

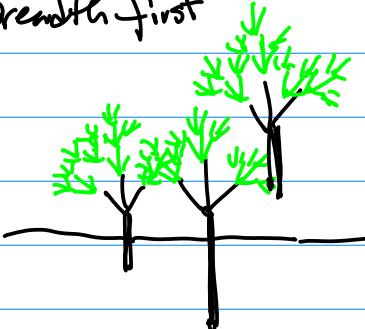
- a A set of Squares Rotation
with One Same Color
b Change location/Color/size

C

Keep the patterns, please
don't remove them!

Part II Once Part I is Done
Switch to Part I.

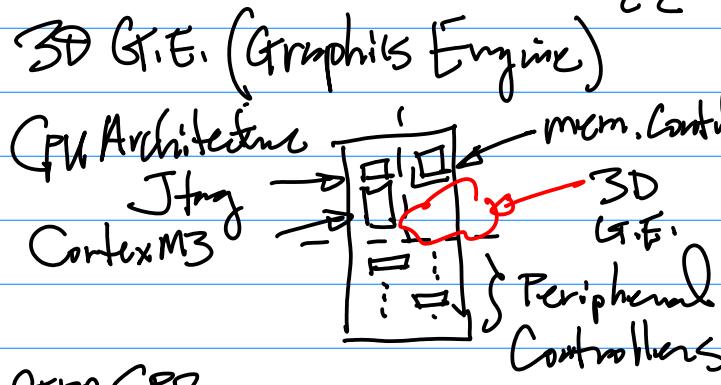
a Breadth first



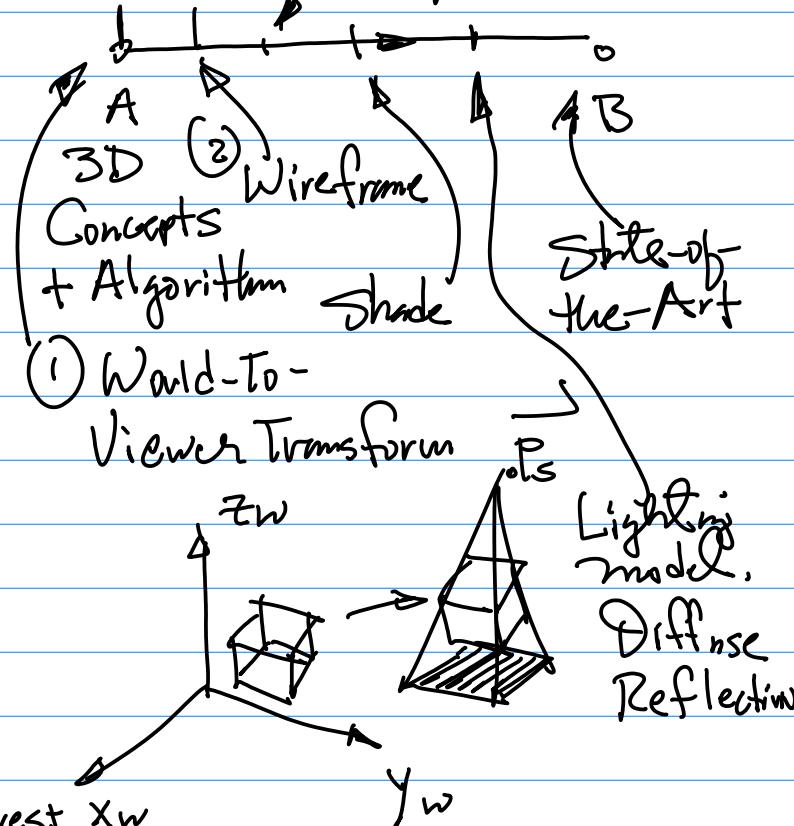
b L>> C Create A patch of Forest
Changing Location/Size

Note: Please Don't freeze the
Drawing Till All the trees
are constructed.

Note: Report Writing.



{ GPP SPRs,
SPI SPRs. Solid Obj.
ITL, HS, Removal



March 15 (Monday)

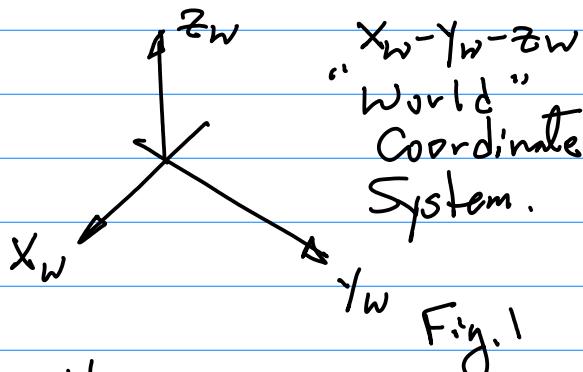
Note: 1st midterm A week from
this Wednesday (March 24th)

2nd Smartphone (Cam Capability)
A piece of Paper to the
Class, Test the Environment

Ref: github : 2018F-114-
3D Graphics

Virtual Camera Location

1°



Right-Hand System,

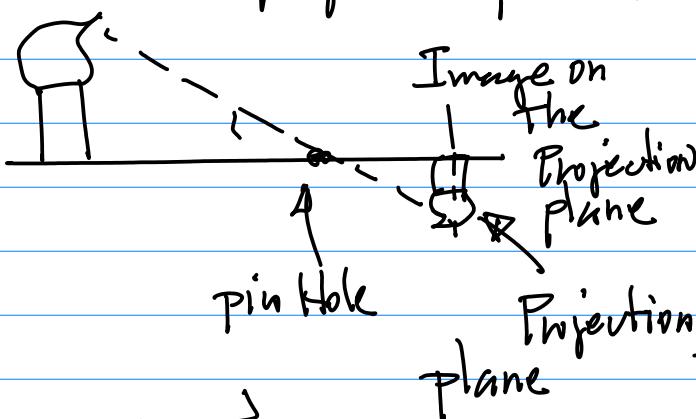
2° Virtual Camera

a Virtual Enclosure

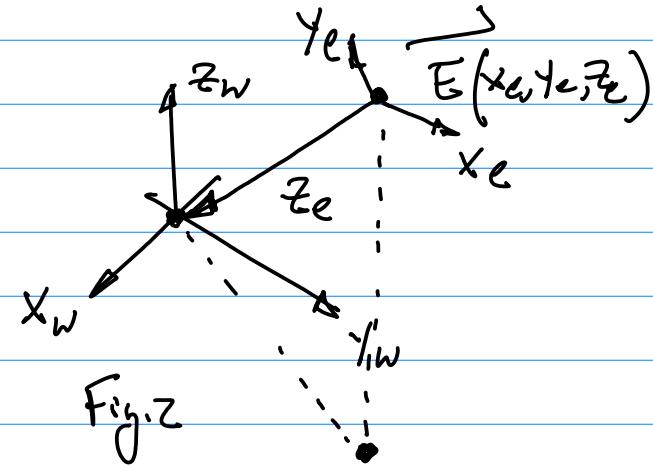
b Optic Lens (Pin Hole)

$\phi_d \approx b$ Very Small diameter

c Projection plane, the plane to form projected Image when light passing through the lens and reaches the projection plane.



3. Denote $E(x_e, y_e, z_e)$ as



Projection of $E(x_e, y_e, z_e)$
On to $X_w - Y_w$ Plane in the
 $X_w - Y_w - Z_w$ World Coordinate System

4. Viewer Coordinate System

Viewer ~ Virtual Camera

$x_e - y_e - z_e$ Viewer Coordinate System. "Eye"

a $E(x_e, y_e, z_e)$ As the origin.

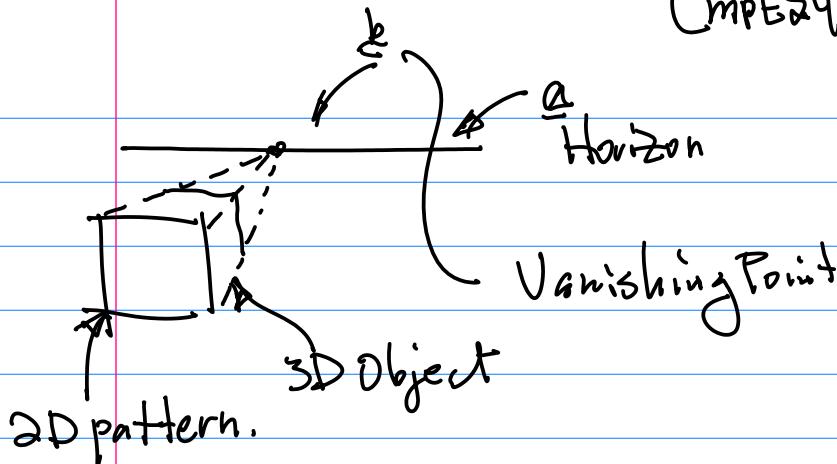
b Left-Hand System.

c z_e -axis points to the origin.

5. Perspective Projection

Complex

24



3D Object formed on 2D plane

7. Transformation Pipeline

To Display 3D object(s) on to 2D LCD Display Screen,

W2V (World-2-Viewer)
Transform
Perspective Projection.

