

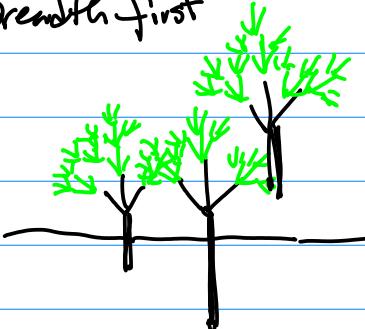
- a A set of Squares Rotation  
with One Same Color  
b Change location/Color/size

C

Keep the patterns, please  
don't remove them!

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

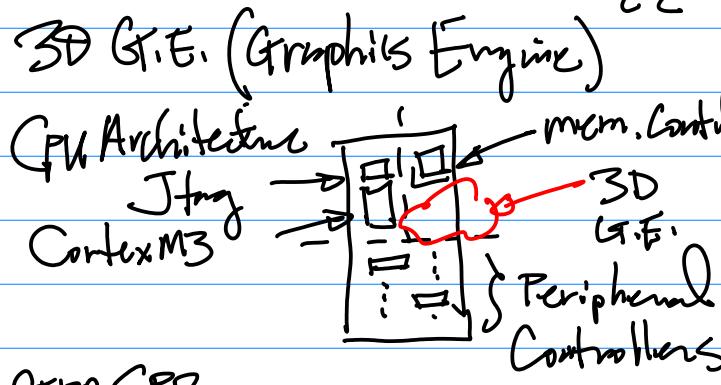
a Breadth first



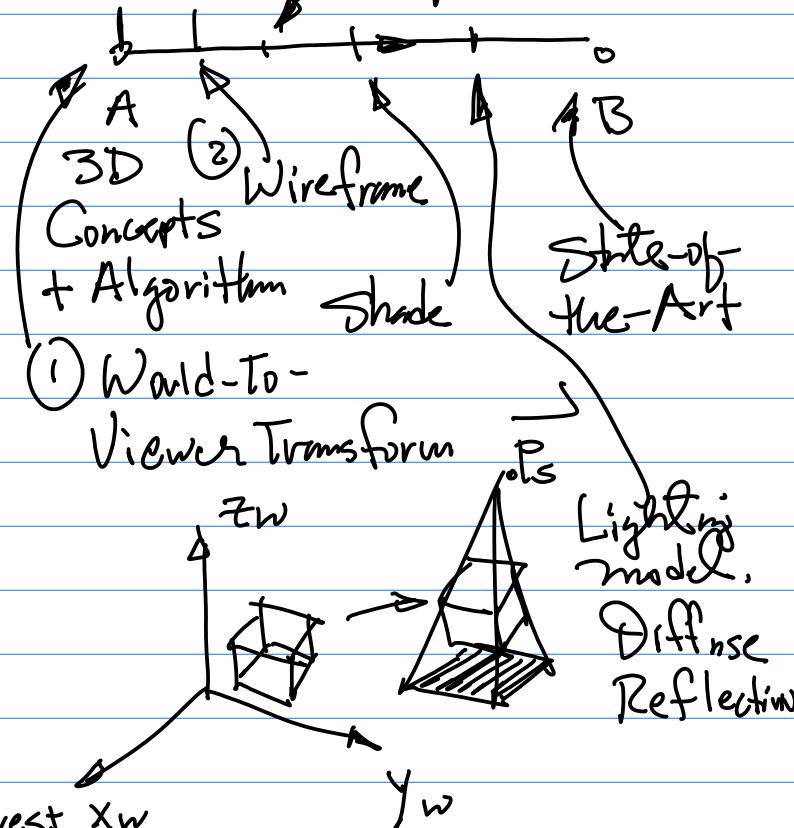
b L>> C Create A patch of Forest  
Changing Location/Size

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)

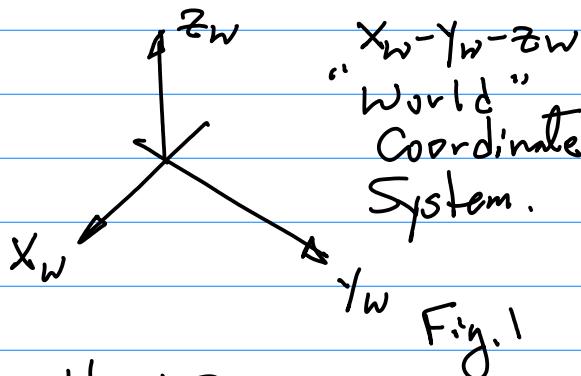
Note: 1<sup>st</sup> midterm A week from  
this Wednesday (March 24th)

