!)

Uneven Surfaces



Difficult to simulate how light reflect/refract/disperse nicely due to different surface and material

Real Materials are Complex

some are very difficult to simulate well

e.g. human skin / face

porcelain

liquid

Simplest (yet complete) Image Rendering Algorithm: Forward Ray Tracing

Set up the scene (put objects, light sources, and location of camera)

For (a very large number of times)

(Simulate) a single ray from one light source:

Bounce the ray as needed given the objects in the scene until:

Ray hits the camera - add the ray colour to the image

or

Ray leaves the scene (i.e. it doesn't hit anything and goes off into empty space) or a pre-defined max number of bounces is reached.)

LEC 02 - LINES, CURVES & TRANSFORMATION 17 September

$\bar{p}$ = point $\vec{d}$ = vector (although both represented as vector)

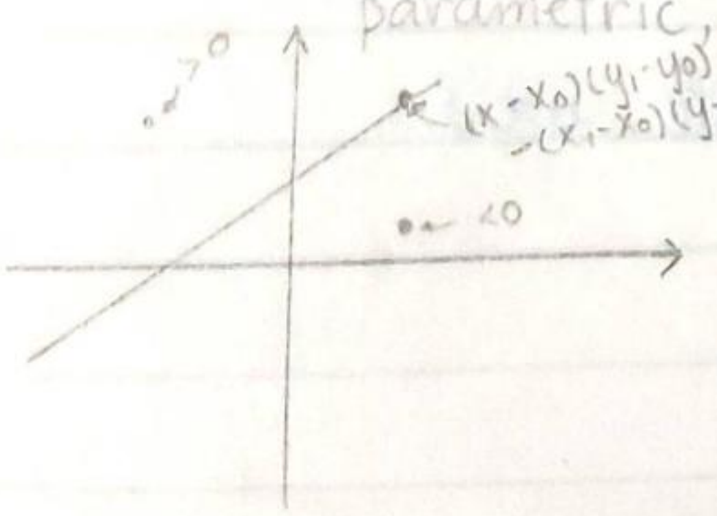
Lines (2D)

Explicit: $y = mx + b$ ($\to \infty$ for vertical) (problem: cannot represent vertical lines)

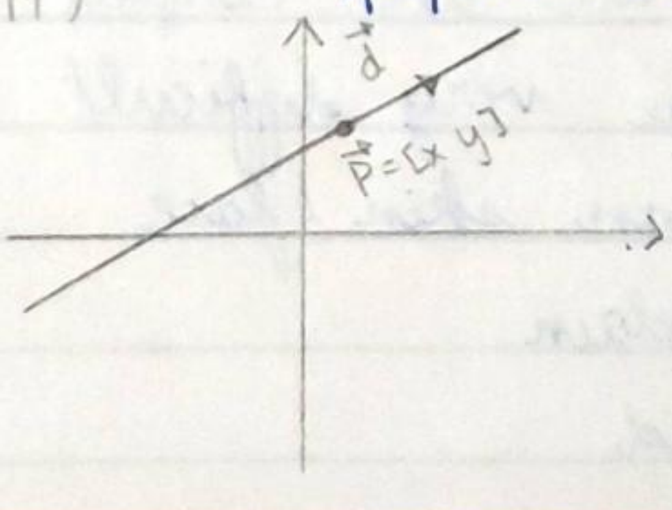
Implicit: $(x - x_0)(y_1 - y_0) - (y - y_0)(x_1 - x_0) = 0$ (can evaluate any pair of (x, y) and if = 0 the point on the line)

Parametric: $\vec{l}(\lambda) = \vec{p} + \lambda\vec{d}$

(If there is a "parameter", then parametric, else, implicit)



$(\bar{p} - \bar{p}_0) \cdot \vec{n} = 0$



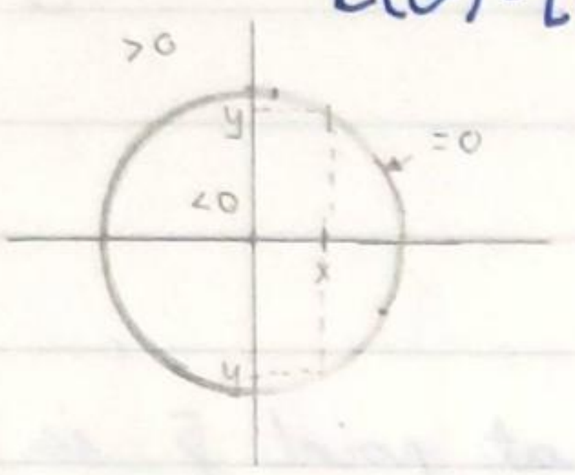
Circles (Ellipses)

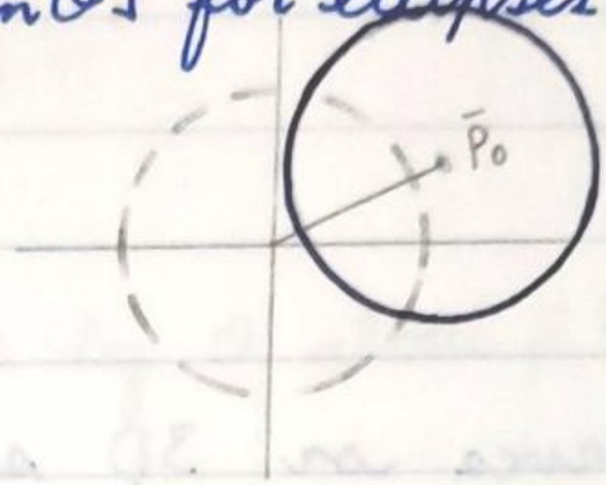
Explicit : N/A

Implicit : $x^2 + y^2 - r^2 = 0$

Parametric : $\bar{C}(\theta) = [r\cos\theta, r\sin\theta]$, $\theta \in [0, 2\pi]$

$\bar{E}(\theta) = [a\cos\theta, b\sin\theta]$ for ellipses



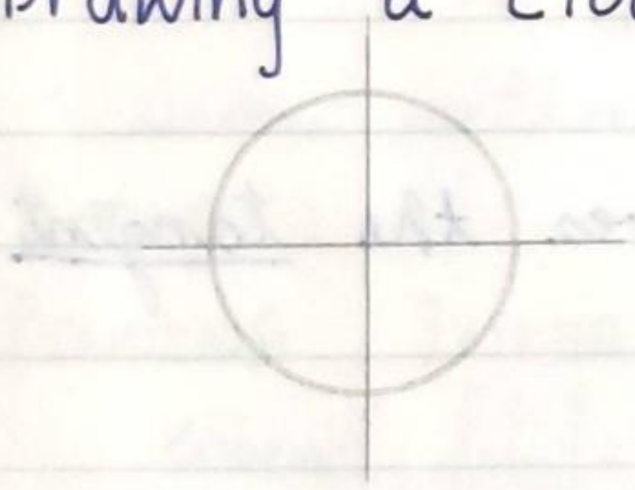


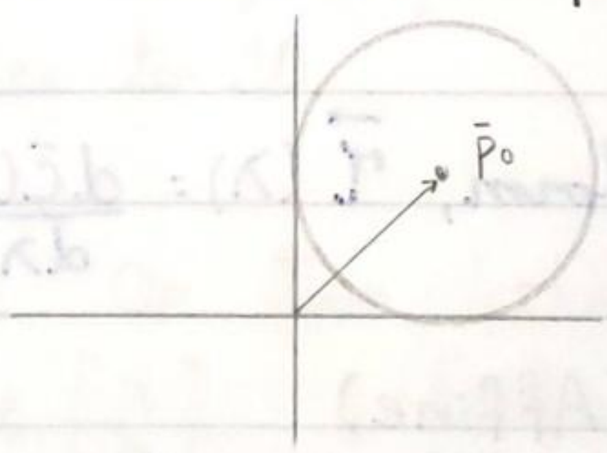
circle with origin at $\bar{p}_0 = [x_0 \;\; y_0]$

$\bar{C}_2(\theta) = [r\cos\theta + x_0 \quad r\sin\theta + y_0]$

$= \bar{C}(\theta) + \bar{p}_0$

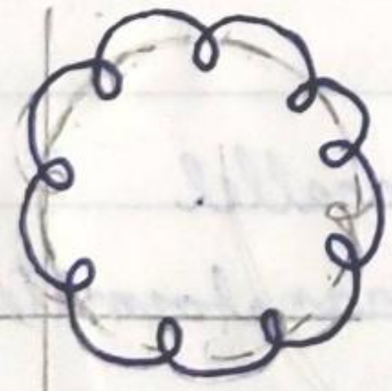
Drawing a cloud from a circle (ellipse)





$\bar{C}_0(\theta) = [r_0\cos\theta \quad r_0\sin\theta]$

$\bar{C}_1(\theta) = [r_0\cos\theta + x_0 \quad r_0\sin\theta + y_0]$



$\bar{C}_2(\theta) = [r_0\cos\theta + x_0 + r_2\cos(n_2\theta)$

$r_0\sin\theta + y_0 + r_2\sin(n_2\theta)]$

$\theta \in [0, 2\pi]$, $n_2 > 1$, $r_2 < r_0$

Intersection of 2 lines

l in parametric form $\bar{l}(\lambda) = \bar{p}_0 + \lambda\vec{d}_0$

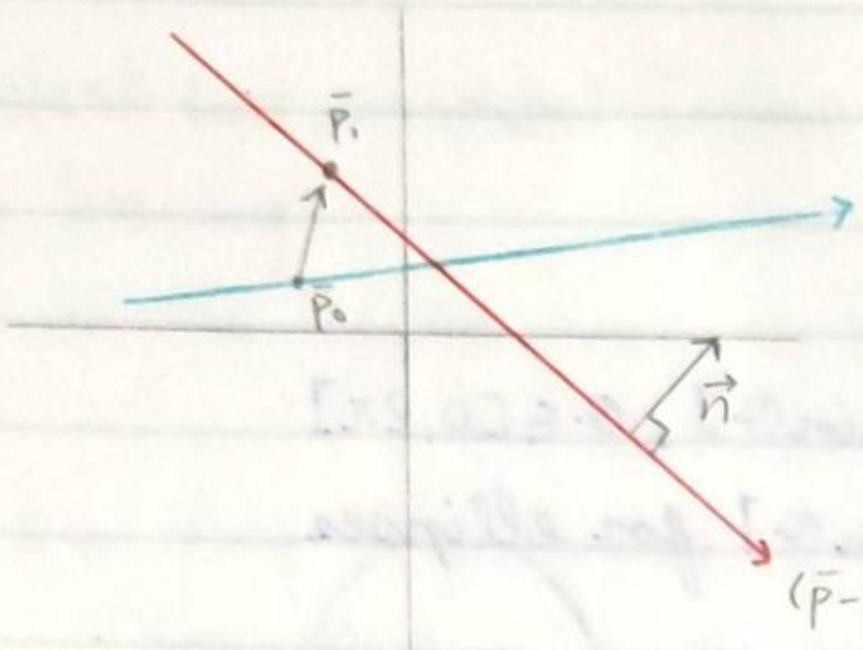
l in implicit form $f(x,y) = (\bar{p} - \bar{p}_1)\cdot\vec{n} = 0$

$\Big\}\ (\bar{p}_0 + \lambda\vec{d} - \bar{p}_1)\cdot\vec{n} = 0$

$\Rightarrow \lambda = \dfrac{(\bar{p}_1 - \bar{p}_0)\cdot\vec{n}}{\vec{d}_0\cdot\vec{n}}$

$\vec{d}_0\cdot\vec{n} = 0 \Rightarrow$ parallel : no point of intersection

$(\bar{p}_1 - \bar{p}_0)\cdot\vec{n} = 0 \Rightarrow$ same line



$$l_0(\lambda) = \bar{P}_0 + \lambda\vec{d}_0$$

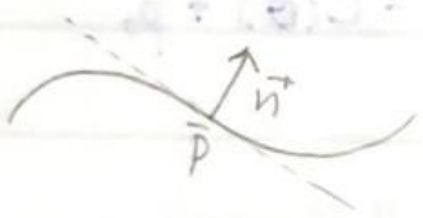
$$= |\bar{P}_1 - \bar{P}_0||\vec{n}|\cos\theta_0$$

$$\lambda = \frac{(\bar{P}_1 - \bar{P}_0)\cdot\vec{n}}{\vec{d}_0\cdot\vec{n}}$$

$$|\vec{d}_0||\vec{n}|\cos\theta_1$$

Gradient ∇f

For 2D curves or 3D surfaces, $\vec{n}$ at point $\bar{p}$ is $\nabla f(x,y)|_{\bar{p}}$ or $\nabla f(x,y,z)|_{\bar{p}}$



For parametric form, $\vec{r}(\lambda) = \frac{d\bar{c}(\lambda)}{d\lambda}$ gives the <u>tangent</u>

Transformations (Affine)

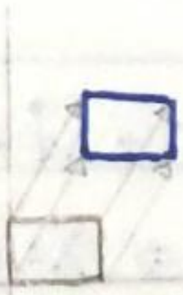
properties

lines map to lines

pal parallel lines remain parallel

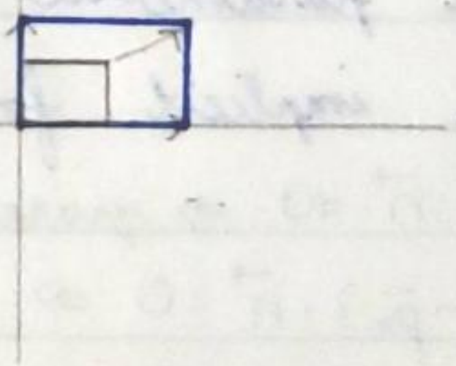
a composition of affine transformation is still affine