6. Perspective Projection \rightarrow @ the Virtual Camera in X_w - Y_w - Z_w , in X_e - Y_e - Z_e .

Objects in X_w - Y_w - $Z_w \rightarrow$ Viewer Coordinate System X_e - Y_e - Z_e

Projection Plane
form Perspective
Projection

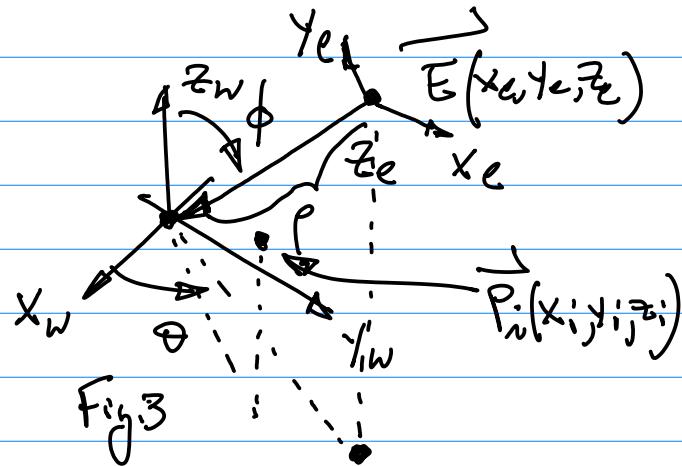


Fig.3

3 parameters:

\hat{a} theta θ Angle

\hat{b} Vector \vec{E} to \vec{D} ,

"roh", Distance

Mathematical Formulation

1. Formula: Transformation $\stackrel{\text{World-to-Viewer}}{=} \text{Pipeline}$ $\stackrel{\text{Transform}}{=}$

$$\sqrt{x_e^2 + y_e^2 + z_e^2}$$

$\stackrel{\text{Z° Foundation}}{=}$

$\stackrel{\text{a Translation}}{=}$

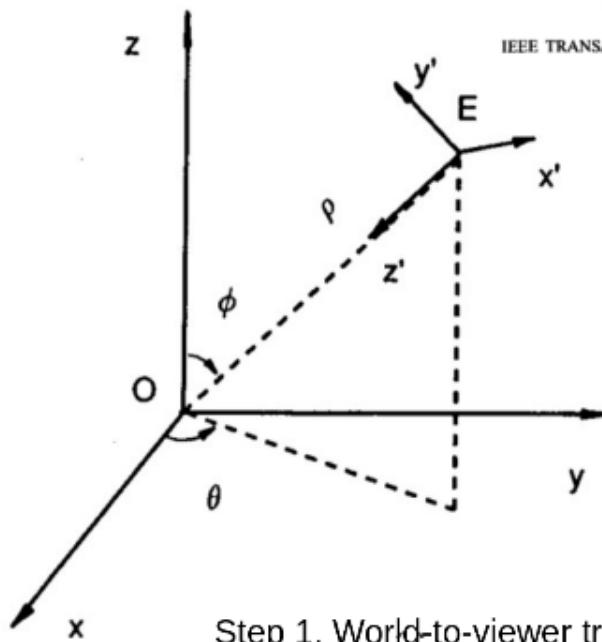
$\stackrel{\text{b Perspective}}{=}$ $\stackrel{\text{Projection}}{=}$ $\stackrel{\text{Angle } \theta_w \text{ and } \theta_e}{\dots (1)}$

$\stackrel{\text{Rotations}}{=}$ $\stackrel{\text{.. wrt } x_w \text{ And}}{\text{.. wrt } y_w}$ $\stackrel{\text{.. wrt Arbitrary Axis}}{\text{.. wrt } z_w}$ $\stackrel{\text{World-Z-Viewer}}{=}$

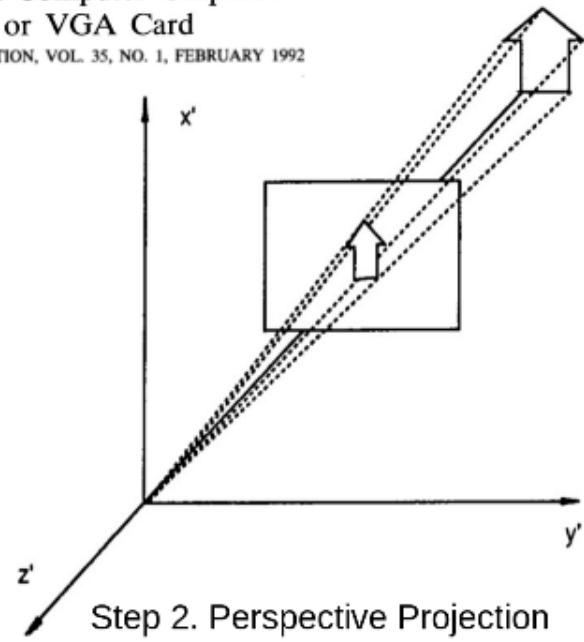
3D Transformation Pipeline Technique

Reference: H. Li Three-Dimensional Computer Graphics
Using EGA or VGA Card

IEEE TRANSACTIONS ON EDUCATION, VOL. 35, NO. 1, FEBRUARY 1992



Step 1. World-to-viewer transform



Step 2. Perspective Projection

$$\mathbf{T} = \begin{bmatrix} -\sin \theta & \cos \theta & 0 & 0 \\ -\cos \phi \cos \theta & -\cos \phi \sin \theta & \sin \phi & 0 \\ -\sin \phi \cos \theta & -\sin \phi \sin \theta & -\cos \phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x_p = x_e \left(\frac{D}{z_e} \right)$$

$$y_p = y_e \left(\frac{D}{z_e} \right)$$

Harry Li, Ph.D.

Transform, Map P_i from the World-coordinate to Viewer Coordinate.

$$\begin{pmatrix} x'_i \\ y'_i \\ z'_i \end{pmatrix} = T \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

After
 $x_w - y_w - z_w$

Before
 $x_w - y_w - z_w$

World to Viewer Transform.

March 17 (wed)

Topics : 1° Hardware Architecture 2° Software SPRs Init & Config.

Example: 2D & 3D F.T.
SPI I/F LCD Display to work with LPC1769.

GPIO(GPP) SPI
A set of b_n'

Per. Cont.

- System Configuration
 - PWR
 - CLK
- * Configuration of the Peripheral Cont. Multi-plexing

$$\begin{pmatrix} \sin\theta & \cos\theta & 0 & 0 \\ \cos\theta & \sin\theta & " & " \\ 0 & 0 & " & " \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Part I.
"θ"

Conjunto to Rotation Matrix

$$\begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ \cos\psi & \cos\phi \sin\psi & " & " \\ \sin\psi & \sin\phi \cos\psi & " & " \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Part II
"ψ"
"φ"

RTOS
Peripheral Controller

GPP
SPI

SPI's SPR.

1. Naming Convention

LPC_SC → PCONP

2. Power Up the Selected Peripheral Controller By Setting the Corresponding.

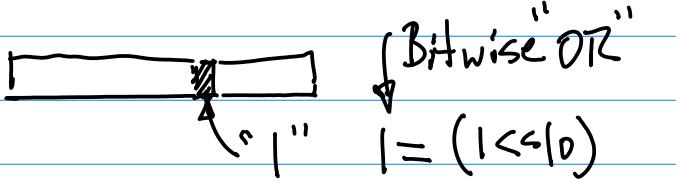
$$\begin{pmatrix} - & - & - & P \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Part III
" "

roh

CmpE240

26



Tech Spec.

1° 8 bit Transfer,

$$CR\phi[3:\phi] = 0111 = 0x7$$

$$\& = \sim(3 \ll 20)$$

"AND"
"11"
"Negation", "0D"

2° SPI

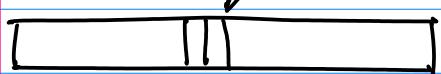
$$CR\phi[5:4] = 00 \text{ (SPI)}$$

3° Clock f_{SPI}

$$\cong CR\phi[15:8]$$

$$8 \text{ bits } 2^{25} = 2^8$$

Clear
2 Bits



Set 2 bits
as "01"

$$f = \frac{\text{PCLK}}{\text{SPI } (SCR+1) \text{ CPSDVSR}} \dots (1)$$

$$[0, 2^{25}]$$

Much harder.

Today's Topics :

1° Midterm Review

2° SPRs CR ϕ , CR λ
for SPI I/F

Ref:

1° CPU Datasheet.

PP431-433.

Note: $\begin{cases} CR\phi \\ CR\lambda \end{cases}$ LPC_SSP1 \rightarrow CR ϕ
Control Register

Table 57 |



CR ϕ

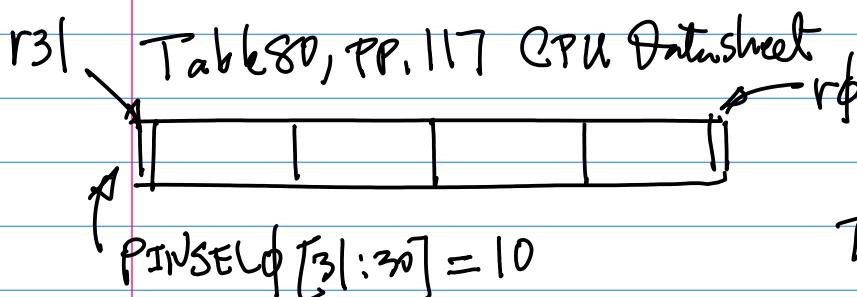
2° Sample code

SPI_init(Draw a
Line)

Example: SSPC Source Code

Walk-Through: 15/-208

Line 162-165 PINSEL0



$$f_{\text{SPI}} = \frac{1 \times 10^6}{(7+1) * \text{DVR}}$$

To find f_{SPI} .

$$2^8 = 256, \text{ SCR} \in [0, 255]$$

PP. 433, CPSDVSR < [2, 254]

Midterm Review.

1° Video On, Mandatory.

a Submission to CANVAS
15 min. File Uploading

No Late Submission

After the Deadline

Paper will be disqualifed

b If CANVAS Disrupted,

then E-mail Submission

= file in "zip"

Line 173

$\text{CR} = 0x0707 \rightarrow$ Tech

(\rightarrow Spec.)

... :0000;0111 :0000;0111

fb upper
Bits all
zeros.

$\text{CR}[5:8]$ SCR $\text{CR}[5:4] = 00$ SPI
 Clock

$$f_{\text{SPI}} = \frac{\text{PCLK}}{(\text{SCR}+1) * \text{DVR}}$$

$\text{CR}[5:8] = 0 \times 7$

FirstName + 4 Digits + CmPE240
SID mid.zip

2° 3 Questions ±

Hardware { CPU Block Diagram
Memory Map

SPRs.