2<sup>nd</sup> 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,

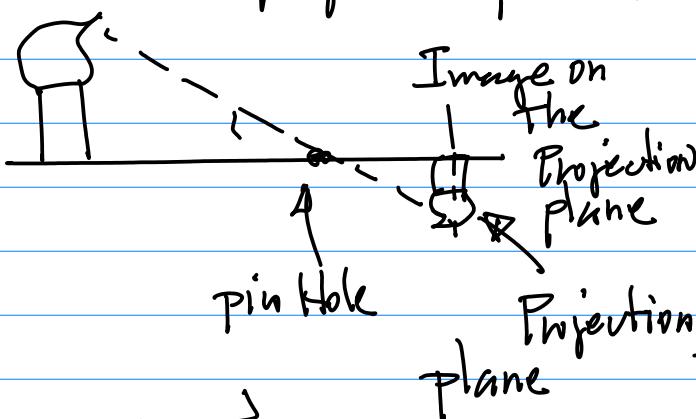
2° Virtual Camera

a Virtual Enclosure

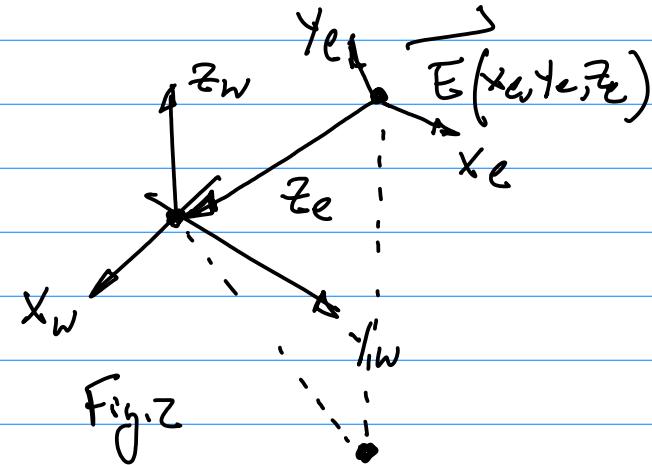
b Optic Lens (Pin Hole)

$\phi_d \approx b$  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.

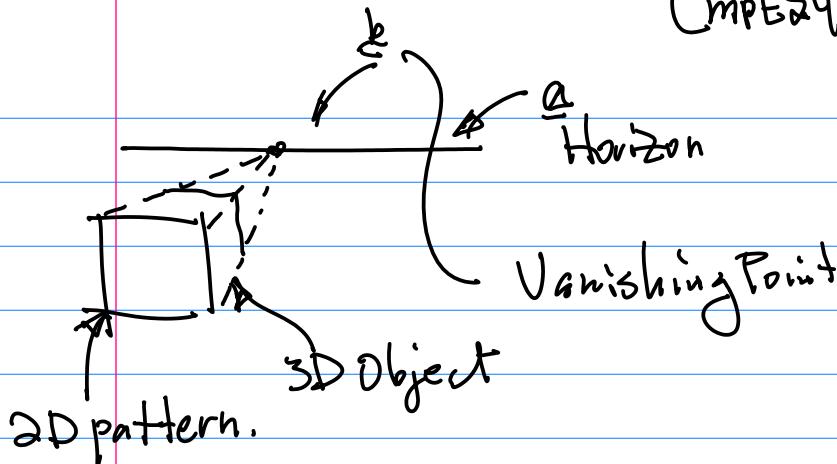
b Left-Hand System.

c  $Z_e$ -axis points to the origin.