Translation : $\bar{q} = \bar{p} + \vec{t}$

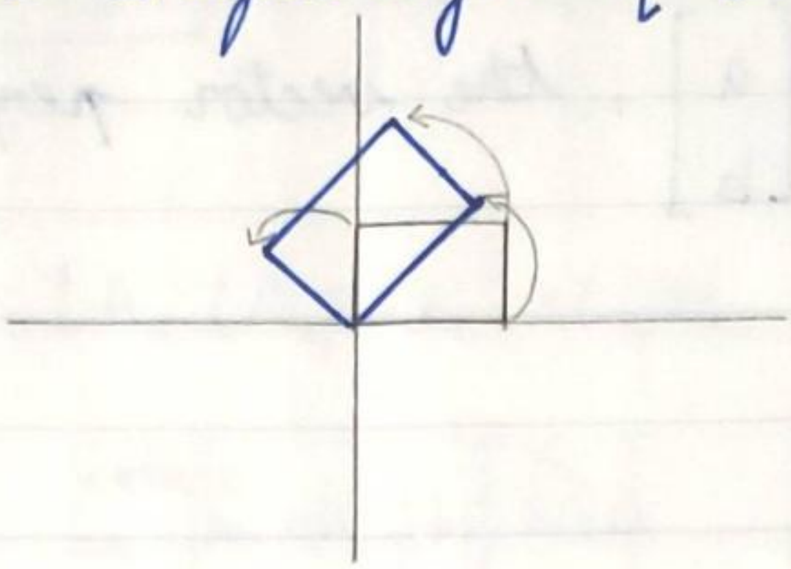


Scaling: $\bar{q} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}\bar{p}$

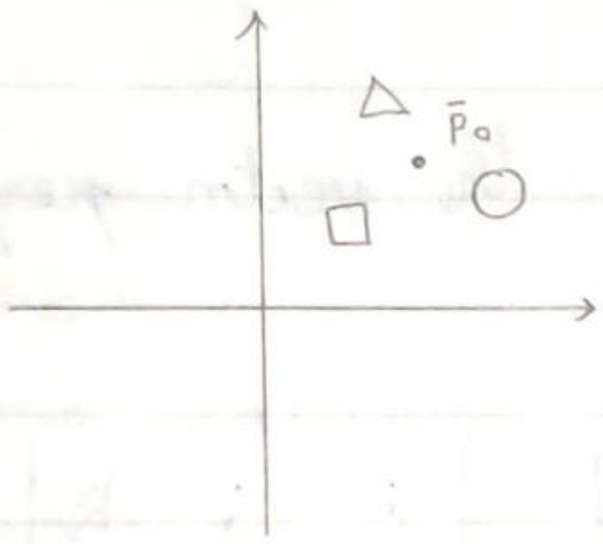
$a=b \Rightarrow$ uniform scaling



Rotation around origin by θ : $\bar{q} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \bar{p}$



e.g.



Rotate by θ about $\bar{p}$

1. move everything to origin
   $\bar{q}_1 = \bar{p} + \vec{t}$ where $\vec{t} = -\bar{p}$

2. rotate by θ
   $\bar{q}_2 = A\bar{q}_1$

3. move back to $\bar{p}$
   $\bar{q}_3 = \bar{q}_2 + \vec{t}_2$ $\vec{t}_2 = \bar{p}$

Same method applies to 3D

TUT-01

FINDING INTERSECTION OF TWO LINES

Given $\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + t_1 \begin{bmatrix} a \\ b \\ c \end{bmatrix}$, $\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \bar{x}_0 \\ \bar{y}_0 \\ \bar{z}_0 \end{bmatrix} + t_2 \begin{bmatrix} d \\ e \\ f \end{bmatrix}$

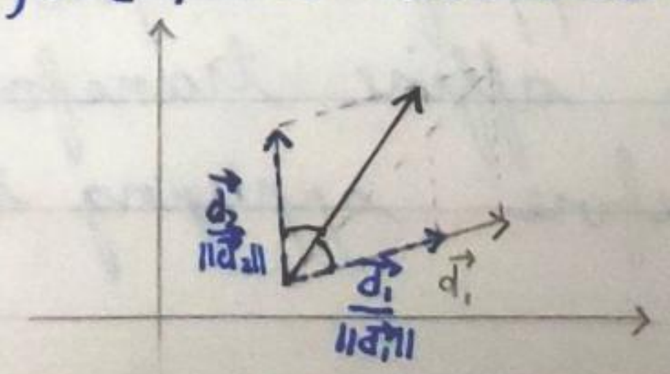
can find point(s) of intersection by solving

$$\left[\begin{array}{cc|c} a & -d & \bar{x}_0 - x_0 \\ b & -e & \bar{y}_0 - y_0 \\ c & -f & \bar{z}_0 - z_0 \end{array}\right]$$ to find t_1 and t_2

ANGLE BISECTOR

Given two direction vectors $\vec{d}_1, \vec{d}_2$, the direction of the angle bisector $= \frac{\vec{d}_1}{\|\vec{d}_1\|} + \frac{\vec{d}_2}{\|\vec{d}_2\|}$

(same for 3D)



PERPENDICULAR LINE

Given a vector $\begin{bmatrix} a \\ b \end{bmatrix}$, the vector perpendicular to it

$$= \begin{bmatrix} -b \\ a \end{bmatrix}$$

check: $\begin{bmatrix} a \\ b \end{bmatrix} \cdot \begin{bmatrix} -b \\ a \end{bmatrix} = -ab + ab = 0$

Given two vectors $\begin{bmatrix} a \\ b \\ c \end{bmatrix}, \begin{bmatrix} d \\ e \\ f \end{bmatrix}$, the vector perpendicular to both vectors $= \begin{bmatrix} a \\ b \\ c \end{bmatrix} \times \begin{bmatrix} d \\ e \\ f \end{bmatrix} =$ determinant $\begin{vmatrix} i & j & k \\ a & b & c \\ d & e & f \end{vmatrix}$

LEC 03: AFFINE TRANSFORM

24 September 2021

HOMOGENEOUS COORDINATES

Augment a point $\bar{p} = \begin{bmatrix} x \\ y \end{bmatrix}$ or $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ by adding an extra component 1.

$\hat{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ or $\hat{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ so that applying an affine transform becomes:

homogeneous point

scale/rotate: $\begin{bmatrix} A_{2\times2/3\times3} \\ 0\ 0\ (0) \end{bmatrix}$ translate: $\begin{matrix} t_x \\ t_y \\ (t_z) \\ 1 \end{matrix}$

$$\hat{q} = \hat{A}\hat{p} = \begin{bmatrix} \begin{bmatrix} A \\ 2\times2/3\times3 \end{bmatrix} & \begin{matrix} t_x \\ t_y \\ (t_z) \end{matrix} \\ 0\ 0\ (0) & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ (z) \\ 1 \end{bmatrix}$$

To apply multiple affine transforms, just <u>left</u> multiply the affine transform matrix (or compute them all before applying to a point)

$$\hat{q} = \underbrace{\hat{A}_3(\hat{A}_2(\hat{A}_1\hat{p}))}_{\hat{A}_T\,\hat{p}} \quad \text{for } T_1 \to T_2 \to T_3$$

$\hat{A}_T$: transform

instead of $A_3[A_2(A_1\bar{p} + t_1) + t_2] + t_3$

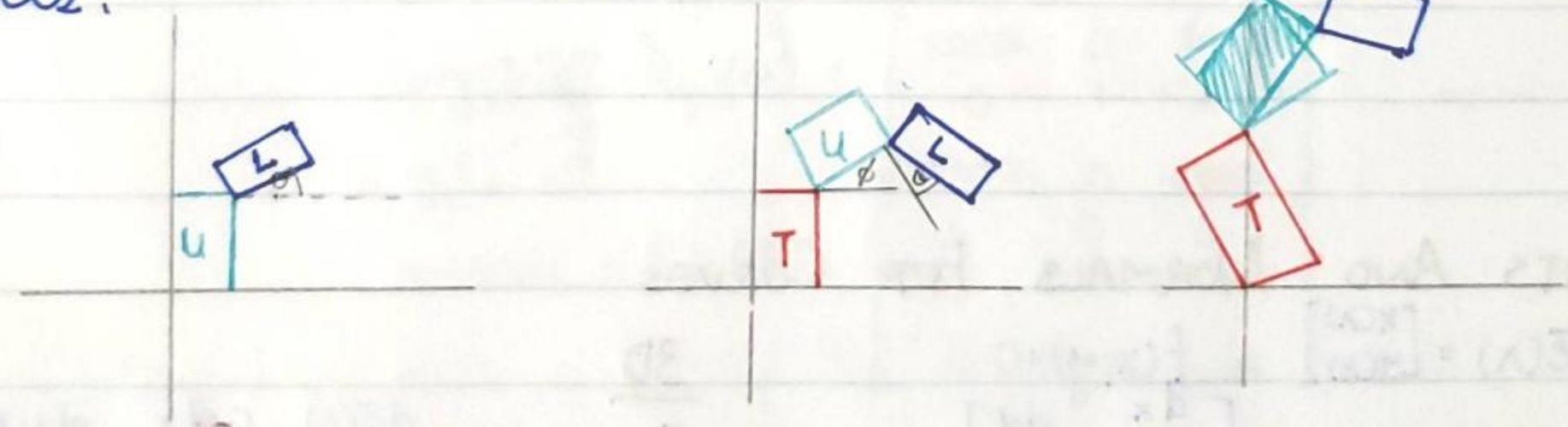
From $\hat{q} = \begin{bmatrix} x_q \\ y_q \\ z_q \\ w_q \end{bmatrix}$ $\xrightarrow{\text{reduce to 3D}}$ $\bar{q} = \begin{bmatrix} \frac{x_q}{w_q} \\ \frac{y_q}{w_q} \\ \frac{z_q}{w_q} \end{bmatrix}$

Homogeneous coordinates have property that any scalar multiple of a point in homogeneous coordinates represent the same cartesian point.

e.g. $\hat{q}_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$ $\hat{q}_2 = \begin{bmatrix} 2 \\ 4 \\ 6 \\ 2 \end{bmatrix}$ both represent $\bar{q} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

HIERARCHICAL TRANSFORM

Define transforms for parts with respect to parent pieces.



$M_L = T(1,2)S(.8)R(\theta)$

3. translate to (1, 2) 2. scale by .8 1. rotate by θ anticlockwise

$M_U = T(1,2)S(.9)R(\phi)$

$M_T = R(\varphi)$

So T has transform matrix $M_T' = M_T$

U has transform matrix $M_U' = M_T M_U$

L has transform matrix $M_L' = M_T M_U M_L$

* ONLY use uniform scaling as non-uniform scaling will break the hierarchy.

Transformation Matrices - 2D

Translation by $t = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$: $\bar{p}_1 = \begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Rotation anticlockwise by θ: $\bar{p}_1 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Scaling x by a, y by b: $\bar{p}_1 = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

uniform if $a=b$

Shear by h: $\bar{p}_1 = \begin{bmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

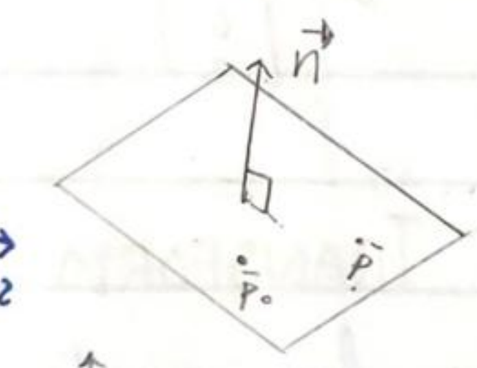
3D Surfaces / Geometry

Planes

Implicit: $(\bar{p} - \bar{p}_0) \cdot \vec{n} = 0$

Parametric: $\bar{S}(\alpha, \beta) = \bar{p}_0 + \alpha \vec{d}_1 + \beta \vec{d}_2$

(line) $\bar{l}(\lambda) = \bar{p}_0 + \lambda \vec{d}$



∝ both on plane, not linear combination of one another

Tangents And Normals For Curves

2D $\bar{c}(\lambda) = \begin{bmatrix} x(\lambda) \\ y(\lambda) \end{bmatrix}$ $f(x,y) = 0$

Tangent: $\frac{d\bar{c}(\lambda)}{d\lambda} = \begin{bmatrix} \frac{dx}{d\lambda}, \frac{dy}{d\lambda} \end{bmatrix}$

Normal: $\nabla f(x,y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x}, \frac{\partial (f(x,y))}{\partial y} \end{bmatrix}$

3D

Tangent: $\frac{d\bar{c}(\lambda)}{d\lambda} = \begin{bmatrix} \frac{dx}{d\lambda}, \frac{dy}{d\lambda}, \frac{dz}{d\lambda} \end{bmatrix}$

Normal: $\nabla f(x,y,z)$

Idea: parametric for tangent, implicit for normal

For 3D surfaces $\bar{S}(\alpha, \beta)$

Tangent Plane: defined by tangent vectors $\frac{\partial S}{\partial \alpha}$ and $\frac{\partial S}{\partial \beta}$.