CIC, SCH Design

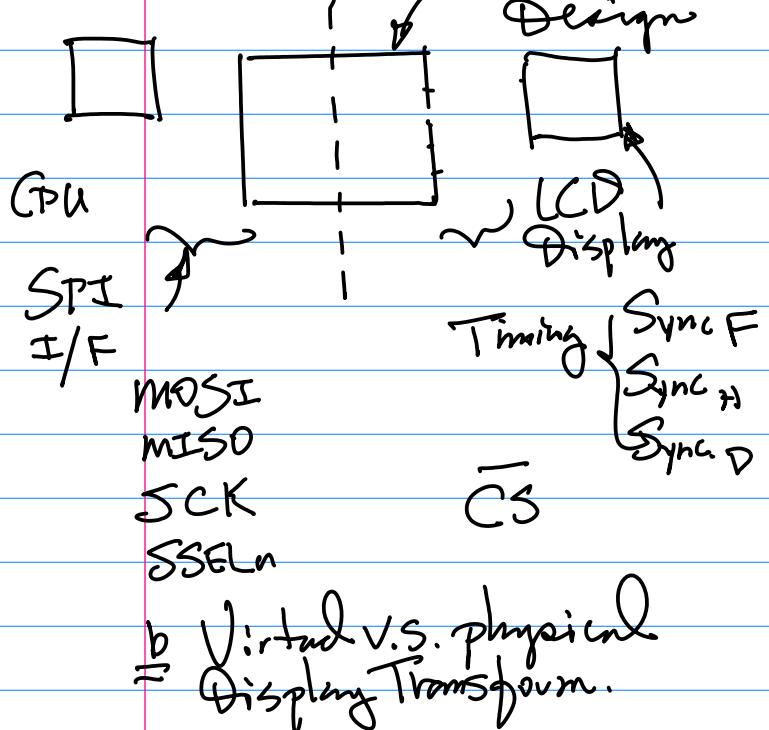
$\text{CR1}[2] = 0$, Software: Coding
for "Master"

CmpE240

28

SPR → Binary Pattern
 Coding → for Init & Config
 Debugging Purpose
 Algorithm:
 2D Vector Graphics.
 G.F.

\approx a LEC CPU LPC Driver.
 \approx LPC Driver
 Design



\approx b Virtual v.s. physical
 \approx Display Transform.

$$\text{C } \vec{P} = \vec{P}_i + \lambda (\vec{P}_{i+1} - \vec{P}_i)$$

Screen Saver. { i. Rotation!
 " " w/o Rotation
 " " matrix
 " " 2D Transforms

Composition of
 2D Transform.

$\begin{cases} R_{3 \times 3} \\ T_{3 \times 3} \end{cases} \rightarrow \text{Tree.}$

Preprocess + $R_{3 \times 3}$ + Post ~

Formula: One Page Formula Sheet; is allowed,
 However, No Example, or Verbal
 Explanation is Not Allowed,
 Submission of the formula
 Page is required with your
 mid-term paper.
 No multiple choice question.

SCH: Requires All the pins needed
 in the design to have Label;
 wire: "Arrow" to indicate direction.

Block Diagram: wire(s), Label(s)
 direction(Arrow)

CPU Datasheet will be provided

C Code program will be provided for
 Answering questions, or
 for Redesign.

Calculator is allowed;

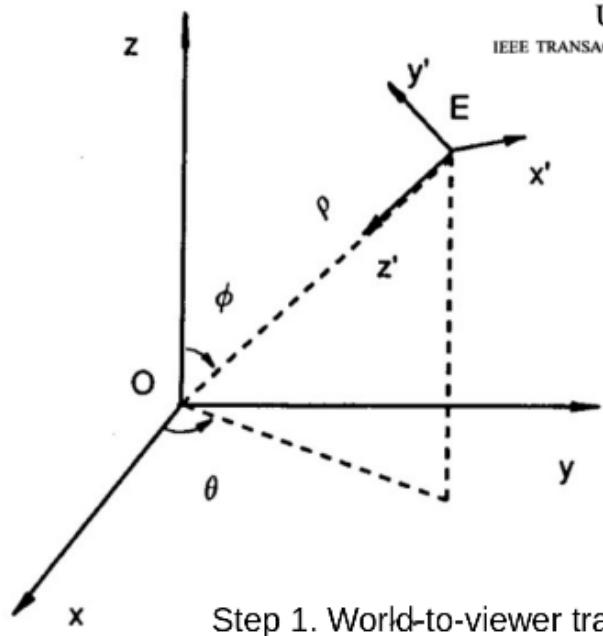
April 5 (Monday)

1. Midterm Key on github, "Key" To search.

3D Transformation Pipeline Technique

Reference: H. Li Three-Dimensional Computer Graphics
Using EGA or VGA Card

IEEE TRANSACTIONS ON EDUCATION, VOL. 35, NO. 1, FEBRUARY 1992



Step 1. World-to-viewer transform

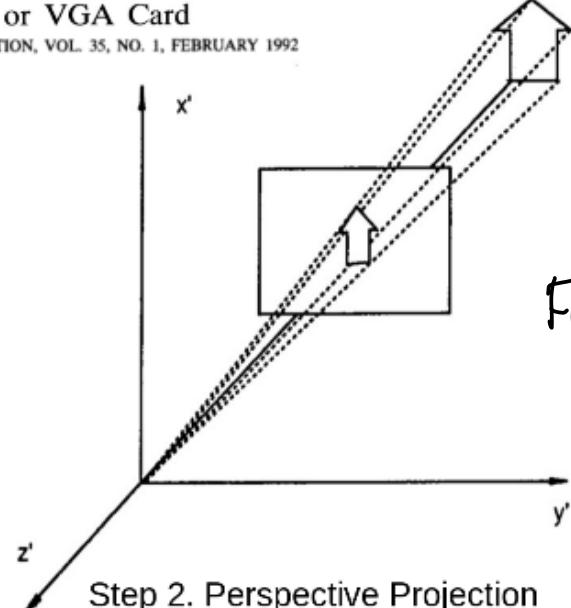


Fig1.

$$\mathbf{T} = \begin{bmatrix} -\sin \theta & \cos \theta & 0 & 0 \\ -\cos \phi \cos \theta & -\cos \phi \sin \theta & \sin \phi & 0 \\ -\sin \phi \cos \theta & -\sin \phi \sin \theta & -\cos \phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

... (1)

$$x_p = x_e \left(\frac{D}{z_e} \right)$$

$$y_p = y_e \left(\frac{D}{z_e} \right)$$

... (2)

... (2)

Harry Li, Ph.D. mem!

Today's Topics: 3D G.E.

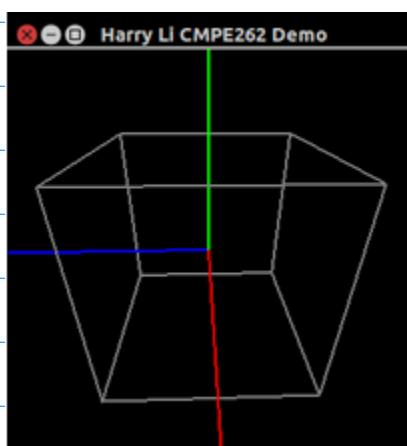


Fig2a

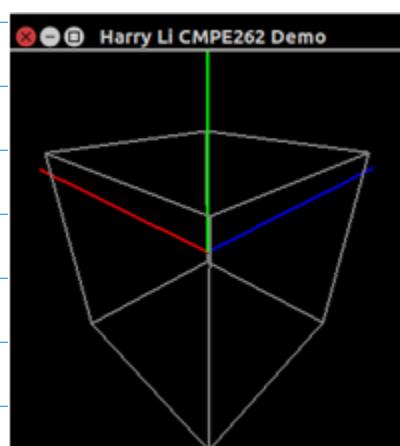
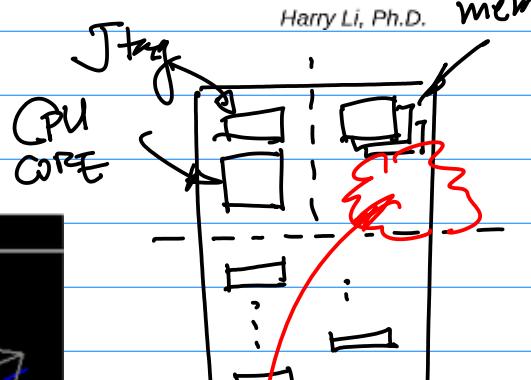


Fig2b



2D G.E.
3D G.E. → GPU
Peripheral Controllers

from github. Kang "111a,"...

Note: 2D G.F. { Vector Graphics
Transformation
Primitive Graphics }

{ $\stackrel{a}{=} \text{D.D.A.}$
 $\stackrel{b}{=} \text{line}$
 $\stackrel{c}{=} \text{Arc/circle etc.}$ } Transistor/Gate Level.

Example: 3D wireframe model

First, $X_w-Y_w-Z_w$ World Coordinate System.

$\stackrel{a}{=}$ Right System, r-g-b for X_w, Y_w, Z_w axis
 $\stackrel{b}{=}$ Transformation Pipeline

{ $\stackrel{1st}{=} \text{World-Z-Viewer}$
 $\stackrel{2nd}{=} \text{Perspective Projection}$

$$T_{4 \times 4} : (x_w, y_w, z_w) \rightarrow (x_v, y_v, z_v) \quad \dots (3)$$

Second. Design of Dataset e.g. ... (3b)

4 Vertices for $X_w-Y_w-Z_w$ Axis

$\vec{P}(x, y, z)$ 3D pt. in $X_w-Y_w-Z_w$

Step 1. $\vec{P}(x, y, z) \in \mathbb{R}^3$

for the Origin $\vec{P}_o(x_o, y_o, z_o) = (0, 0, 0)$... (4)

$\vec{P}_x(x_x, y_x, z_x) = (100, 0, 0)$, ... (4-1)

$\vec{P}_y(x_y, y_y, z_y) = (0, 100, 0)$, and ... (4-2)

$\vec{P}_z(x_z, y_z, z_z) = (0, 0, 100)$... (4-3)

Define X_w -axis

Define Z_w -axis

Now, Implementation (Drawing r-g-b axis) \rightarrow AT&T 121m

Homework: Draw r-g-b axis on your CPC Display.
Bring your program Board to the Next Class.