5. Perspective Projection

Complex

24



3D Object formed on 2D plane

## 7. Transformation Pipeline

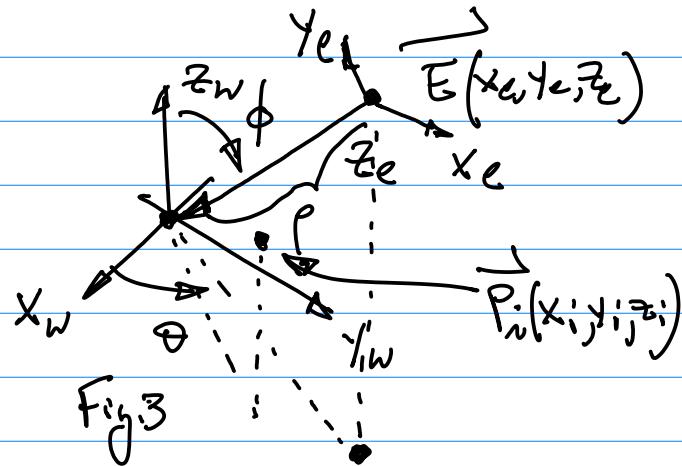
To Display 3D object(s) on to 2D LCD Display Screen,

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



3 parameters:

$\hat{a}$  theta  $\theta$  Angle

$\hat{b}$  Vector  $\vec{E}$  to  $\vec{O}$ ,

"röh", Distance

## Mathematical Formulation

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

$\hookrightarrow$  2<sup>o</sup> Foundation

$\hookrightarrow$  a Translation

$\hookrightarrow$  Rotations  $\left\{ \begin{array}{l} \text{wrt } X_w \text{ and} \\ \dots \text{ wrt } Y_w \\ \dots \text{ wrt } Z_w \end{array} \right.$  wrt Arbitrary Axis

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

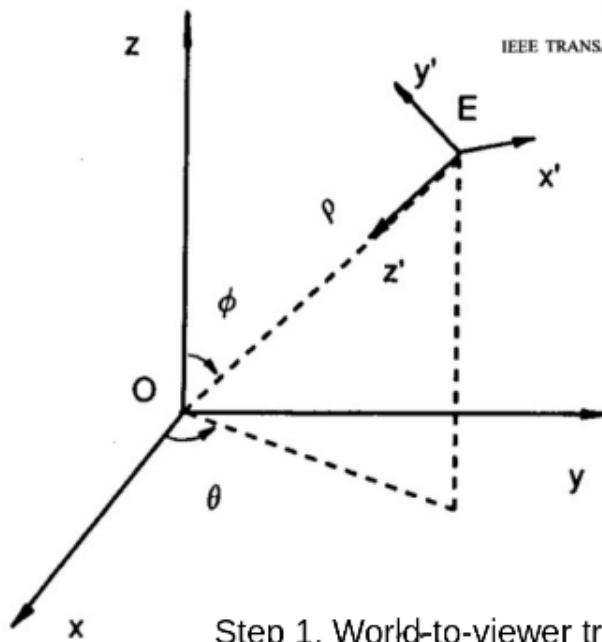
$\phi$ .

World\_2\_Visitor

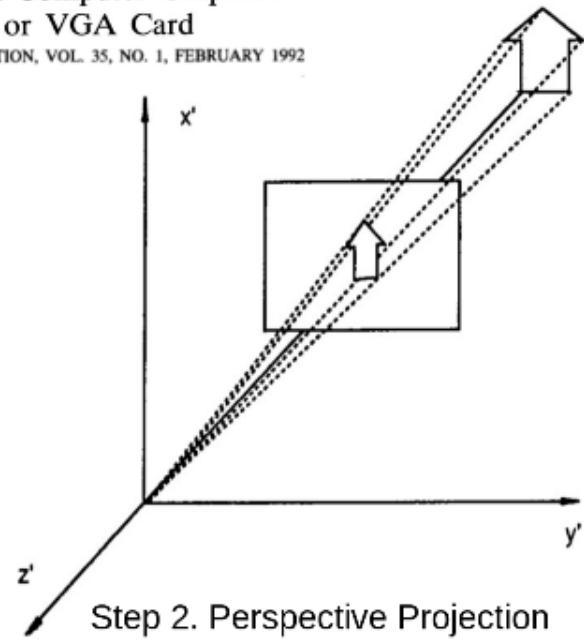
# 3D Transformation Pipeline Technique

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

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

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

Part II  
"φ"

RTOS

Peripheral Controller

GPP  
SPI

SPI's SPR.