Normal: $\frac{\partial S}{\partial \alpha} \times \frac{\partial S}{\partial \beta}$

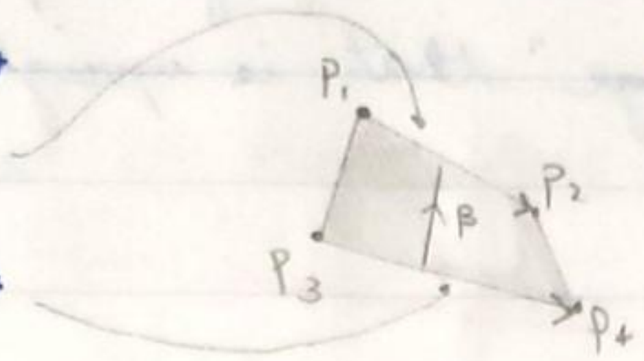
BILINEAR PATCH

given p_1, p_2, p_3, p_4

$\bar{l}_1(\alpha) = (1-\alpha)p_1 + \alpha p_2$

$\bar{l}_2(\alpha) = (1-\alpha)p_3 + \alpha p_4$

for $0 \le \alpha \le 1$

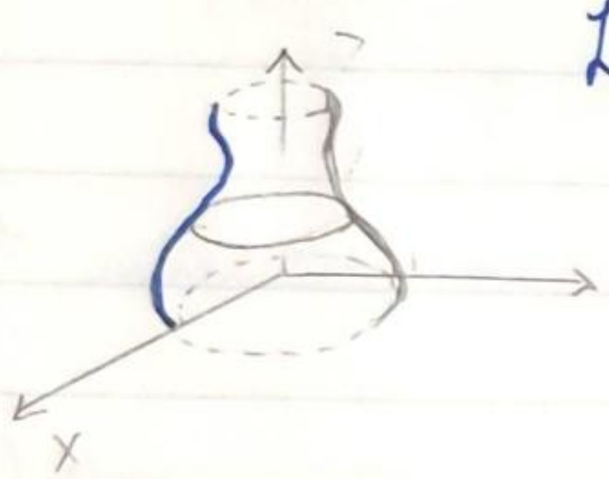


connect $\bar{l}_1$ and $\bar{l}_2$ with a line

$\bar{p}(\alpha,\beta) = (1-\beta)\bar{l}_1(\alpha) + \beta\bar{l}_2(\alpha)$

for $0 \le \alpha, \beta \le 1$

SURFACE OF REVOLUTION



Let a curve be on x-z plane: $\bar{C}(\beta) = (x(\beta), 0, z(\beta))$

rotate about z-axis:

$\bar{S}(\alpha,\beta) = (x(\beta)\cos(\alpha), x(\beta)\sin(\alpha), z(\beta))$

draws a circle with radius $x(\beta)$ parallel to x-y plane at height $= z(\beta)$

TRANSFORM MATRICES - 3D

Translation, & Scaling similar to 2D:

Rotation about z-axis: $R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

y-axis: $R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$

x-axis: $R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$

LEC 04: ALL ABOUT CAMERAS

| October, 2021

Suppose we have a box that has a pinhole that let's exactly one light ray through for one point in screen:

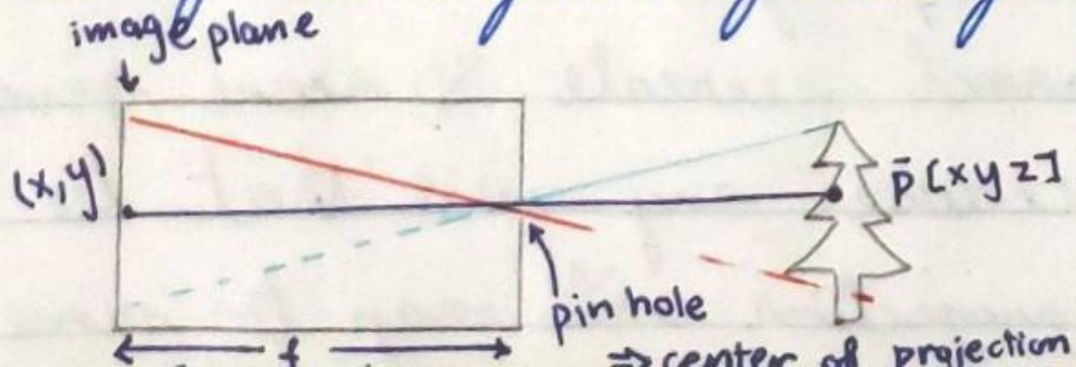
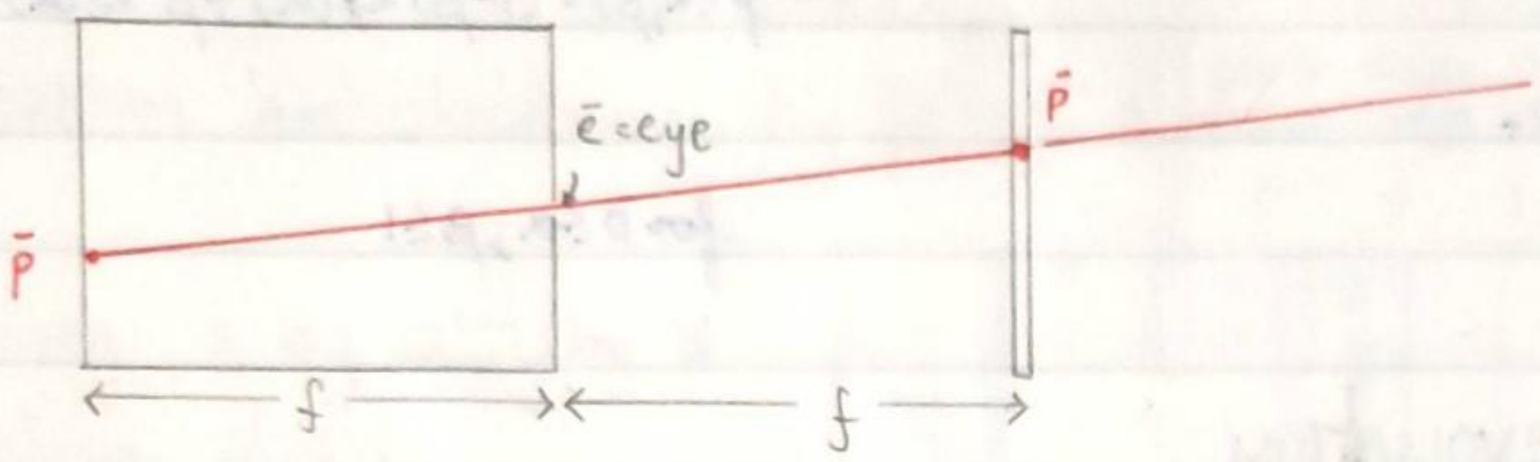
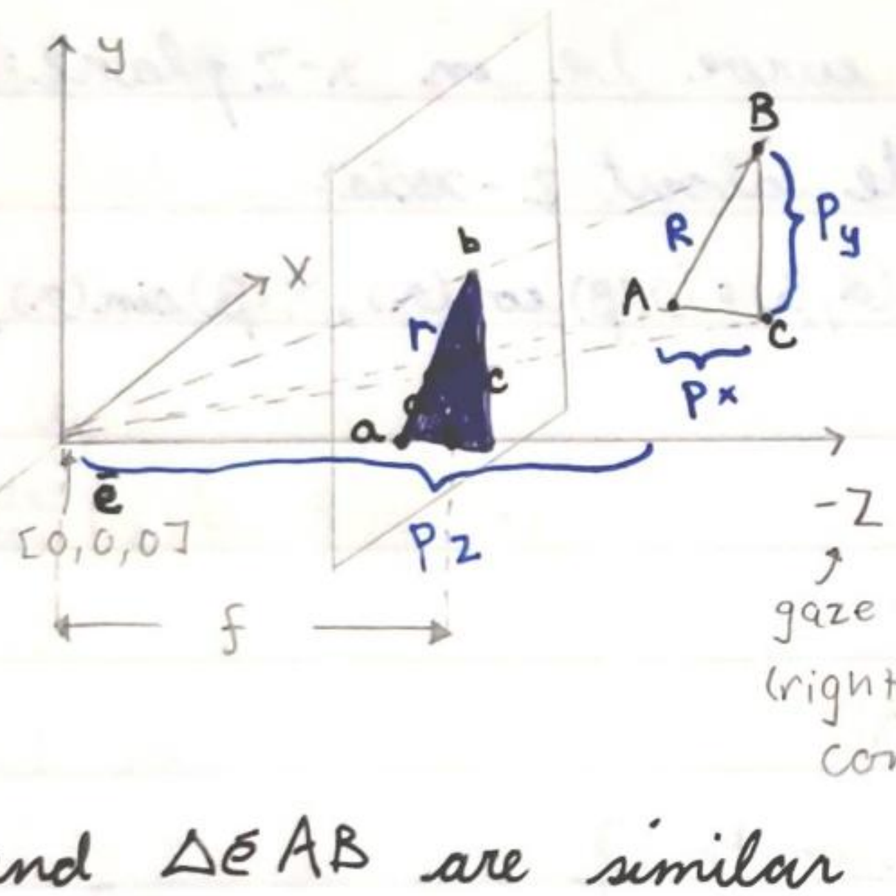


image:  upside-down and reflected left ↔ right

But if we put screen f units away from pinhole, we can sort of get "same image" that is upright





$\Delta \bar{e}ab$ and $\Delta \bar{e}AB$ are similar (assume a and A sits on Z-axis)

$$\Rightarrow \frac{f}{P_z} = \frac{r}{R}$$

also, Δabc and ΔABC are similar,

$$\frac{P_x}{P_x} = \frac{P_y}{P_y} = \frac{r}{R} = \frac{f}{P_z}$$

$$\Rightarrow p_x = \frac{P_x \cdot f}{P_z}$$

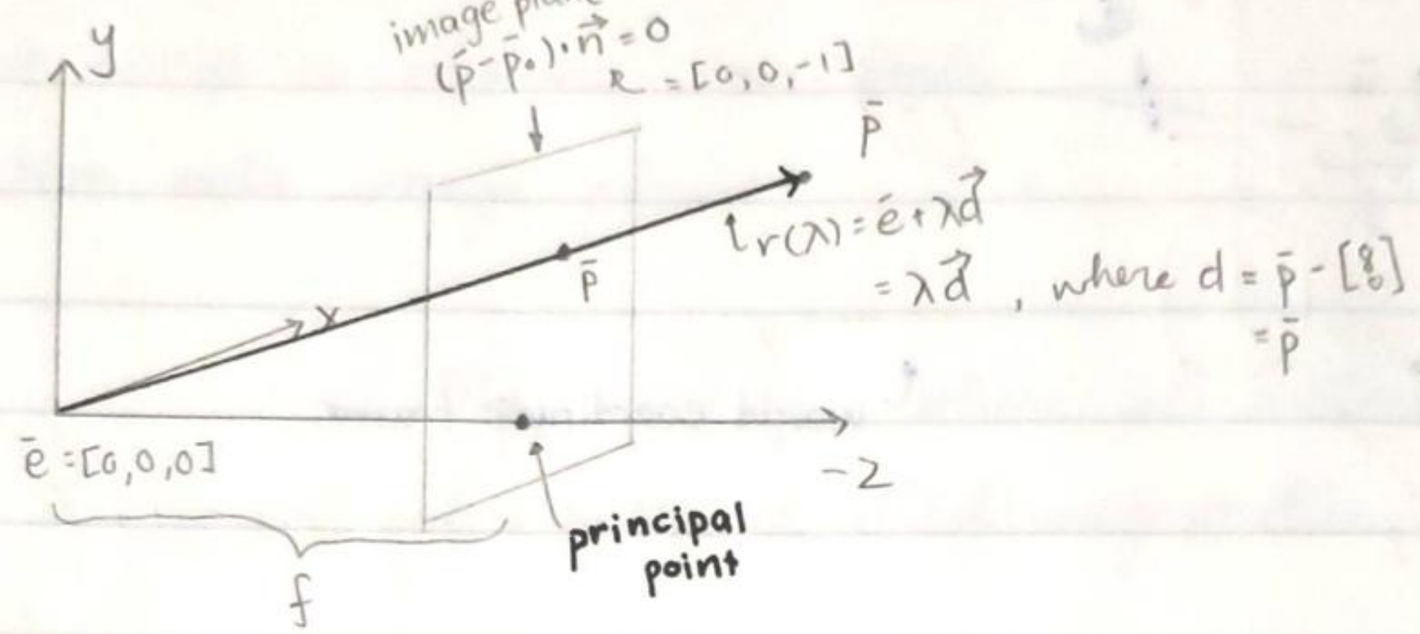
$$p_y = \frac{P_y \cdot f}{P_z}$$

$\therefore$ size of p_x and p_y (size of image on screen) is directly proportional to f.
(P_x, P_y, P_z fixed for an eye position and point of an object)

Note: depth is lost, cannot recreate 3D scene given one 2D image because any point that lie along same line of projection will map to same

point on plane.