Step 2. World-Z-Viewer

$$T_{4 \times 4} : (x_w, y_w, z_w) \in \mathbb{R}^3 \rightarrow$$

$$(x_v, y_v, z_v) \in \mathbb{R}^3$$

$$\begin{pmatrix} x_v \\ y_v \\ z_v \\ 1 \end{pmatrix} = T_{4 \times 4} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \dots (5)$$

"After"

"Before"

Now, find the X_w -axis in Viewer Coordinate

Step 3. Perspective Projection

$$P: (x_e, y_e, z_e) \in \mathbb{R}^3 \xrightarrow{\text{Proj}} (\underline{x_i}, \underline{y_i}) \in \mathbb{R}^2$$

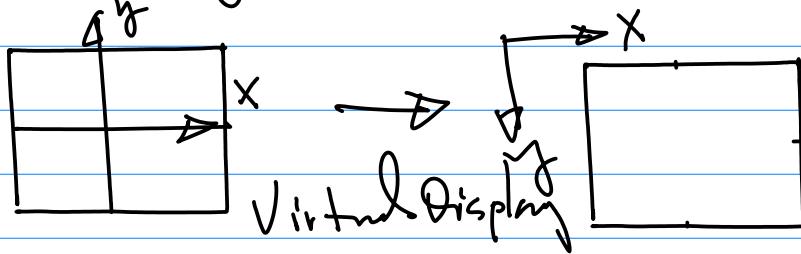
Coordinate on your
LPC1768 Display
Device.

Note:

$(\underline{x_i}, \underline{y_i})$ is defined on 2D

Virtual Display Coordinate

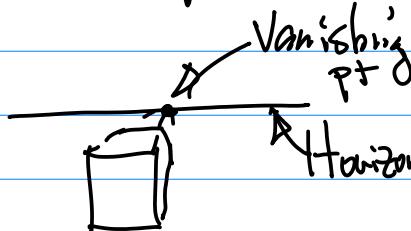
Need to perform Virtual to
Physical Transformation in order
to plot your result.



Back projection plane:
the only light that can
reach the projection plane
is the light passing
through the "pin hole".
(Plane is at the Back)
projection

Frontal projection plane:
move the back projection
plane to the outside of
the virtual camera.

D: Distance from the projection
plane to the "pin hole".



$$\begin{cases} x_p = \frac{D}{z_v} \cdot x_r \\ y_p = \frac{D}{z_v} \cdot y_r \end{cases} \quad \text{Virtual Cam} \quad \text{focal length}$$

(x_p, y_p)
On your 2D
Display

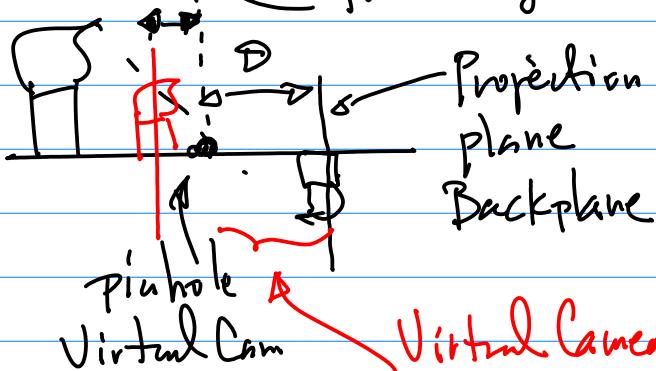
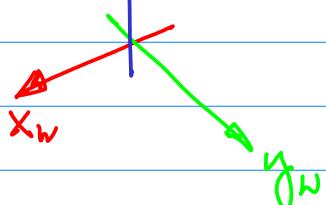


Fig1.

In Homework,
 $D=20$

$$D = 20$$

$$z_w$$



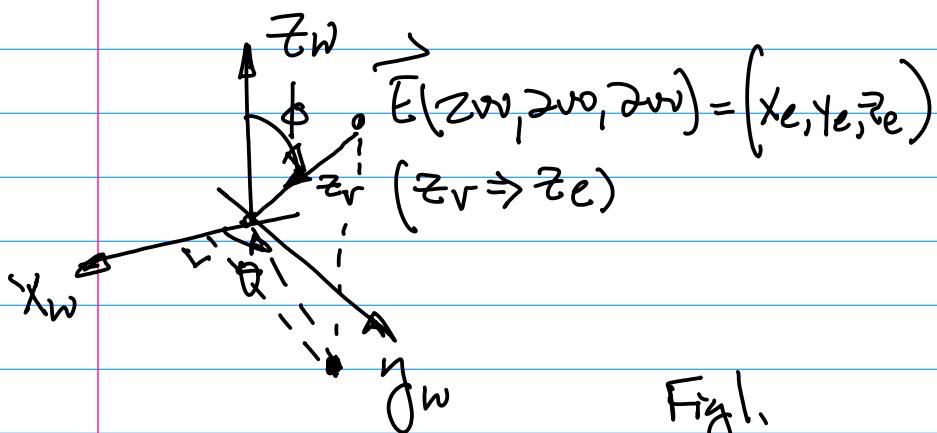


Fig. 1.

$$\sin \theta = \frac{a}{b}$$

a Vector Passing Through \vec{E} , Perpendicular to $X_w - Y_w$ plane, So form an intersection pt.

$$a = y_e = 200$$

$$b = \sqrt{x_e^2 + y_e^2} = 200\sqrt{2}$$

$$\therefore \sin \theta = \frac{200}{200\sqrt{2}} = \frac{\sqrt{2}}{2}$$

$$\cos \theta = \frac{x_e}{b} = \frac{200}{200\sqrt{2}} = \frac{\sqrt{2}}{2}$$

for ϕ (phi)

Note: Angle θ (Theta) on $X_w - Y_w$ plane
wrt Positive X_w -axis And
Counter Clockwise Direction

Note: Angle ϕ (phi) on $Z_v - Z_w$
plane,

Now, find each entry on 4×4 matrix, April 7 (Wed)

So World-2d View Transform Note: $X_v - Y_v - Z_v$ Left Hand System
Can be performed.

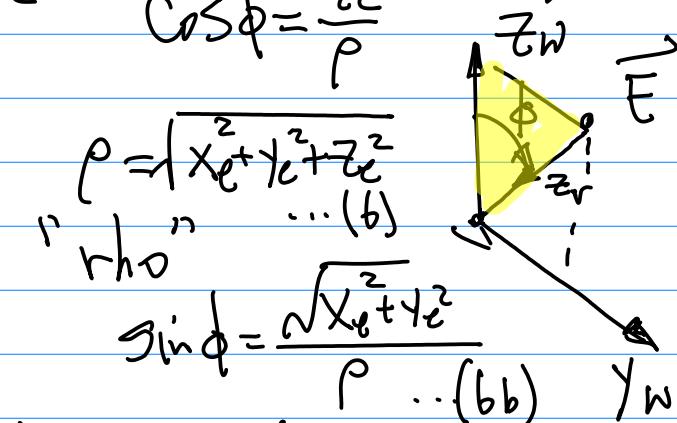
for Angle θ (Theta)

$\sin \theta$ and $\cos \theta$

$$\cos \phi = \frac{z_e}{\rho} \quad \dots (5b)$$

$$\rho = \sqrt{x_e^2 + y_e^2 + z_e^2} \quad \dots (6)$$

$$\sin \phi = \frac{\sqrt{x_e^2 + y_e^2}}{\rho} \quad \dots (6b)$$



$$\cos \phi = \frac{z_e}{\rho} = \frac{200}{\sqrt{200^2 + 200^2 + 200^2}}$$

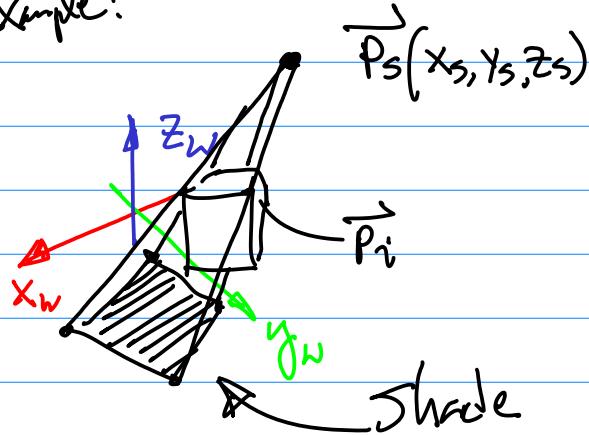
$$= \frac{200}{200\sqrt{3}} = \frac{\sqrt{3}}{3}$$

Similarly

$$\sin \phi = \sqrt{x_c^2 + y_c^2} / \rho = \frac{2\sqrt{2}}{2\sqrt{2}\sqrt{3}} = \sqrt{\frac{2}{3}}$$

Consider "Shade" Calculation.

Example:



1. $x_w - y_w - z_w$.

Assignment (Monday)

Note: 1^o Homework Submission (E-mail)

$x_w - y_w - z_w$ Drawing.

Source Code — Photo

(Exported Project) By Wednesday.

2^o Bring your Prototype

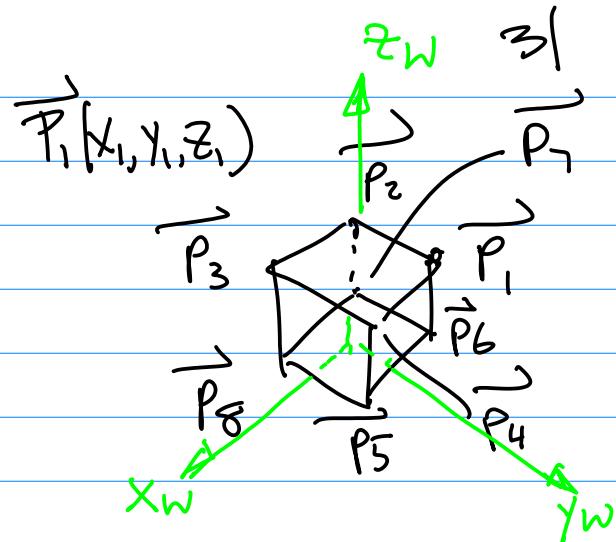
Board to each Session for
Inspection, Show & Tell.

Discussion on 3D Shade

Computation

Conditions for this discussion

1^o Define $\{\vec{P}_i(x_i, y_i, z_i) | i=1, 2, \dots, 8\}$



One Side
of the cube Overlapped
with z_w -axis
size of 100.