1. Naming Convention

LPC\_SC → PCONP

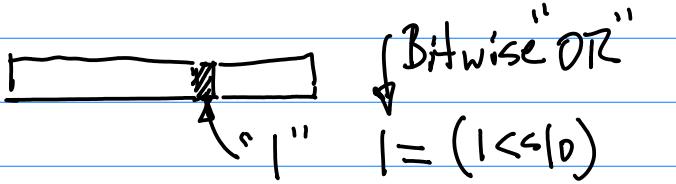
2. Power Up the Selected Peripheral Controller By Setting the Corresponding.

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

$r_{oh}$

# CmpE240

26



Tech Spec.

1° 8 bit Transfer,

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

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

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

2° SPI

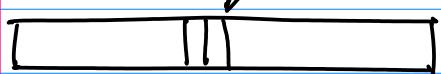
$$CR\phi[5:4] = 00 \text{ (SPI)}$$

3° Clock  $f_{\text{SPI}}$

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

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

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 $\phi$ , CR $\lambda$   
for SPI I/F

Ref:

1° CPU Datasheet.

PP431-433.

Note:  $\begin{cases} CR\phi \\ CR\lambda \end{cases}$  LPC\_SSP1  $\rightarrow$  CR $\phi$   
Control Register

Table 57 |



CR $\phi$

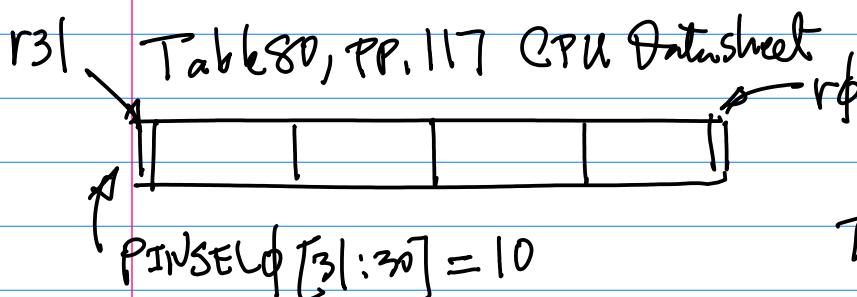
2° Sample code

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  $f_{\text{SPI}}$ .

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

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

c 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

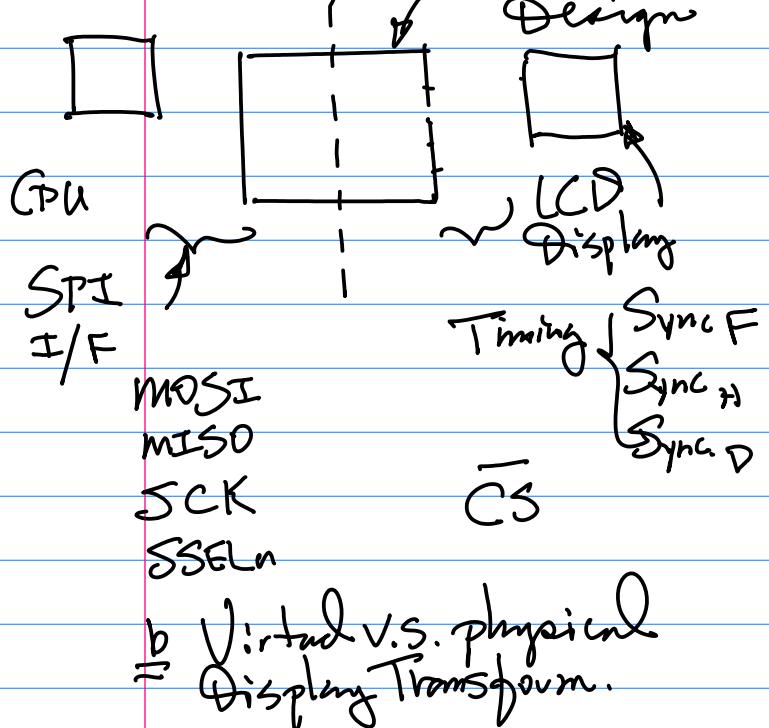
$\text{CR1}[2] = 0$ , Software: Coding  
for "Master"

# 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



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

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

Composition of  
 2D Transform.