FINDING X,Y POINT ON PLANE (~~ALIF~~ USING PLANE EQUATION



⇒ substitute $\bar{e}+\lambda\vec{d}$ into $(\bar{p}-\bar{p}_0)\cdot\vec{n}=0$ to find lambda

3D HOMOGENEOUS PROJECTION MATRIX

$$\begin{bmatrix} P_{xh} \\ P_{yh} \\ P_{zh} \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} fP_x \\ fP_y \\ fP_z \\ P_z \end{bmatrix} \Rightarrow \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} \frac{fP_x}{P_z} \\ \frac{fP_y}{P_z} \\ f \end{bmatrix}$$

↑ inhomogeneous coord ↑ object point ↑ image point

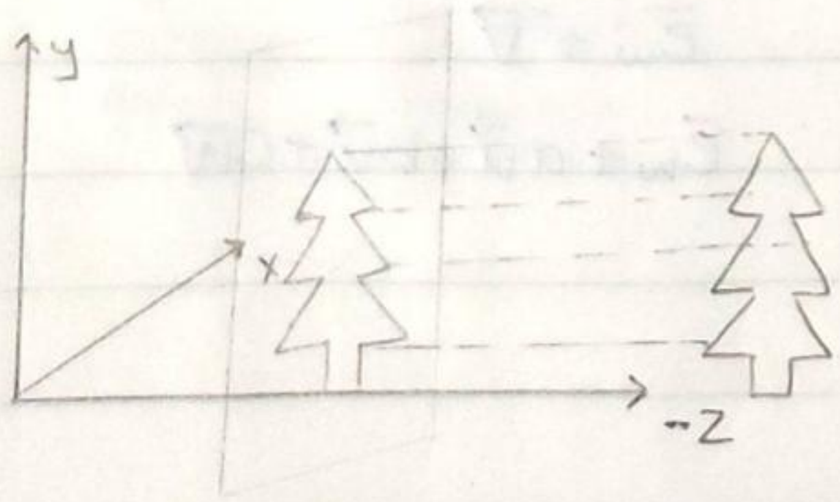
or

$$\begin{bmatrix} P_{xh} \\ P_{xy} \\ P_{zh} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \\ P_z \\ \frac{P_z}{f} \end{bmatrix} \Rightarrow \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} P_x \cdot \frac{f}{P_z} \\ P_y \cdot \frac{f}{P_z} \\ P_z \cdot \frac{f}{P_z} \end{bmatrix}$$

↑ object point ↑ image point

less computation needed!

ORTHOGONAL PROJECTION (useful for CAD, industrial design)

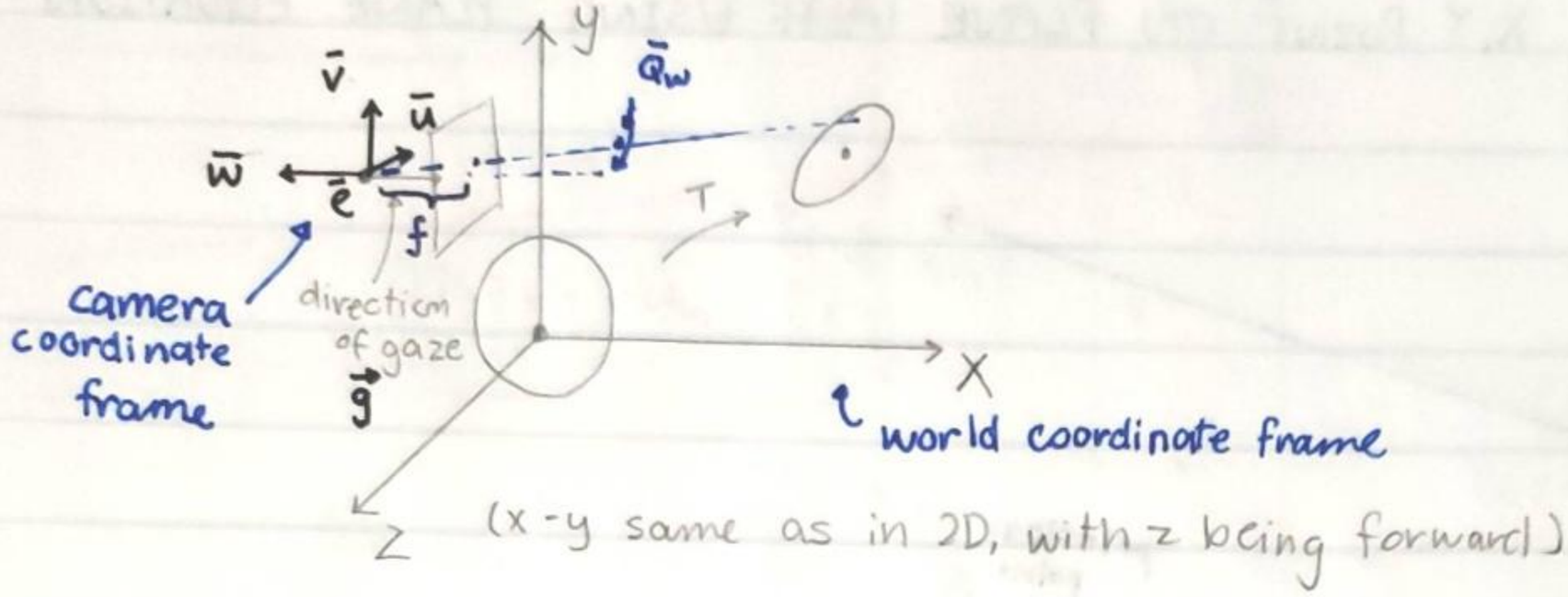


orthogonal Projection Matrix

(dimension/perspective preserves parallelism)

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

COORDINATE FRAMES



Defining Camera Coordinate Frame

1. location of camera's center of projection in WORLD COORD $\bar{e}_w$
2. gaze direction $\vec{g} = \bar{Q}_w - \bar{e}_w$
3. $\bar{w} = \frac{-\vec{g}}{\|\vec{g}\|}$ (normalize)
4. let $\vec{t} = [0, 1, 0]$ be a up direction not parallel to $\bar{w}$.

$$\bar{u} = \frac{\bar{w} \times \vec{t}}{\|\bar{w} \times \vec{t}\|} \quad \Rightarrow u \perp w$$

5. obtain $\bar{v}$ which is orthogonal to both u and w:

$$\bar{v} = \frac{\bar{u} \times \bar{w}}{\|\bar{u} \times \bar{w}\|}$$

CONVERSION:

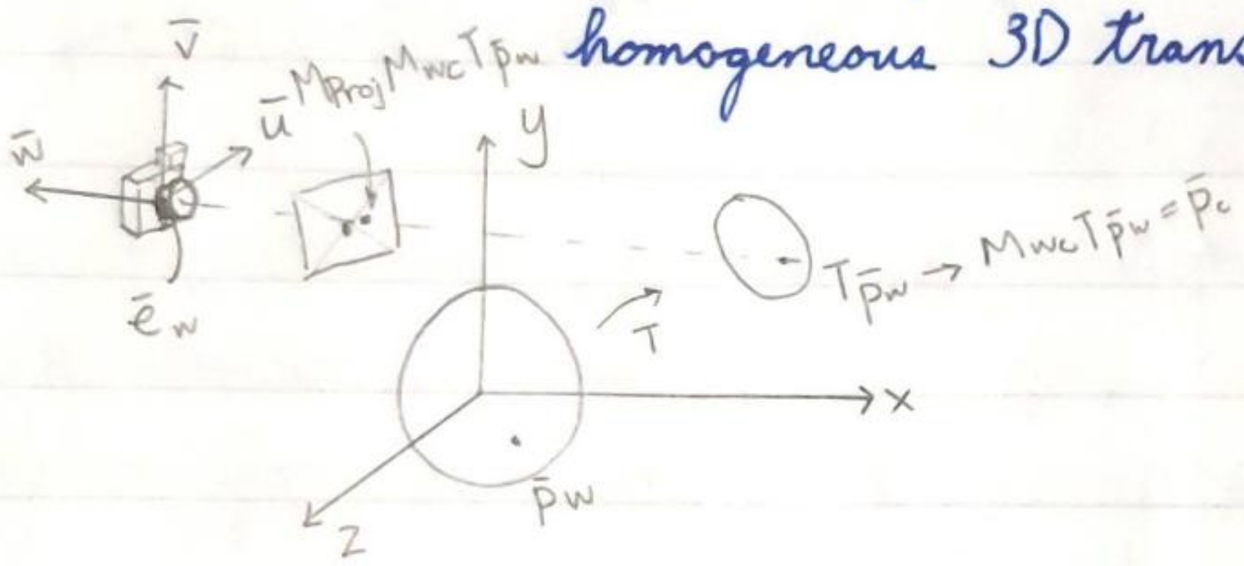
Camera coordinates	World coordinates
[0, 0, 0]	$\bar{e}_w$
[0, 0, 1]	$\bar{e}_w + \vec{w}$
[0, 1, 0]	$\bar{e}_w + \vec{v}$
[a, b, c]	$\bar{e}_w + a\vec{u} + b\vec{v} + c\vec{w}$

$$\Rightarrow \bar{p}_w = \underbrace{\begin{bmatrix} | & | & | \\ \vec{u} & \vec{v} & \vec{w} \\ | & | & | \end{bmatrix}}_{M_{cw}} \bar{p}_c + \bar{e}_w$$

PROJECTION PIPELINE (for an object)

1. affine transform to form the shape and move it to ~~correct~~ location
2. convert world to camera coordinates
3. projection onto image plane

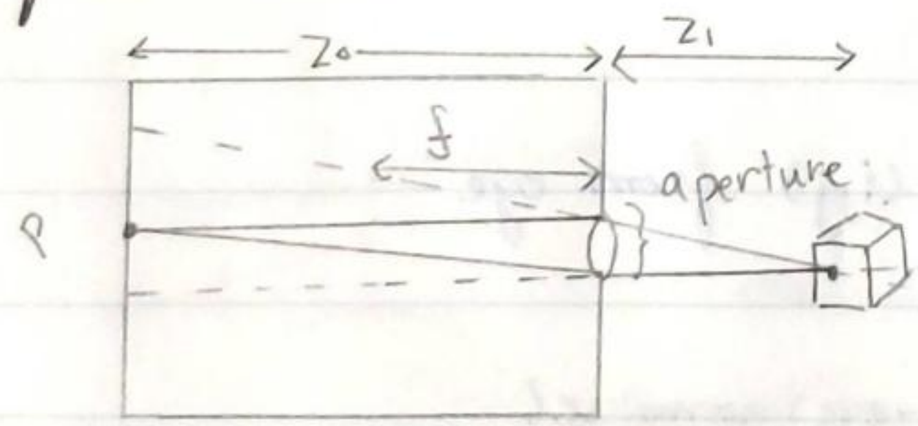
$M_{proj} M_{wc} T\ \bar{p}_w$ where all matrix performs homogeneous 3D transformations



Problem! pinhole doesn't let in enough light! ⇒ increase aperture

But light hitting same point of object end up in more spread locations

↳ lose focus ⇒ add lens!



anything from here will be OUT of focus!

THIN LENS MODEL

$$\frac{1}{|f|} = \frac{1}{z_1} + \frac{1}{z_0}$$

- $|f|$: fixed by lens
- z_1: location where image is in focus
- z_0: fixed

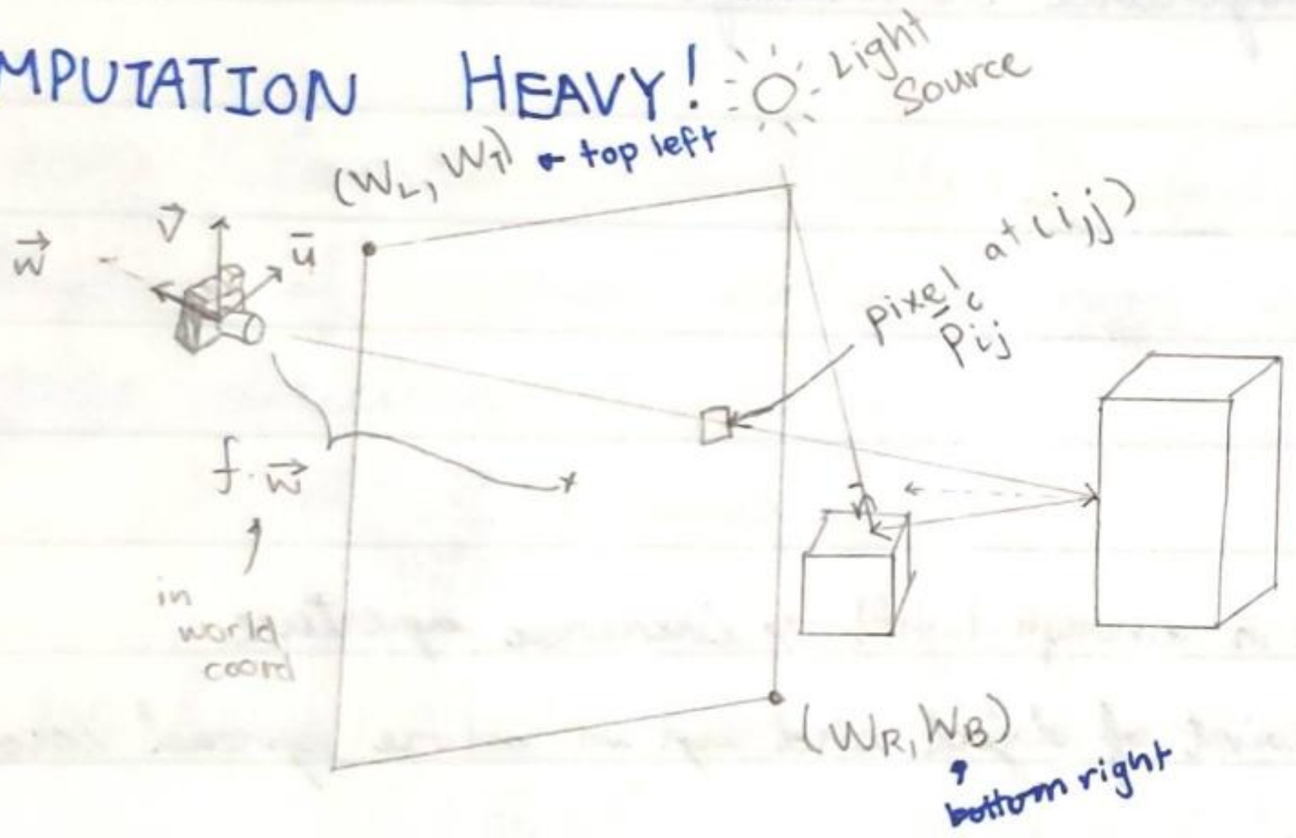
actual cameras will adjust z_0 or f to change z_1.