$\vec{P}_1(0, 100, 110)$
 $\vec{P}_2(0, 0, 110), \vec{P}_3(100, 0, 110)$
 $\vec{P}_4(100, 100, 110)$

Note: P_1, P_2, \dots , are
arranged Counter
Clockwise

$\vec{P}_5(100, 100, 10), \vec{P}_6(0, 100, 10)$

$\vec{P}_7(0, 0, 10), \vec{P}_8(100, 0, 10)$

2. Point Light Source

$\vec{P}_S(150, 100, 200)$

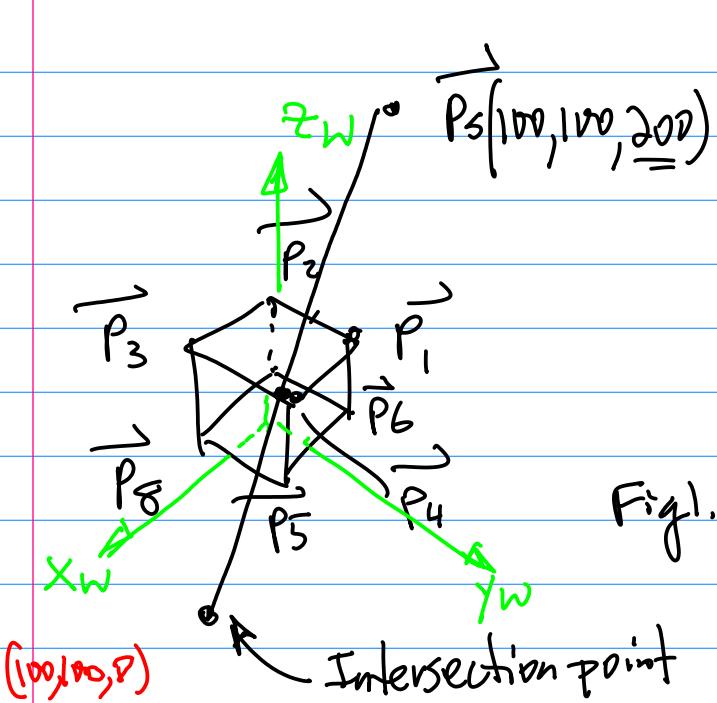


Fig.1.

3. Ray Equation, Connecting

\vec{P}_3 to \vec{P}_4 ,

$$\vec{R} = \vec{P}_3 + \lambda (\vec{P}_4 - \vec{P}_3) \dots (1)$$

$$\vec{P}_w = \vec{P}_4$$

OR,

$$\vec{P} = \vec{P}_3 + \lambda (\vec{P}_w - \vec{P}_3) \dots (1b)$$

Find intersection point on
Xw-Yw plane.

4. Plane Equation

$$\underline{\underline{a}} \text{ Normal Vector } \vec{n} (n_x, n_y, n_z)$$

Perpendicular to the given
plane, Xw-Yw plane

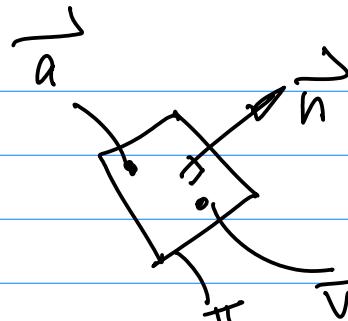


Fig.2

4b. plane π .

An Known, Arbitrary pt
On π denoted
 $\vec{a}(a_x, a_y, a_z)$

An arbitrary point

$$\vec{v}(v_x, v_y, v_z) \text{ on } \pi$$

$$\vec{n} \cdot (\vec{a} - \vec{v}) = 0 \dots (2)$$

OR

$$\vec{n} \cdot (\vec{v} - \vec{a}) = 0 \dots (2-a)$$

↑
Normal Vectn
↑
Line Segment

5. Intersection pt.

$$\vec{R} = \vec{P}_3 + \lambda (\vec{P}_w - \vec{P}_3) \dots (3a)$$

$$\vec{n} \cdot (\vec{v} - \vec{a}) = 0 \dots (3b)$$

From (3b),

$$\vec{n} \cdot (\vec{v} - \vec{a}) = 0$$

$$\vec{v} = \vec{r}, \vec{v} = \vec{p}_S + \lambda(\vec{p}_i - \vec{p}_S)$$

Sub. $\vec{v} = \vec{r}$ above into (3b)

$$\vec{n} \cdot (\vec{v} - \vec{a}) \Big|_{\vec{v} = \vec{R}} = 0$$

$$\vec{n} \cdot (\vec{R} - \vec{a}) \Big|_{\vec{R} = \vec{p}_S + \lambda(\vec{p}_i - \vec{p}_S)} = 0$$

$$\vec{n} \cdot (\vec{p}_S + \lambda(\vec{p}_i - \vec{p}_S) - \vec{a}) = 0$$

$$\vec{n} \cdot \vec{p}_S + \lambda \vec{n} \cdot (\vec{p}_i - \vec{p}_S) - \vec{n} \cdot \vec{a} = 0$$

$$\lambda \vec{n} \cdot (\vec{p}_i - \vec{p}_S) = \vec{n} \cdot \vec{a} - \vec{n} \cdot \vec{p}_S$$

$$\lambda = \frac{\vec{n} \cdot (\vec{a} - \vec{p}_S)}{\vec{n} \cdot (\vec{p}_i - \vec{p}_S)} \dots (4)$$

from Eqn (3a), Ray Egn.

With λ we can find the intersection

Point.

In C/C++ coding.

$$\lambda = \frac{\vec{n} \cdot (\vec{a} - \vec{p}_S)}{\vec{n} \cdot (\vec{p}_i - \vec{p}_S)} = \frac{(n_x, n_y, n_z) \cdot (a_x - x_s, a_y - y_s, a_z - z_s)}{(n_x, n_y, n_z) \cdot (p_{i_x} - p_{s_x}, p_{i_y} - p_{s_y}, p_{i_z} - p_{s_z})}$$

$$= \frac{n_x(a_x - x_s) + n_y(a_y - y_s) + n_z(a_z - z_s)}{n_x(p_{i_x} - p_{s_x}) + n_y(p_{i_y} - p_{s_y}) + n_z(p_{i_z} - p_{s_z})} \dots (4a)$$

Example: Compute the shade by finding intersection pt formed by \vec{p}_S and \vec{p}_4 .

Sol: from Eqn (3a), we have

$$\vec{R} = \vec{p}_S + \lambda(\vec{p}_4 - \vec{p}_S)$$

$$= (x_s, y_s, z_s) + \lambda(x_4 - x_s, y_4 - y_s, z_4 - z_s)$$

from Eqn (4), where

$$\vec{n}(n_x, n_y, n_z) = (0, 0, 1)$$

$$\vec{a}(a_x, a_y, a_z) = (0, 0, 0)$$

Hence,

$$\lambda = \frac{0 \cdot (a_x - x_s) + 0 \cdot (a_y - y_s) + 1 \cdot (a_z - z_s)}{0 \cdot (x_4 - x_s) + 0 \cdot (y_4 - y_s) + 1 \cdot (z_4 - z_s)}$$

$$= \frac{a_z - z_s}{z_4 - z_s} = \frac{0 - 200}{110 - 200} = \frac{200}{90}$$

$$= 20/9$$

Therefore, Sub λ back to the Ray Egn.

$$\begin{aligned}
 \vec{R} &= \vec{P_s} + \lambda (\vec{P_4} - \vec{P_s}) \Big|_{\lambda = 20/q} \\
 &= (100, 100, 200) + \frac{20}{q} (x_4 - x_s, y_4 - y_s, z_4 - z_s) \\
 &= (100, 100, 200) + \frac{20}{q} (102 - 100, 102 - 100, 110 - 200) \\
 &= (100, 100, 200) + \frac{20}{q} \cdot (0, 0, 10) \\
 &= (100, 100, 200) + (0, 0, -\frac{20}{q} \times 10) \\
 &= (100, 100, 200) + (0, 0, -200) = (100, 100, 0)
 \end{aligned}$$

Note: 1. Finish Last Homework,
then Expand to a Cube.
(Display it).

2. Compute/Implement this
Algorithm, to calculate (Hm)

Each of Every 4 pts of top
Surface of the cube.

Note: Homework Submission Extended to 18th
Sunday, 11:59 pm.
 via e-mail ; Subject:
 FirstName+SID(4 Digits)+CmpE240+HW2
 April 14 (Th).

See the figure Next page

```

float Xe = 200.0f;
float Ye = 200.0f;
float Ze = 200.0f;
float Rho = sqrt(pow(Xe,2) + pow(Ye,2) + pow(Ze,2));
float D_focal = 20.0f;
  
```

```
//define the x-y-z world coordinate
world.X[0] = 0.0; world.Y[0] = 0.0; world.Z[0] = 0.0; // origin
world.X[1] = 50.0; world.Y[1] = 0.0; world.Z[1] = 0.0; // x-axis
world.X[2] = 0.0; world.Y[2] = 50.0; world.Z[2] = 0.0; // y-axis
world.X[3] = 0.0; world.Y[3] = 0.0; world.Z[3] = 50.0; // z-axis
```

for X_w - Y_w - Z_w World-Coordinate System

```
typedef struct {
    float X[UpperBD];
    float Y[UpperBD];
    float Z[UpperBD];
} pworld;
```

for X_v - Y_v - Z_v Viewer (Virtual Camera)

```
typedef struct {
    float X[UpperBD];
    float Y[UpperBD];
    float Z[UpperBD];
} pviewer;
```

for Perspective Projection

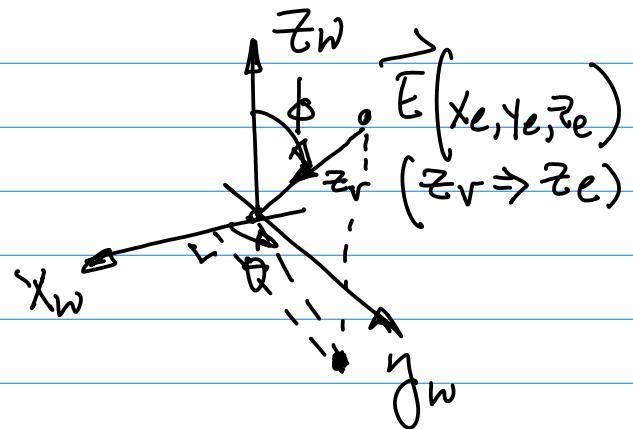
```
typedef struct{
    float X[UpperBD];
    float Y[UpperBD];
} pperspective;
```