LEC · 05 · RAY TRACING

FORWARD RAY TRACING

create a light ray from light source
find light source, compute colour
bounce and repeat until hit camera → store that colour
or until light hits off screen

COMPUTATION HEAVY!



BACKWARD RAY TRACING

1. cast a ray through pixel (i,j) from eye

$$\bar{r}^c(\lambda) = \bar{a}_{ij}^c + \lambda \bar{d}_{ij}^c$$

$\bar{a}_{ij}^c$: ~~eye~~ point on (i,j)

$\bar{d}_{ij}^c$: (pixel (i,j) − eye) normalize?

$$\bar{P}_{ij}^c = (W_L + i\Delta u, W_T + j\Delta v, f)$$

$$\Delta u = \frac{W_R - W_L}{n_c - 1} \qquad \Delta v = \frac{W_B - W_T}{n_r - 1}$$

n_c: # col pixels; n_r: # row pixels

Δu, Δv: units depends on camera screen size and # image resolution

1.2. convert to world coordinates using M_{cw} ($\bar{P}_{ij}^c \rightarrow \bar{P}_{ij}^w$)

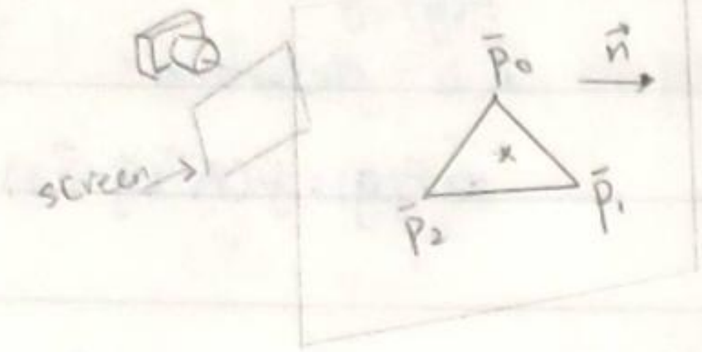
1.3. $\bar{r}_{ij}(\lambda) = \bar{P}_{ij}^w + \lambda \vec{d}_{ij}$ where $\vec{d}_{ij} = (\bar{P}_{ij}^w - \bar{e}^w)$

2. Trace $\bar{r}_{ij}(\lambda)$, get a colour for pixel at (i,j) by
   - check for closest intersection between $\bar{r}_{ij}(\lambda)$ and scene objects.

Intersection Testing

Triangle

plane this triangle lies



normal of $\Delta = \dfrac{(\bar{p}_2 - \bar{p}_0) \times (\bar{p}_1 - \bar{p}_0)}{\|(\bar{p}_2 - \bar{p}_0) \times (\bar{p}_1 - \bar{p}_0)\|} = \vec{n}$ eqn of plane = $(\bar{p} - \bar{p}_1) \cdot \vec{n} = 0$

So for $r_{ij}(\lambda) = \bar{p}_{ij}^{w} + \vec{d}_{ij}$, $\lambda^* = \dfrac{(\bar{p}_1 - \bar{p}_{ij}^{w}) \cdot \vec{n}}{\vec{d}_{ij} \cdot \vec{n}}$

So $\bar{r}_{ij}(\lambda^*)$ gives point of intersection between plane and ray

Check whether in triangle:

Solve for $\bar{p}(\alpha, \beta) = \bar{p}_0 + \alpha(\bar{p}_1 - \bar{p}_0) + \beta(\bar{p}_2 - \bar{p}_0) = \bar{p}_{ij} + \lambda\vec{d} = \bar{r}(\lambda)$

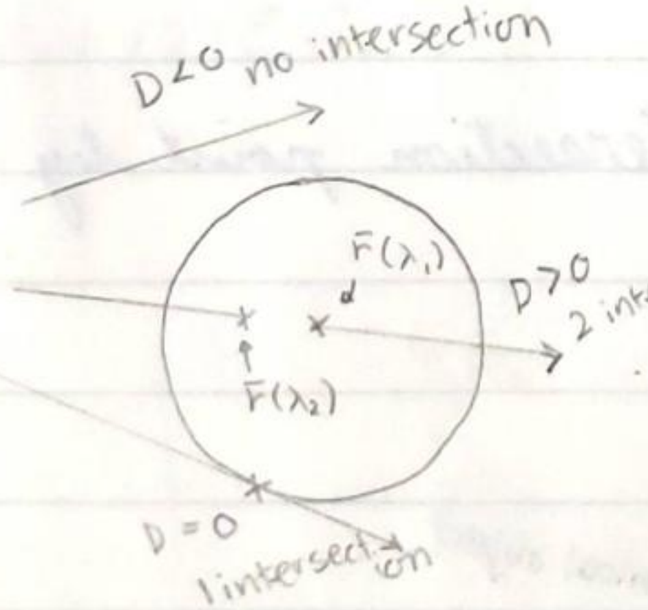
$$\Rightarrow \begin{pmatrix} | & | & | \\ -(\bar{p}_1 - \bar{p}_0) & -(\bar{p}_2 - \bar{p}_0) & \vec{d} \\ | & | & | \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \lambda \end{pmatrix} = (\bar{p}_0 - \bar{p}_{ij})$$

The intersection is in the triangle if $\alpha \geq 0$

$\beta \geq 0$

$\alpha + \beta \leq 1$

Sphere



unit sphere centered at [0,0,0] : $\|\bar{p}\|^2 = 1$

$\bar{c}$: $\|\bar{p} - \bar{c}\|^2 = 1$

$\Rightarrow (\bar{p}_{ij} + \lambda\vec{d} - \bar{c}) \cdot (\bar{p}_{ij} + \lambda\vec{d} - \bar{c}) - 1 = 0$

$\hookrightarrow A\lambda^2 + 2B\lambda + C = 0$

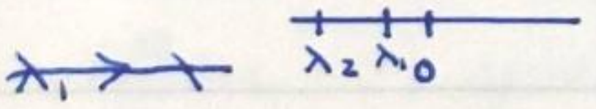
where $A = \vec{d} \cdot \vec{d}$

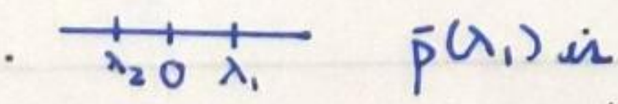
$B = (\bar{p}_{ij} - \bar{c}) \cdot \vec{d}$

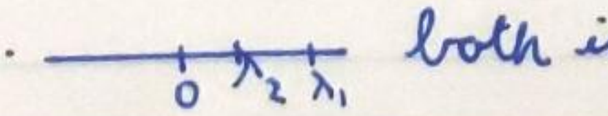
$C = (\bar{p}_{ij} - \bar{c}) \cdot (\bar{p}_{ij} - \bar{c}) - 1$

$\Rightarrow \lambda = \dfrac{-2B \pm \sqrt{4B^2 - 4AC}}{2A} = \dfrac{-B}{A} \pm \dfrac{\sqrt{D}}{A}$

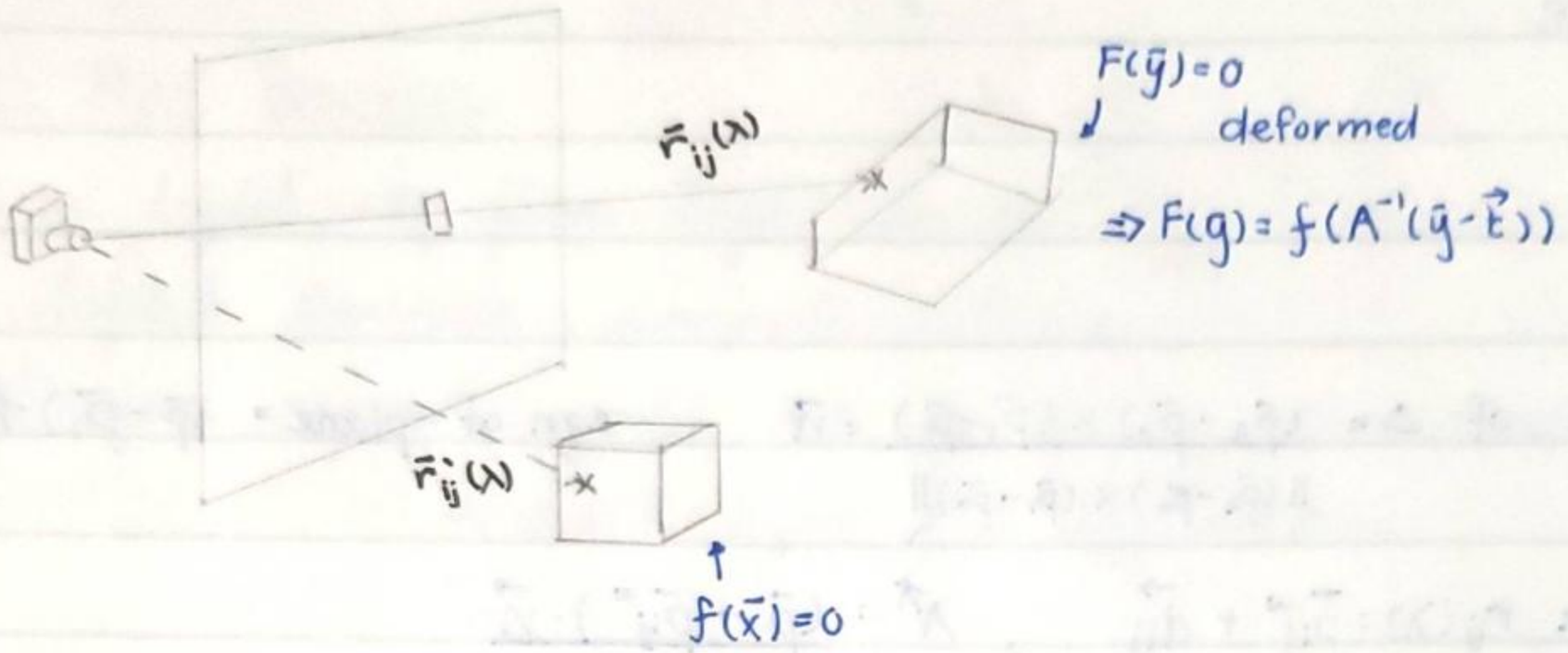
where $D = B^2 - AC$

For D>0, 1.  behind view-plane, not visible

2.  $\bar{p}(\lambda_1)$ is visible, $\bar{p}(\lambda_2)$ is not

3.  both in front of view-plane, $\bar{p}(\lambda_2)$ is closest intersection

Intersection With Deformed Object



Let $\bar{y}=A\bar{x}+\vec{t}$ be the transformation from canonical object to deformed object.

Given we know the intersection method for $f(\bar{x})=0$.

Substitute $\bar{r}(\lambda)$ ~~sto~~ into $f=F(\bar{y})$

$$F(\bar{r}(\lambda)) = f(A^{-1}(\bar{r}(\lambda)-\vec{t}))$$
$$= f(A^{-1}(\bar{p}+\lambda\vec{d}-\vec{t}))$$
$$= f(\bar{a}'+\lambda\vec{d}')$$
$$= f(\bar{r}'(\lambda))$$

where $\bar{a}' = A^{-1}(\bar{p}-\vec{t})$

$\bar{d}_i = A^{-1}\vec{d}$

Using $\bar{a}'$ and $\bar{d}'$, find intersection point by substitute λ^* to $\bar{r}(\lambda)$

Normal With Deformed Objects

$$\vec{n} = \frac{(A^{-1})^T\vec{n}'}{\|(A^{-1})^T\vec{n}'\|}$$ ← normal of canonical object

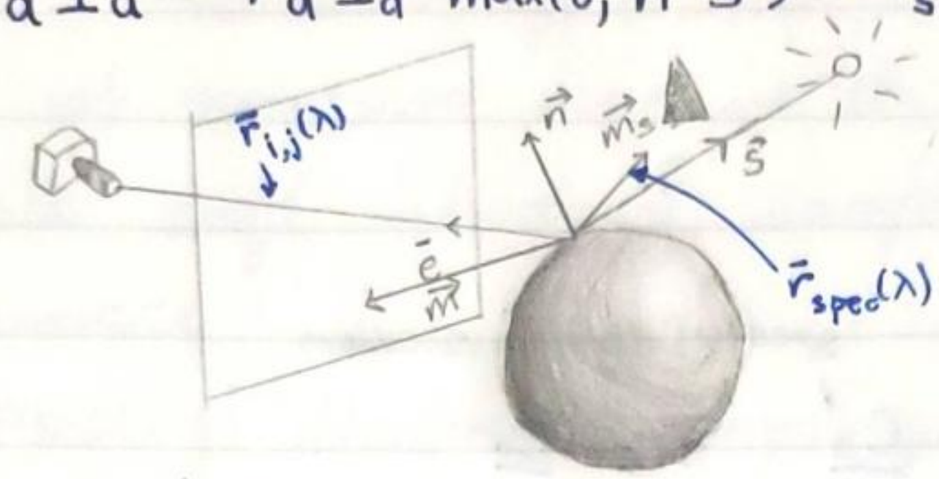
Scene Signature

assign colours to objects to check ~~ob~~ whether intersection of objects done correctly.