Declaration of Each Coordinate System

```
pworld world;
pviewer viewer;
pperspective perspective;
```

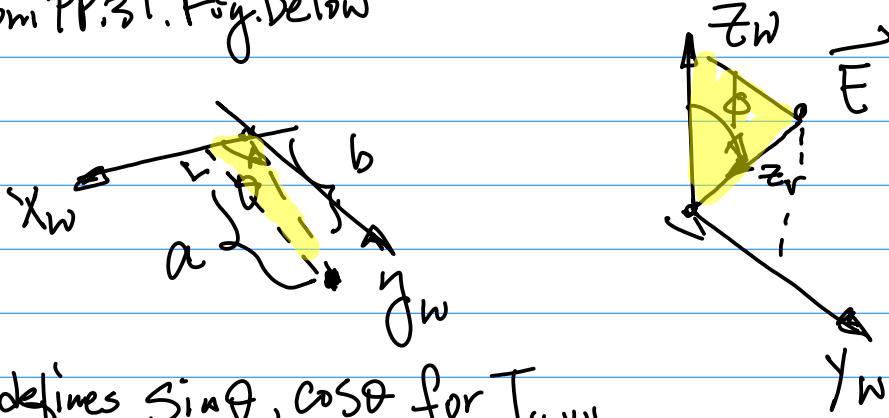
Initialization By Defining 4 vectors

(pts) $\vec{P_1}, \vec{P_2}, \vec{P_3}, \vec{P_4}$ for X_w, Y_w, Z_w
axis



```
//define the x-y-z world coordinate
world.X[0] = 0.0; world.Y[0] = 0.0; world.Z[0] = 0.0; // origin
world.X[1] = 50.0; world.Y[1] = 0.0; world.Z[1] = 0.0; // x-axis
world.X[2] = 0.0; world.Y[2] = 50.0; world.Z[2] = 0.0; // y-axis
world.X[3] = 0.0; world.Y[3] = 0.0; world.Z[3] = 50.0; // z-axis
```

From PP.31. Fig. Below



We defines $\sin\theta, \cos\theta$ for T_{4x4}

World-To-Viewer Transform.

```
float sPhi = Ye / sqrt(pow(Xe,2) + pow(Ye,2));
float cPhi = Xe / sqrt(pow(Xe,2) + pow(Ye,2));
float sTheta = sqrt(pow(Xe,2) + pow(Ye,2)) / Rho;
float cTheta = Ze / Rho;
```

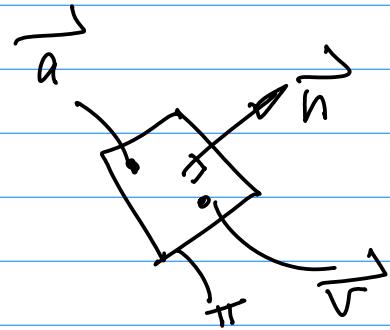
From PP. 32

Define plane Equation By

\cong Normal Vector $\vec{n} (n_x, n_y, n_z) = (0, 0, 1)$

\cong Arbitrary pt $\vec{a} = (0, 0, 0)$

Then a point light Source $\vec{Ps} (x_s, y_s, z_s)$



```
world.X[45] = -200.0; world.Y[45] = 50.0; world.Z[45] = 200.0; // Ps (point source)
world.X[46] = 0; world.Y[46] = 0; world.Z[46] = 0; // arbitrary vector A on x-y plane
world.X[47] = 0; world.Y[47] = 0; world.Z[47] = 1; // normal vector for x-y
```

Now, the Implement for Computing World-to-Viewer By T_{4x4}

CMPED240

37

```
for(int i = 0; i <= UpperBD; i++)
{
    viewer.X[i] = -sPhi * world.X[i] + cPhi * world.Y[i];
    viewer.Y[i] = -cPhi * cPhi * world.X[i]
    - cPhi * sPhi * world.Y[i]
    + sPhi * world.Z[i];
    viewer.Z[i] = -sPhi * cPhi * world.X[i]
    - sPhi * cPhi * world.Y[i]
    -cPhi * world.Z[i] + Rho;
}
```

from TP.29, T_{4x4}

$$\mathbf{T} = \begin{bmatrix} -\sin \theta & \cos \theta & 0 & 0 \\ -\cos \phi \cos \theta & -\cos \phi \sin \theta & \sin \phi & 0 \\ -\sin \phi \cos \theta & -\sin \phi \cos \theta & -\cos \phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{pmatrix} X_v \\ Y_v \\ Z_v \\ 1 \end{pmatrix} = T_{4 \times 4} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

World-to-
Viewer
Transform

"After" "Before"

Now, Perspective Projection from TP.29

```
perspective.X[i] = D_focal * viewer.X[i] / viewer.Z[i];
perspective.Y[i] = D_focal * viewer.Y[i] / viewer.Z[i];
```

$$x_p = x_e \left(\frac{D}{z_e} \right)$$

$$y_p = y_e \left(\frac{D}{z_e} \right)$$

For Intersection Computation
 ≡ find x in $X_w - Y_w - Z_w$,
 ≡ find intersection pt(S),
 in $X_w - Y_w - Z_w$

Then, Computer Virtual Display (2D)

to Physical Display ! Then plot the points !

Then Transform
 with pipeline
 then Done !

Ch 10

Diffuse Reflection.

Background / Basic Concepts

1. Objective: To generate realistic looking 3D Graphics

Lighting Models

After Trans.
Pipeline

$$\vec{I}(x,y) = (r(x,y), g(x,y), b(x,y))$$

Graphics (Vector)
Graphics

Primitive ... (1)
Colors,

r, g, b : 8 bit Resolution.

$r, g, b \in [0, 255]$, for 15 bits, or 16 bit

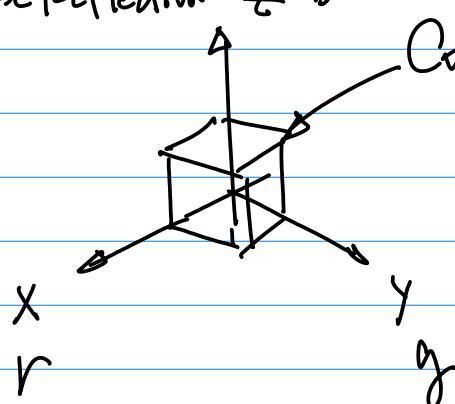
r, g, b. is Common.

2. Color Space

April 19 (Monday)

Diffuse Reflection $\approx b$

Fig 1.



Color Cube

$$\vec{I}(x,y,z) = \vec{I}_1(x,y,z) + \vec{I}_2(x,y,z) + \vec{I}_3(x,y,z) \quad \dots (2)$$

3D Space in $x_w - y_w - z_w$

$\vec{I}_1(x,y,z)$: Diffuse Reflection

Diffuse Reflection (Ref. on \vec{P}_S)

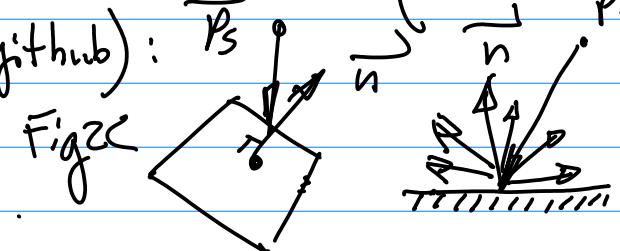


Fig 2C

Note: in r-g-b Color Space,

(255, 0, 0) : Brightest Red

(0, 255, 0) : Brightest Green, etc.

line connecting $(0,0,0)$ to $(1,1,1)$: gray, $(0,0,0)$ Black, $(1,1,1)$ Bright White

3. Color Contribution to Each pixel on a Surface of a given Object in 3D Space.

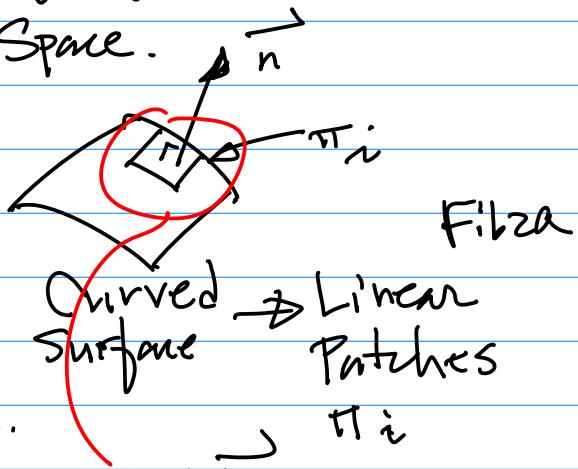
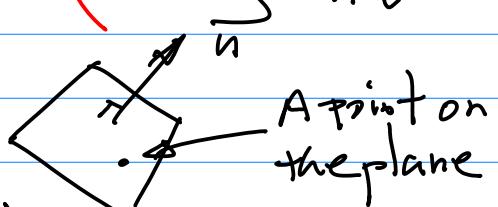


Fig 2b



A point on the plane

$$\vec{I}(x,y,z) = \vec{I}_1(x,y,z) + \vec{I}_2(x,y,z) + \vec{I}_3(x,y,z) \quad \dots (2)$$

Comptext

39

Diffuse Reflection: \curvearrowright that reflects incoming light uniformly in all different directions.

Example: Diffuse Reflection model.

Step 1. $\vec{I}(x, y, z)$ Color on a pt. of a surface in

4. Reflectivity for all 3 different light models (I_r, I_g, I_b)

$\vec{I}(x, y, z) \leftarrow$ 3D Space.
 $(\vec{I}_r(x, y, z), \vec{I}_g(x, y, z),$
 $\vec{I}_b(x, y, z))$

Define $R = (r_r, r_g, r_b) \dots [3]$

Simplify the discussion by focusing on one primitive color

$$0 \leq r_r, r_g, r_b \leq 1$$

$r_r = r_g = r_b = 0$ Black

$r_r = r_g = r_b = 1$ White

$r_r = 0, r_g = 0.75, r_b = 0$: Green

$$I_r(x, y, z)$$

Energy of Photons from the Source reaching the pt. is inversely proportional to the squared distance

Physical Characteristics of the given Surface

Now, Specular Reflection

I_s : Reflection Produces bright light, it is a function of E position.

I_a : Ambient light, from indirect light source, can be simulated by adding constant values to r, g, b .

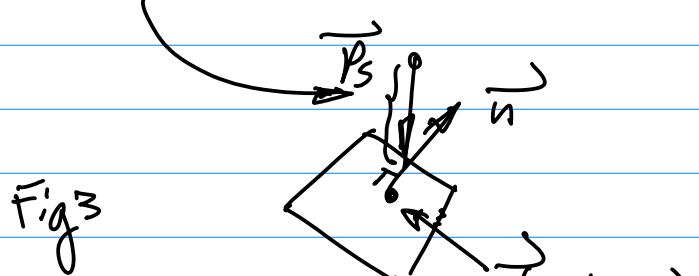
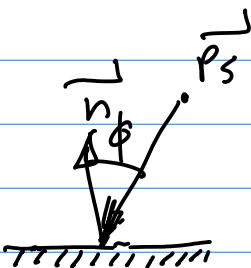


Fig 3

$$\|\vec{P}_s - \vec{P}_i\|^2 = \|\vec{r}\|^2 \dots (4)$$

Ray Equation \vec{r}

$$\vec{I}_r(x, y, z) \sim \frac{1}{\|\vec{r}\|^2} \dots (5)$$



ϕ : phi Angle
of the incoming
light

$\phi = 0$, Strongest incoming light \rightarrow "1" Normalize it

$\phi = \frac{\pi}{2}$ photons miss the pt. on the
Surface \rightarrow Normalize it.
"0"

Cosphi Rule. Dot Product

$$\vec{n} \cdot (-\vec{P}_s) = \|\vec{n}\| \|\vec{-P}_s\| \cos\phi \dots (5)$$

$$\cos\phi = \frac{\vec{n} \cdot (-\vec{P}_s)}{\|\vec{n}\| \|\vec{-P}_s\|} = -\frac{\vec{n} \cdot \vec{P}_s}{\|\vec{n}\| \|\vec{P}_s\|} \quad \text{From Eqn(5), (10)}$$

... (b)

define $-\vec{P}_s$ as Ray Vector

$$\vec{r} \triangleq \vec{P}_i - \vec{P}_s \quad \dots (i)$$

$$\vec{I}_r(x, y, z) = K_r \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} \frac{1}{\|\vec{r}\|^2} \quad \dots (ii)$$

Hence

$$\cos\phi = \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} \quad \dots (8)$$

where
 $\|\vec{r}\|^2 = \|\vec{P}_i - \vec{P}_s\|^2$

$$\vec{I}_r(x, y, z) \sim \cos\phi \left(= \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} \right) \quad \dots (g)$$

$$= (x_i - x_s)^2 + (y_i - y_s)^2 + (z_i - z_s)^2$$

$$\vec{I}_r(x, y, z) \sim K_r \underset{(i)}{\text{Reflectivity for } r} \quad \text{Preparation for}$$

April 21 (Wed)

Lab (Project) 3D GE, (Diffuse Reflection) Due 2 weeks, May 9th Sunday 11:59 pm.

- a. 2018F-115 - Rubrics
- b. 2018F-116, C++ Sample code
- c. 2018F-117 DDA
- d. 2018F-118 Interpolation

Example: Design/Implementation for Project On Diffuse Reflection.

(See github to find Ref. 2018S ~ Diff)

Step 1. Define Data Points (Vertices,

$\vec{P}_i(x_i, y_i, z_i)$ in the World

Coordinate System)

$\left\{ \vec{P}_i(x_i, y_i, z_i) \mid i=1, 2, \dots, 8 \right\}$ for a Cube,

Only interested in Top Surface

$\left\{ \vec{P}_i(x_i, y_i, z_i) \mid i=1, 2, \dots, 4 \right\}$

Cube with Side of 100, And Elevated by 10.

Only Consider Diffuse Reflection

On the top Surface formed by

$\{ \vec{P}_1, \vec{P}_2, \vec{P}_3, \vec{P}_4 \}$

$$\begin{aligned} \vec{P}_1(0, 100, 110), \vec{P}_2(0, 0, 110) \\ \vec{P}_3(100, 0, 110), \vec{P}_4(100, 100, 110) \end{aligned}$$

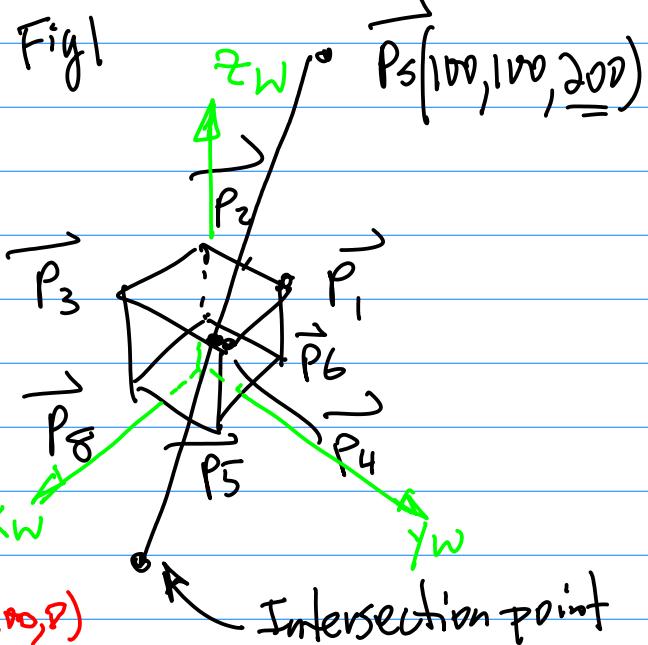
Step 2. Define Point Light

Sample $\vec{P}_s(200, 200, 200)$, white and Reflectivity

$$\vec{K}(K_r, K_g, K_b) = (0.8, 0, 0)$$

Question:

Do we have to perform Transformation Pipeline Computations in order to Compute Diffuse Reflection? No



Chap 24

42

$$I_r(x_4, y_4, z_4) = K_r \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} = \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} = \frac{z_4 - z_s}{\sqrt{(x_4 - x_s)^2 + (y_4 - y_s)^2 + (z_4 - z_s)^2}}$$

Since $\vec{P}_s(200, 200, 200)$,

$$\vec{r} = \vec{P}_i - \vec{P}_s \quad |_{n=4} = \vec{P}_4 - \vec{P}_s$$

$$= (x_4 - x_s, y_4 - y_s, z_4 - z_s) \quad ... (2)$$

$$\|\vec{r}\|^2 = \left(\sqrt{(x_4 - x_s)^2 + (y_4 - y_s)^2 + (z_4 - z_s)^2} \right)^2$$

$$= (x_4 - x_s)^2 + (y_4 - y_s)^2 + (z_4 - z_s)^2$$

$$= 100^2 + 100^2 + 90^2$$

And $\cos \phi$

$$\vec{n}(n_x, n_y, n_z) = (0, 0, 1)$$

$$\vec{n} \cdot \vec{r} = (0, 0, 1) \cdot (r_x, r_y, r_z)$$

$$= (0, 0, 1) \cdot (x_4 - x_s, y_4 - y_s, z_4 - z_s)$$

$$= 0 \cdot (x_4 - x_s) + 0 \cdot (y_4 - y_s) + 1 \cdot (z_4 - z_s)$$

$$= z_4 - z_s$$

$$\|\vec{n}\| \|\vec{r}\| = \sqrt{n_x^2 + n_y^2 + n_z^2} \cdot \sqrt{r_x^2 + r_y^2 + r_z^2}$$

$$= \sqrt{0^2 + 0^2 + 1^2} \cdot \sqrt{(x_4 - x_s)^2 + (y_4 - y_s)^2 + (z_4 - z_s)^2}$$

$$= 1 \cdot \sqrt{(x_4 - x_s)^2 + (y_4 - y_s)^2 + (z_4 - z_s)^2}$$

$$\sqrt{(x_4 - x_s)^2 + (y_4 - y_s)^2 + (z_4 - z_s)^2}$$

$$= \frac{110 - 200}{\sqrt{100^2 + 100^2 + 90^2}}$$

No! Positive Value
Only

Change Eqn (2) from
 $\vec{P}_i - \vec{P}_s$ to $\vec{P}_s - \vec{P}_i$

Therefore, we have

$$\frac{200 - 110}{\sqrt{100^2 + 100^2 + 90^2}}$$

So, we have Diffuse
Reflection @ Pt. 4 \vec{P}_4

$$I_{\text{diff}} = K_d \frac{200 - 110}{\sqrt{100^2 + 100^2 + 90^2}}$$

0.8

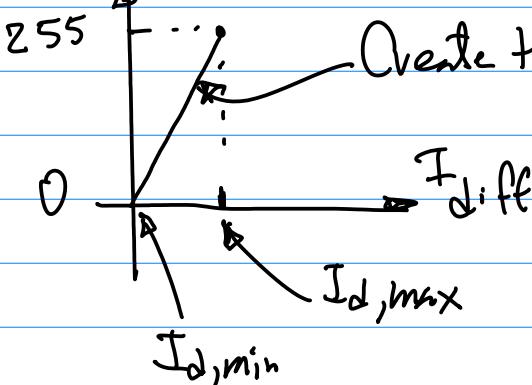
$$\frac{1}{100^2 + 100^2 + 90^2}$$

$$I_{\text{diff}} = 0.8 \frac{200 - 110}{\sqrt{100^2 + 100^2 + 90^2} (100^2 + 100^2 + 90^2)}$$

Note:

1° I_{diff} Computed is Very Small!

You Need Scaling factor to Scale it up for the dynamic Range of your Hardware (LCD) Display Device Scaled for Display



Add Offset 20 to I_{diff} . So it is more visible.

Question: How to Expand the Computation to the Rest of the points on the top Surface?