PHONG MODEL

$$E = r_a I_a + r_d I_d \max(0, \vec{n}\cdot\vec{s}) + r_s I_s \max(0, \vec{e}\cdot\vec{m})^{\alpha} + r_g I_g$$

(light source; obtain by recursive ray trace; ray from camera; light perfect reflection)



(all vectors point away from surface)

where r = reflectivity on types of light (albedo)

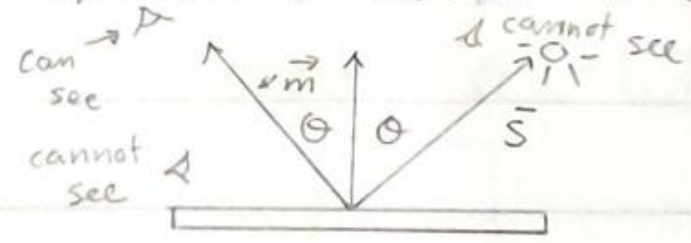
I = intensity of light / colour

a = ambient — illumination due to bounced (indirect light)

d = diffuse — depends on angle between surface and light (NOT viewer location)

s = specular — depends on viewer location and light perfect reflection

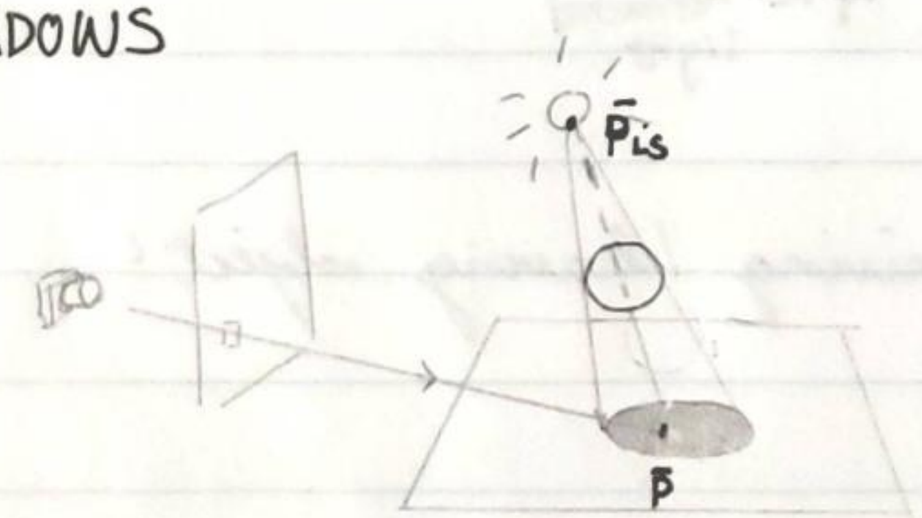
g = global — From recursive ray trace



$$\vec{m} = -\vec{s} + (2(\vec{s}\cdot\vec{n}))\vec{n}$$

$$\vec{m}_s = -\bar{e} + (2(\vec{c}\cdot\vec{n}))\vec{n}$$

SHADOWS



For each light source,

create shadow ray $\bar{sr}(\lambda) = \bar{p} + \lambda \vec{d}_{ls}$

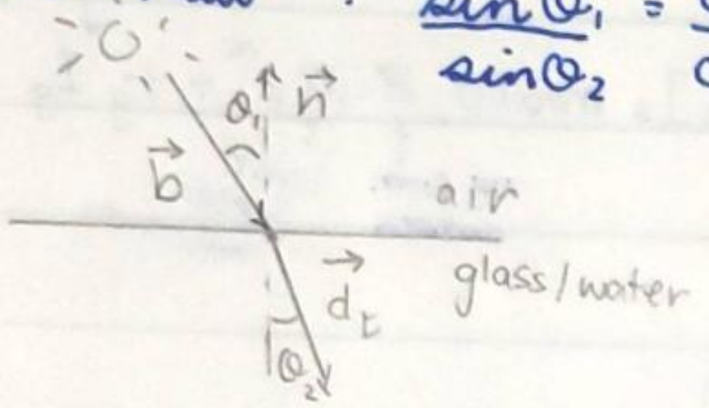
$(\bar{P}_{LS} - \bar{p})$ ← NOT normalized.

If an object is in between $\bar{p}$ and $\bar{P}_{LS}$, then raytrace($\bar{sr}(\lambda)$) will find a λ^* where $0 < \lambda^* < 1$.

Then set diffuse = 0, specular = 0 (only have ambient and global)

Transparency And Reflection

Snell's Law : $\frac{\sin\theta_1}{\sin\theta_2} = \frac{c_1}{c_2}$ ← speed of light in air



← speed of light in medium

index of refraction: $n_1 = \frac{c_0}{c_1}$ $n_2 = \frac{c_0}{c_2}$ ← speed of light in vaccuum

$c_2 < c_1 \Rightarrow n_2 > n_1 \Rightarrow \theta_2 < \theta_1$

Transmitted Ray Direction

Let $c = -\vec{n}\cdot\vec{b}$ $r = \frac{c_2}{c_1} = \frac{n_1}{n_2}$

$$\vec{d_t} = r\vec{b} + \left(rc - \sqrt{1 - r^2(1-c^2)}\right)\vec{n}$$

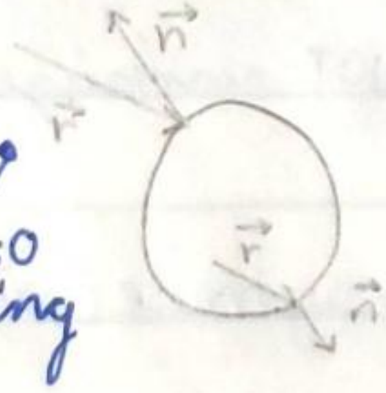
Phong model:

$$E_{\bar{p}} = \underbrace{\alpha_t}_{\text{transparency}}\left(r_a I_a + r_d I_d \max(0, \vec{n}\cdot\vec{s}) + r_s I_s \max(0, \vec{e}\cdot\vec{m})^{\alpha} + r_g I_g\right) + (1-\alpha_t)\,\text{raytrace}(\bar{r}_t(\lambda))$$

(α: shininess; $r_g I_g$: obtained from reflection; raytrace: similar to global by but refracted light)

$\alpha_t \in [0, 1]$ (0: transparent, 1: opaque)

How to check whether light arriving / leaving object?



$\vec{r}\cdot\vec{n} < 0$ arriving

$\vec{r}\cdot\vec{n} > 0$ leaving

Lec 07 Advanced Ray Tracing

Area Light Sources

An object of non-zero size in the scene

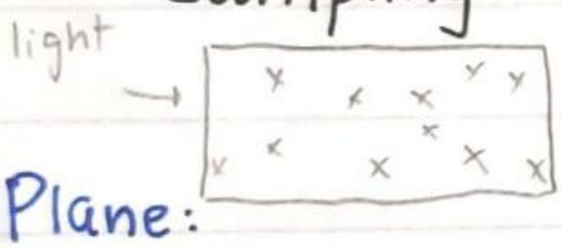
To ~~as~~ create <u>soft</u> shadows, sample points in / on surface of light source ← depends on context

Random Sampling! Uniform sampling will create shoot shadows that looks like layers:

with random sampling

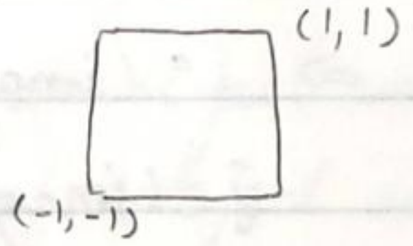
with uniform sampling

✡ Diffuse and Specular terms different for each sample!
i.e. recalculate $\vec{s}$ and $\vec{m}$

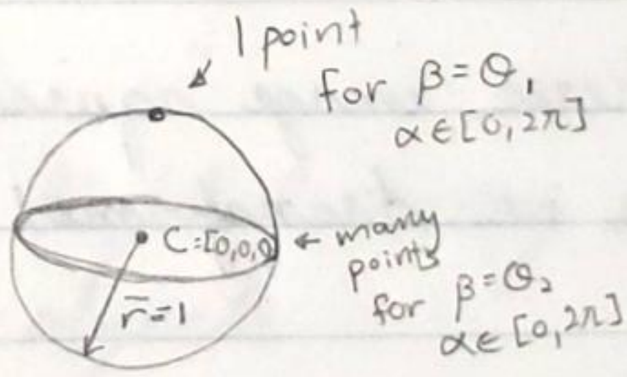


Plane:

1. $r_x = (drand48() * 2) - 1$; // a number $\in [-1, 1]$
2. $r_y = (drand48() * 2) - 1$;
3. apply object transformation



Sphere: Done differently



$\bar{c}(\alpha, \beta) = [\cos\alpha \sin\beta, \sin\alpha \sin\beta, \cos\beta]$

$\alpha \in [0, 2\pi]$, $\beta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$

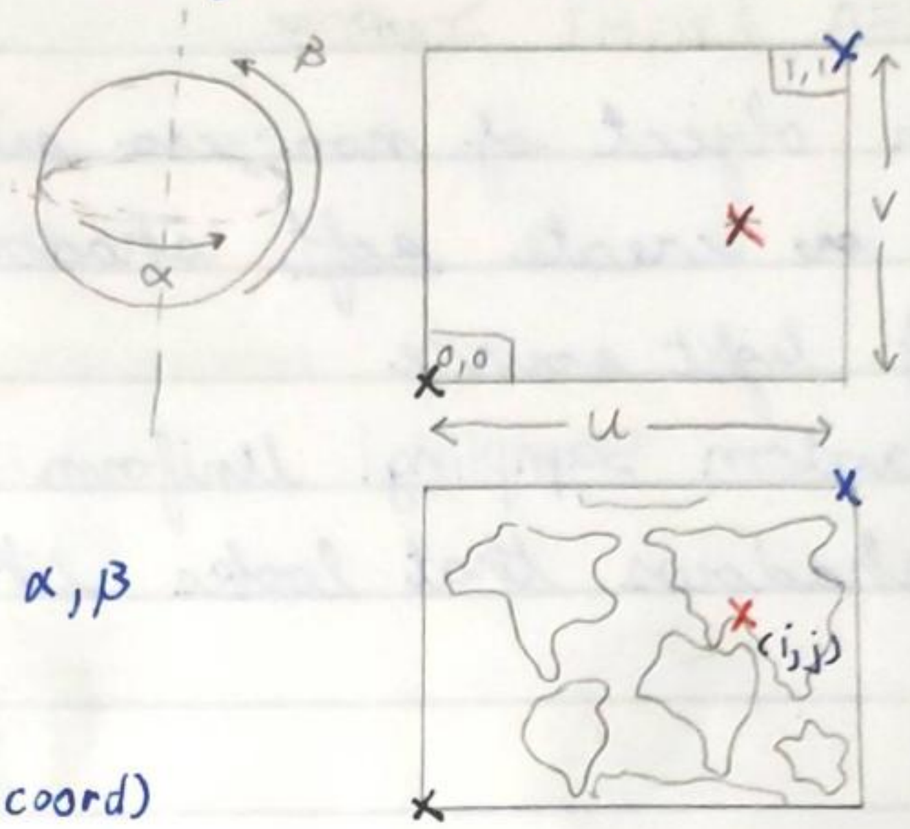
$(0, \pi)$?

Cylinder: can be done similar to planes

TEXTURE MAPPING

Take an image an paste it on surface to change objects colour.

1. Find an image
2. Mapping
3. Texture Coordinate space



1. given point $\bar{p}$ on object, find α, β for parametric curve
2. convert α, β to u, v (texture coord)
3. find pixel in image ~~color~~ corresponding to texture coord

$$\bar{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{aligned} x &= \cos\alpha \sin\beta \\ y &= \sin\alpha \sin\beta \\ z &= \cos\beta \end{aligned}$$

Solve for α, β.

$$u = \frac{\alpha}{2\pi} \qquad v = \frac{\cos\beta + 1}{2}$$

$$\Rightarrow \begin{aligned} i &= (image\_sx - 1) \cdot u \\ j &= (image\_sy - 1) \cdot v \end{aligned}$$

* For spheres, image "squished" near poles, Stretched near equator i.e. transformed spheres texture can look funky

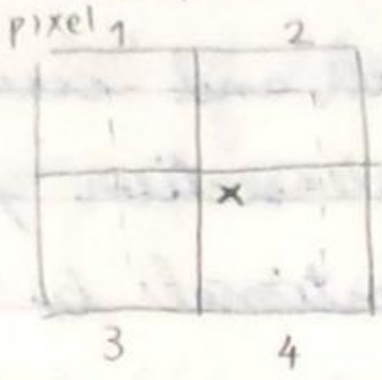
Mapping can be applied to:

- colour,
- transparency
- specular reflectivity
- r_d, r_a, r_g

} changing values in Phong model.

Texture coordinates $\in \mathbb{R}$, so which pixel to pick?

Bi-linear interpolation



take weight average colour of nearby pixels

NORMAL MAPPING

create an image that contains normal direction at each point

so instead of  , changes normals to 

BUMP MAP

shifting a surface point by some amount used on polygon meshes.