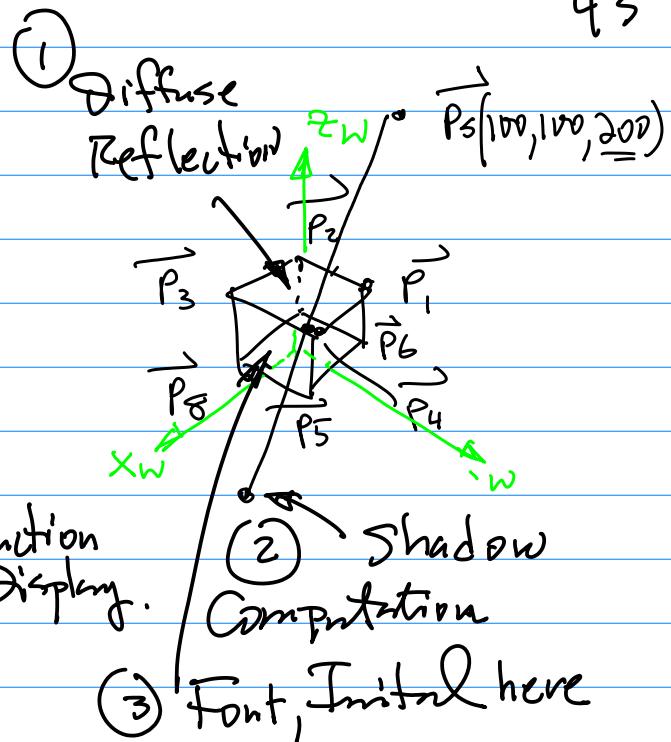
April 26 (mon). Final Exam.

May 19 (wed) | 215-1430

Project Extended to May 12 (wed)

11:59 pm. Written Requirements

Will be posted on CANVAS and on github.



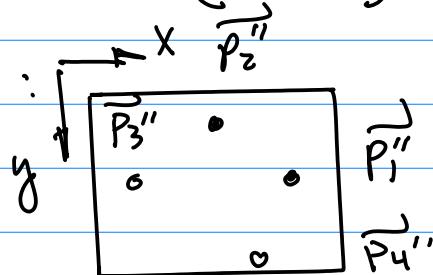
Example:

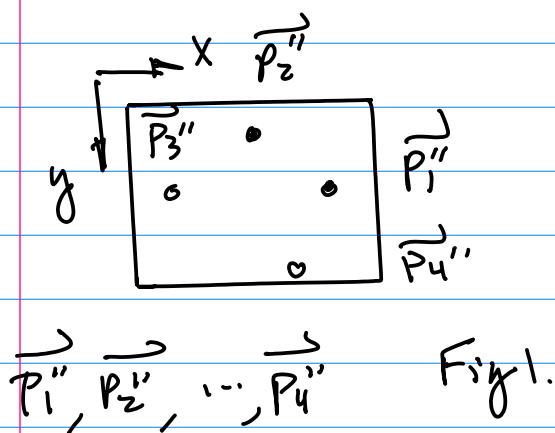
Given the 4 points ($\vec{P}_1, \dots, \vec{P}_4$) Diffuse Reflection Already Computed, ($(x_w - y_w - z_w)$)

Compute the Boundary Diffuse Reflection (Part I)

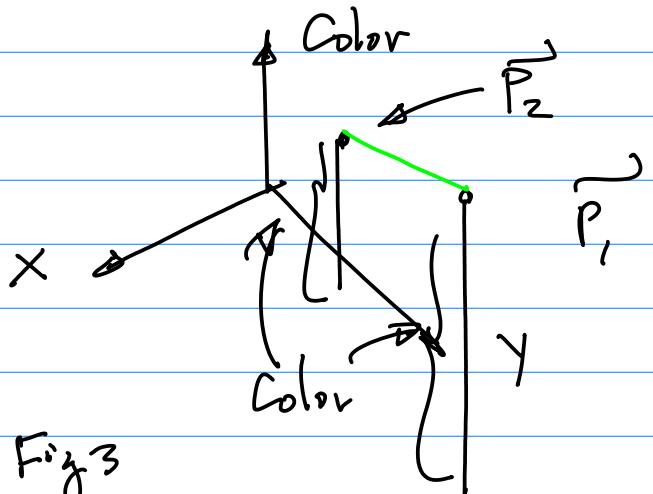
Compute Interior Points of the Surface defined by the Boundaries. (Part II)

PART I :

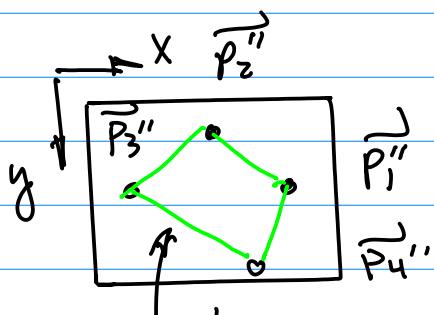




Color Space for Illustration



- 1' Their Location on LCD Display are Computed By Transformation Pipeline
- 2' Their Color & Intensity are Computed in (x_w, y_w, z_w) By Diffuse Reflection Model;



Boundary Color To Be Computed By a Simpler Technique → Interpolation.

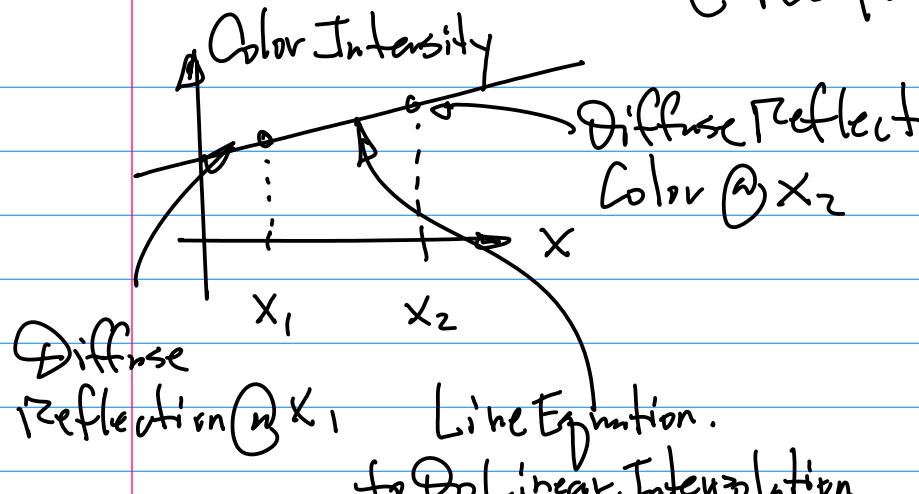
How To Compute Diffuse Reflection from 2 Known pts (Starting & Ending Pt of A Line) ?

Interpolation :
Assume moving from P_1 to P_2
Along "Kine" Line, the Intensity,
e.g. color, is changing as
a function of x and y .

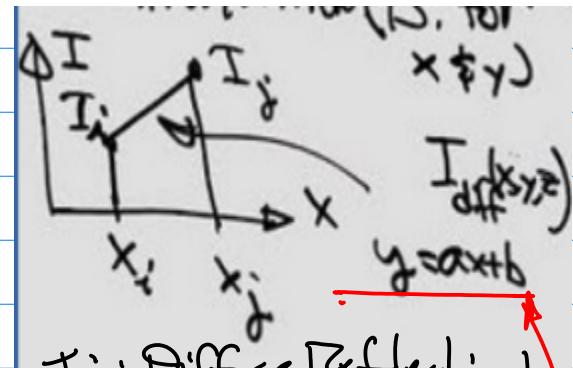
- (1) Contribution from x if color changing
- (2) Contribution of color changing from y

(3) Then the final color is the average of Contribution x and Contribution from y .

Compeacto

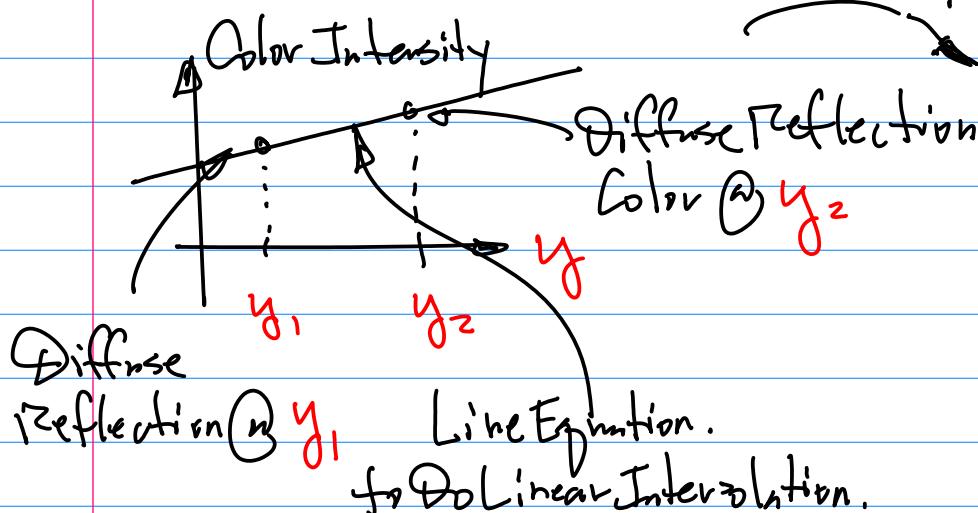


Notation from the 45 (github) PPT.

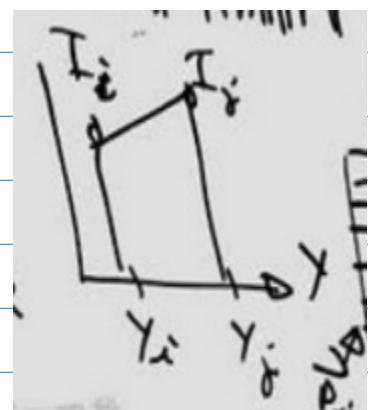


Remember Table - look-up for Sine, Cosine Value Computations.
(Interpolation).

Similarly for y



From P.P.T for y.



Note, Derivation for x and y contributions
is the same.

$$I_{\text{diff}}(y) = \frac{I_{\text{diff}}(y_i) - I_{\text{diff}}(y_j)}{y_i - y_j} y - \frac{y_j (I_{\text{diff}}(y_i) - I_{\text{diff}}(y_j))}{y_i - y_j} + I_{\text{diff}}(y_i)$$

$$\frac{x - x_2}{x_1 - x_2} = \frac{y - y_2}{y_1 - y_2}$$

Diffuse Reflection Contributed, $I_{\text{diff}}(y_i)$, $I_{\text{diff}}(y_j)$
by y. Starting, Ending pt.
Diffuse Reflection, Known. Diff. Value.