For a pixel you are rendering, makes it look at a slightly different location

for normal map, the outline of object is smooth

for bump map, the outline of object is bumpy.

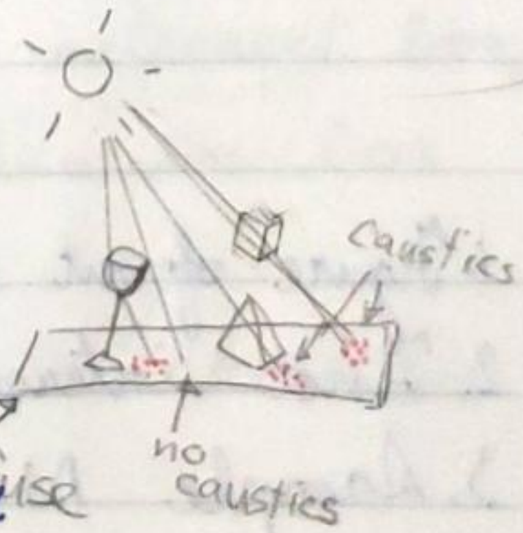
PHOTON MAPPING

Replicate patterns of "caustics" light focusing by the action of refractive / reflective objects

1. FORWARD PASS

Send N rays from light source towards scene

if it hits a diffuse object (not transparent) after bouncing off some transparent objects, accumulate 1 photon



- store in k-d tree or photo / radiance map

2. RENDERING PASS

Evaluate Phong model + look and count photons within a small distance d from intersection point
+ add brightness & proportional to # of photons found

TUT 06

RAY MARCHING

* Used to find intersection between object that are defined by complicated surfaces (using implicit equation)

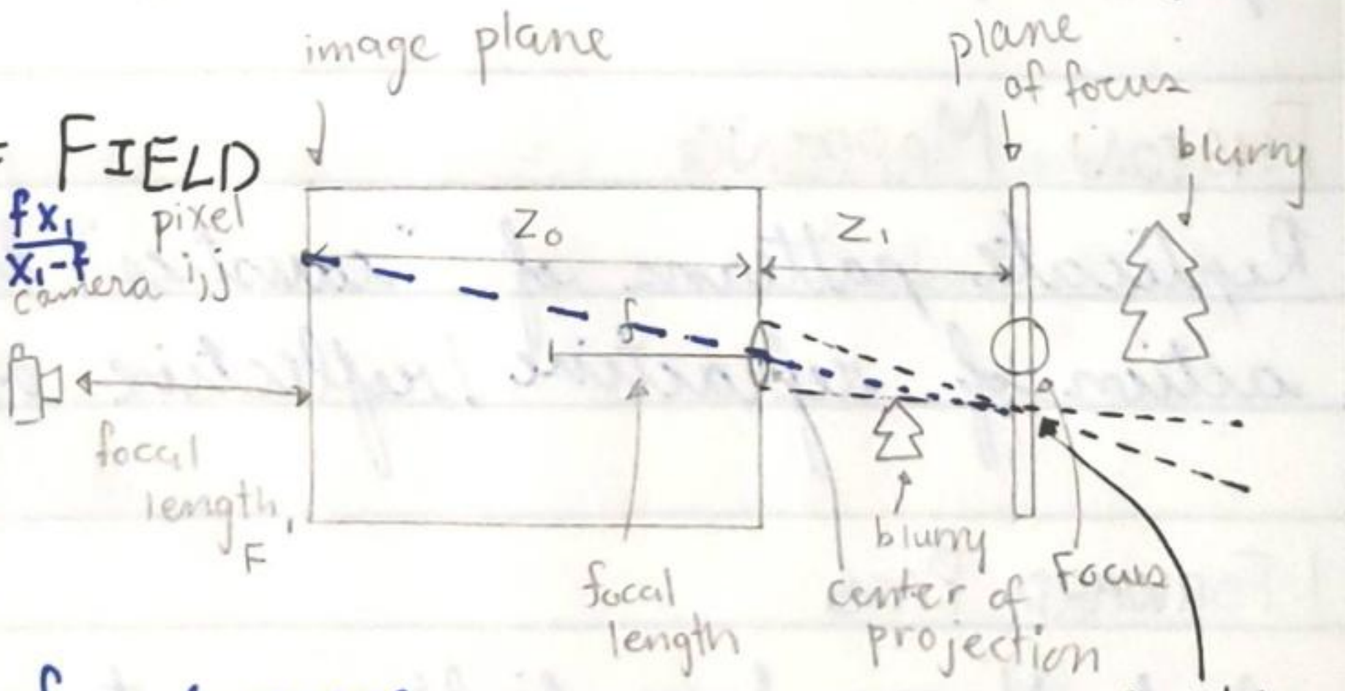
1. Given $\bar{r} = \bar{p} + \lambda\vec{d}$, start by a ~~small~~ & $\lambda = 0$.
2. Increment λ by small amount and evaluate $\bar{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$
3. If $f(x,y,z) > 0$ then not intersect object yet, repeat 2.
   If $f(x,y,z) \leq 0$ then intersected object, evaluate Phong model.

LEC 08

IMPLEMENTING DEPTH OF FIELD

$\frac{1}{z_0} + \frac{1}{z_1} = \frac{1}{f}$ $\Rightarrow x_0 = \frac{fx_1}{x_1 - f}$

- z_0: either from build scene or parameters from command line
- z_1: either from build scene or parameters from command line
- f: fixed

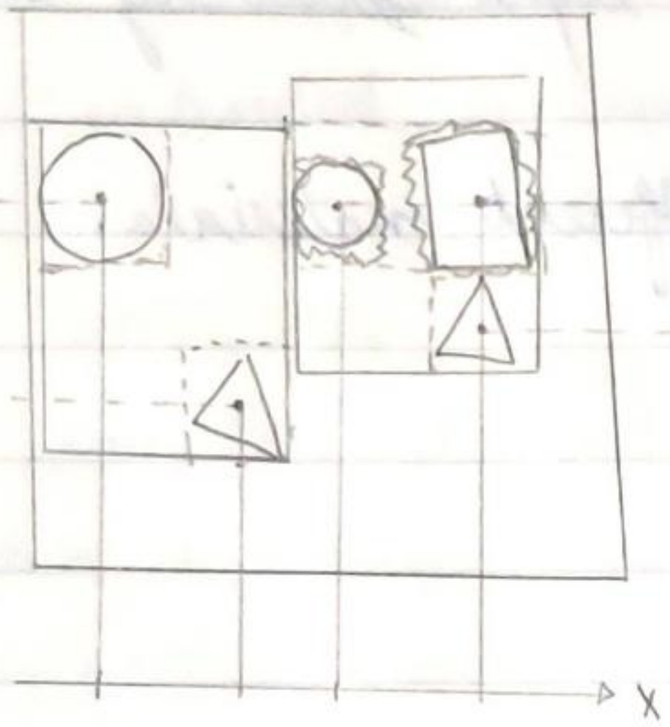


1. Figure of out z_0, z_1 given f and scene
2. Find intersection of $\bar{r}_{ij}(\lambda)$ and plane at z_1 ($\bar{p}_1$)
3. Sample N rays from aperture (given radius) to $\bar{p}_1$ and average colour returned

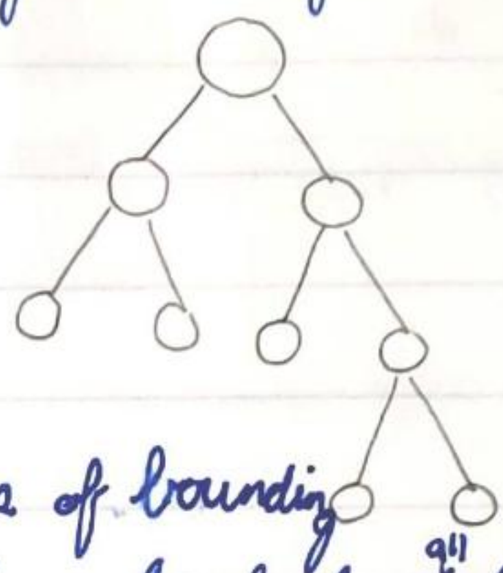
Ray Tracer Acceleration

Currently takes a lot of time to test intersection (for loop)
Use space partitioning - tree with a hierarchy of boxes of decreasing size

Using BVH (Bounded Volume Hierarchy)

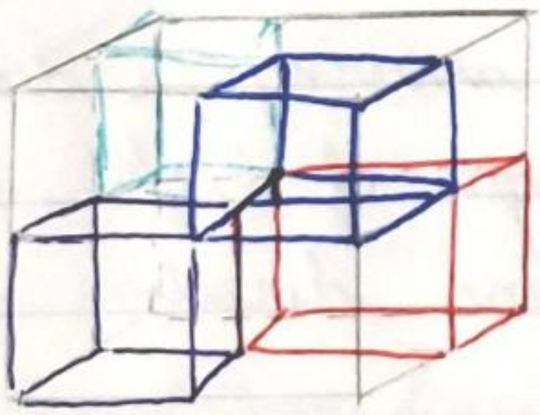


1. choose an axis to split along
   [by the largest x, y, z difference]
2. choose split point
   [by average chosen x, y or z]



Intersection: check against 6 sides of bounding boxes, remember to check for children!
if ray intersects a box, check intersection for smaller box

Using Octree



Split a space into 8 smaller cubes recursively.
If ray does not intersect box, object that is in the box will never intersed ray

Note: object in multiple boxes complicated to check for objects

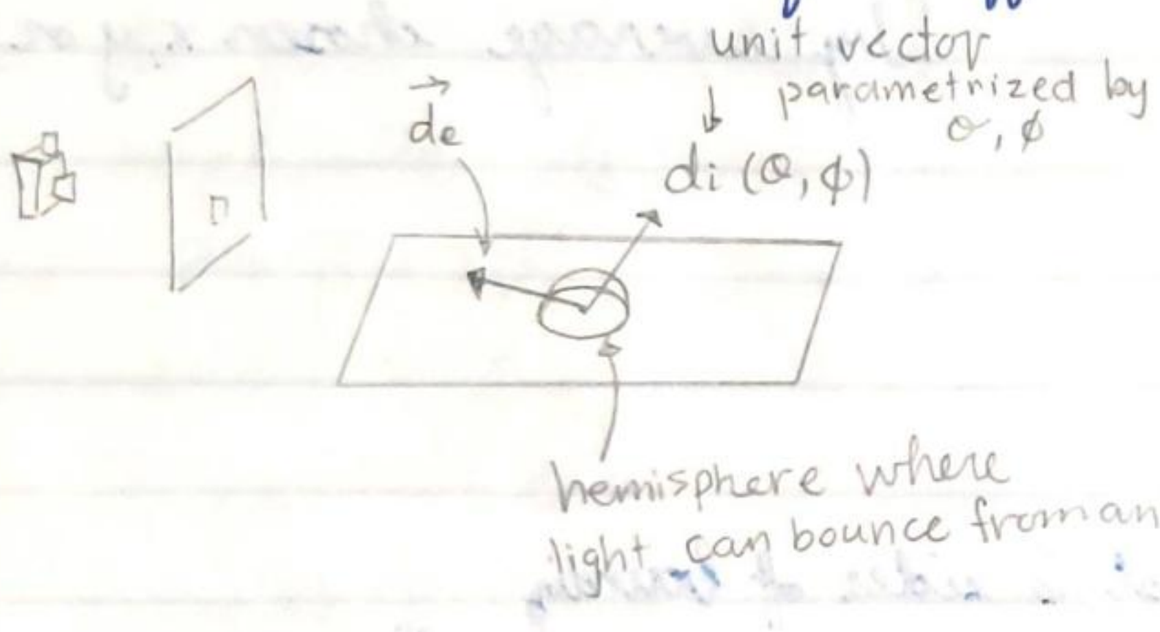
LEC 09

SIMULATE LIGHT TRANSPORT

BRDF - bidirectional reflectance distribution function

$f(\vec{d_e}, \vec{d_i}(\theta, \phi))$ which gives how much light is reflected in direction $\vec{d_e}$ given light arriving in direction $d_i(\theta, \phi)$

Values found in a TABLE for different materials.



RENDERING EQUATION

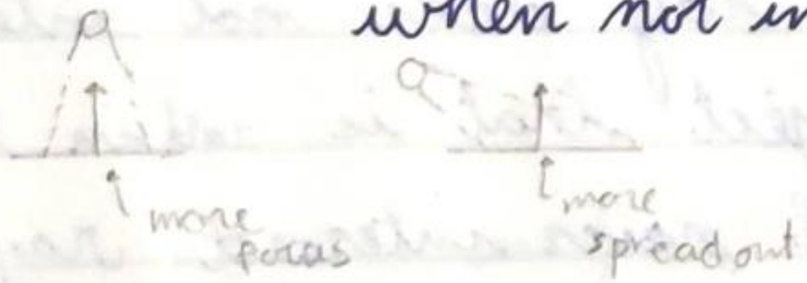
$$L(\vec{d_e}) = \int_{\Omega} f(\vec{d_e}, \vec{d_i}(\theta, \phi)) \, L(\vec{d_i}(\theta, \phi)) \cdot (\vec{n} \cdot d_i(\theta, \phi)) \, d\omega$$

- $L(\vec{d_e})$: accumulate for all angles incoming
- $\int_{\Omega}$: hemisphere centered at $\vec{p}$
- λ: to account for wavelengths of light

$f(\vec{d_e}, \vec{d_i}(\theta, \phi))$ - BRDF

$L(\vec{d_i}(\theta, \phi))$ - intensity of light arriving in direction $\vec{d_i}(\theta, \phi)$

$\vec{n} \cdot d_i(\theta, \phi)$ - account for strength of light different when not in same direction as $\vec{n}$



APPROXIMATING RENDERING EQUATION

cannot integrate! Too computation heavy! Distribution Ray Tracing (DRT)

Suggestion: sample N points on hemisphere. N recursive rays, evaluate rendering equation and accumulate brightness for each N rays

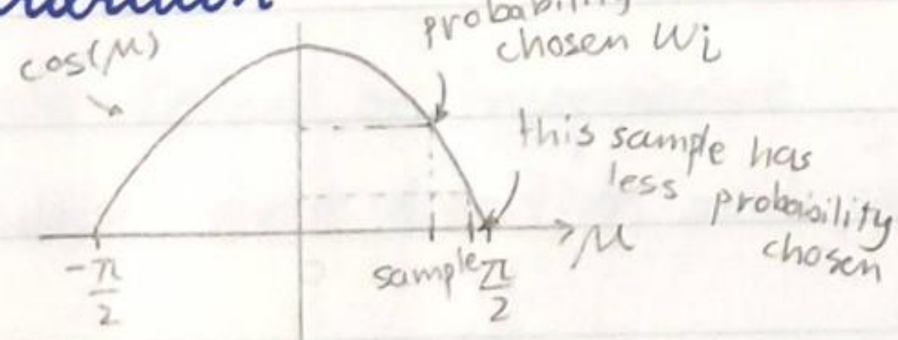
$O(N^{d_{depth}})$ - expensive but accounts for
global illumination
colour bleed
area light source
caustics
complex materials
- extended to transparent and semi transparent object

RANDOM SAMPLING

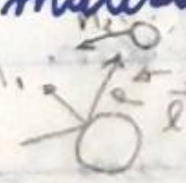
Sample complete light paths:
if within recursion depth hit light source, keep
if bounce too far away, ignore
↳ $O(dN)$ - much better! but image look noisy
- bounce chosen based on BRDF

IMPORTANCE SAMPLING

If $\vec{d_i}$ is almost perpendicular to $\vec{n}$, contribute little color
⇒ focus more samples on directions that give more contribution



when accumulating brightness by Rendering Equation.
$+ \frac{\text{sample } i}{w_i}$ (to remove bias for picking some directions more when material reflects in all directions)



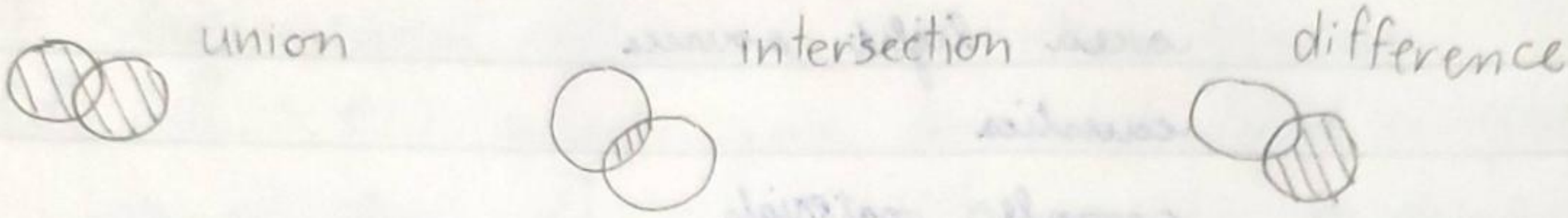
$$\frac{(\vec{n_1}\cdot\vec{l})\,(ALS)\,(\vec{n_2}\cdot-\vec{l})}{\lambda^2}$$
probability of picking source

DIRECT LIGHT SAMPLING

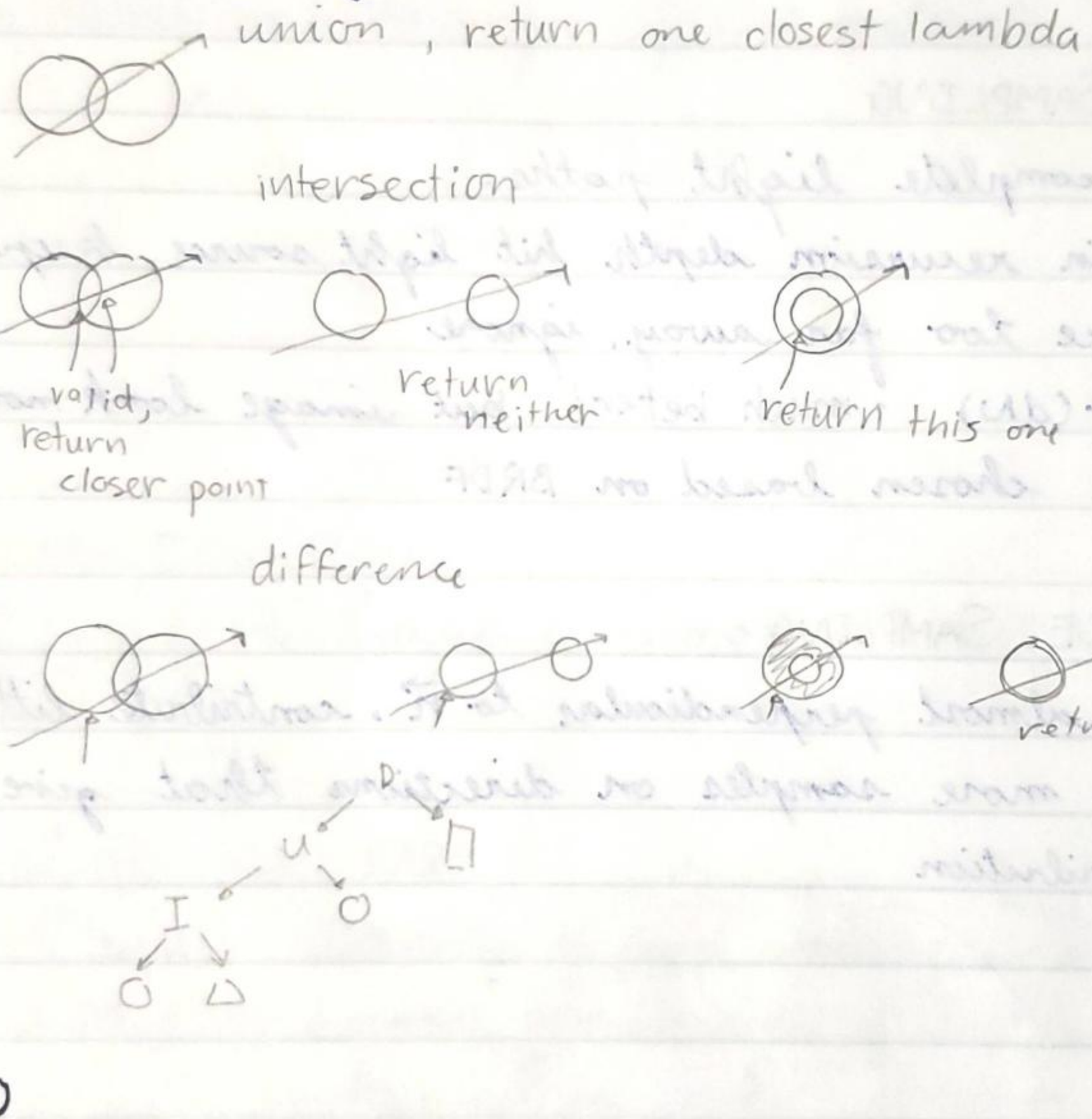
get direct contribution immediately if not blocked by light (adjust the weight $\frac{\text{sample } i}{w_i}$ for explicit sample)
as this choice is not random! ($\propto$ dist to LS, $\propto$ area of a hemisphere, $\propto$ area of LS)

TUT 8

CONSTRUCTIVE SOLID GEOMETRY CSGs



Intersection Testing

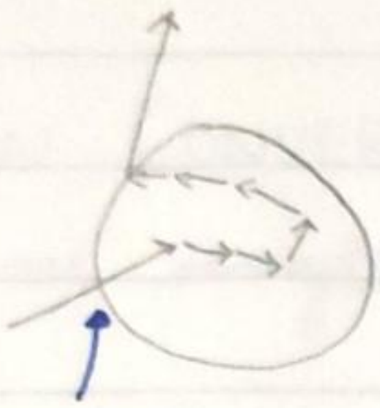


LEC 10

BIDIRECTIONAL PATH TRACING

For ~~tons~~ path of length k, $\frac{k}{2}$ bounces from a light source, then $\frac{k}{2}$ bounces from camera.
If there is a ray that can connect ~~both~~ both ends, a path is completed.

SUBSURFACE SCATTERING



Volumetric object with some density (constant, function, table)

Same as FindFirstHit, keep lambda that will leave object

Then ray march → roll dice → ray march again (based on density)

↘ random direction, random texture,
calculate lambda that will leave object,
accumulate brightness, depth++
ray march again

When leave object, continue bouncing

SIMPLE BRDF object has % diffuse + % reflect + % refract = 1

1. roll dice
2. decide diffuse / reflect / ~~to~~ transmissive
3. if diffuse / reflect: calculate next direction and path trace
4. else:

    calculate fresnel coefficient

    roll dice

    decide reflect or transmissive

    calculate next direction and path trace

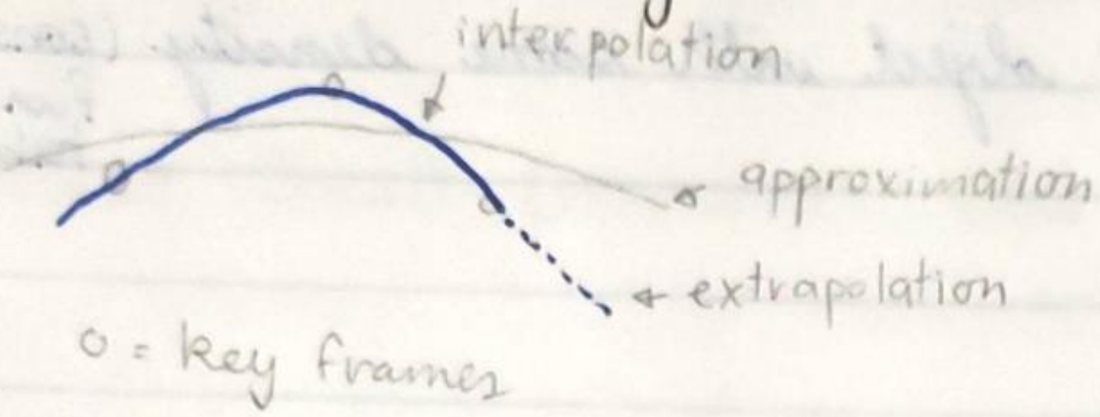
ANIMATION

Motion → object transformations

↘ camera locations, focal length, point of focus

Appearance → colour / texture, material properties

INTERPOLATION (for keyframe based animation)



2D CUBIC INTERPOLATION

given 4 points, find $x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$

$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3$ for curve

$\bar{c}(t) = [x(t) \;\; y(t)]$

$$\Rightarrow x(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \vec{t}^T \vec{a}$$

$$\bar{c}(t) = \vec{t}^T \cdot \begin{bmatrix} \vec{a} & \vec{b} \\ | & | \end{bmatrix}$$

4 points: $\bar{c}_1 = [x_1 \;\; y_1] = [x(0) \;\; y(0)]$

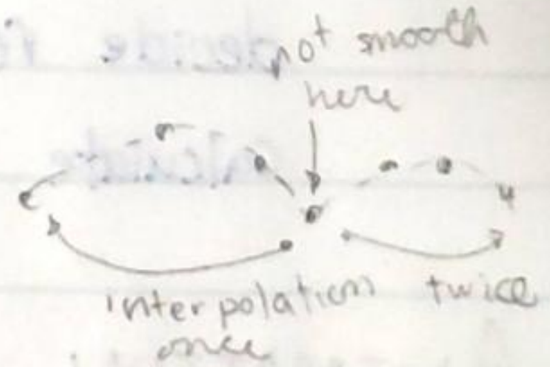
$\bar{c}_2 = [x_2 \;\; y_2] = [x(\frac{1}{3}) \;\; y(\frac{1}{3})]$ ← between 0 and 1

$\bar{c}_3 = [x_3 \;\; y_3] = [x(\frac{2}{3}) \;\; y(\frac{2}{3})]$

$\bar{c}_4 = [x_4 \;\; y_4] = [x(1) \;\; y(1)]$

solve ↓

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \frac{1}{3} & (\frac{1}{3})^2 & (\frac{1}{3})^3 \\ 1 & (\frac{2}{3}) & (\frac{2}{3})^2 & (\frac{2}{3})^3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$



2 points + 2 directions

$\bar{c}_1 = [x(0) \;\; y(0)]$ } $\vec{t} = [1 \;\; t \;\; t^2 \;\; t^3]$

$\bar{c}_2 = [x(1) \;\; y(1)]$ }

$\bar{c}_1' = [x'(0) \;\; y'(0)]$ } $\vec{t}' = [0 \;\; 1 \;\; 2t \;\; 3t^2]$

$\bar{c}_2' = [x'(1) \;\; y'(1)]$ }

$$\begin{bmatrix} x_0 & y_0 \\ x_1 & y_1 \\ x_0' & y_0' \\ x_1' & y_1' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$