$\begin{cases} R_{3 \times 3} \\ T_{3 \times 3} \end{cases} \rightarrow$  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;

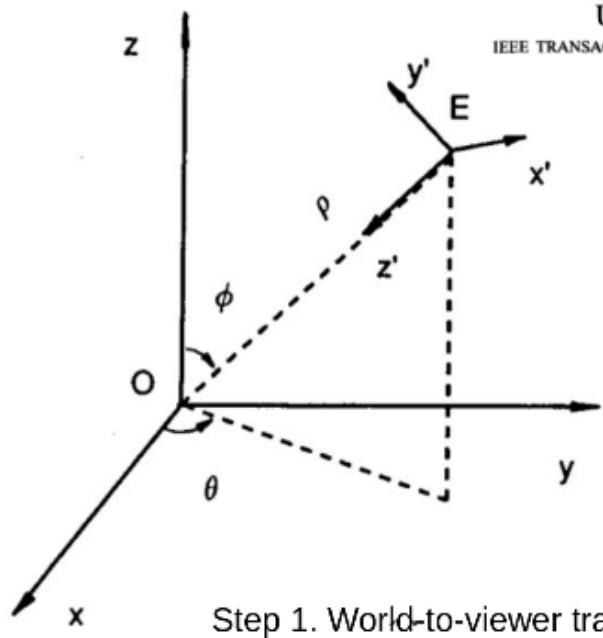
April 5 (Monday)

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

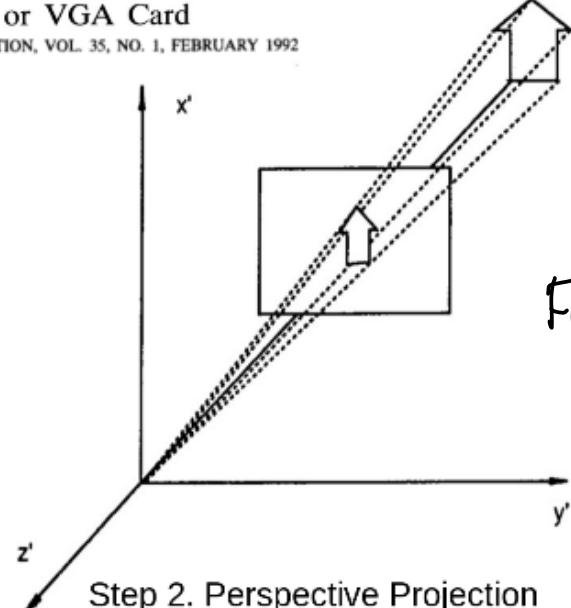


Fig1.

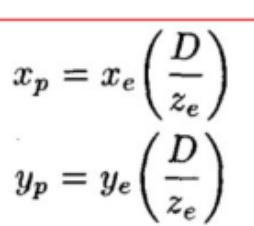
$$\mathbf{T} = \begin{bmatrix} -\sin \theta & \cos \theta & 0 & 0 \\ -\cos \phi \cos \theta & -\cos \phi \sin \theta & \sin \phi & 0 \\ -\sin \phi \cos \theta & -\sin \phi \sin \theta & -\cos \phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

... (1)

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

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

... (2)



Harry Li, Ph.D. mem!

Today's Topics: 3D G.E.

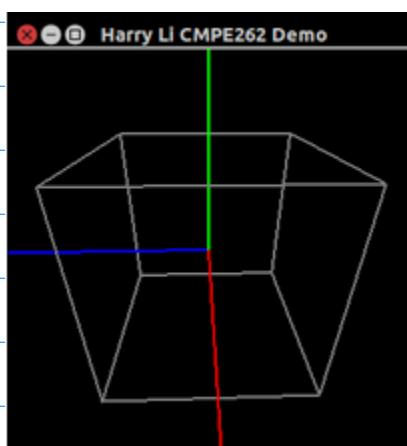


Fig2a

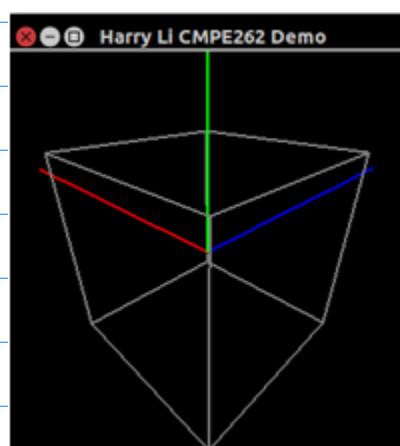
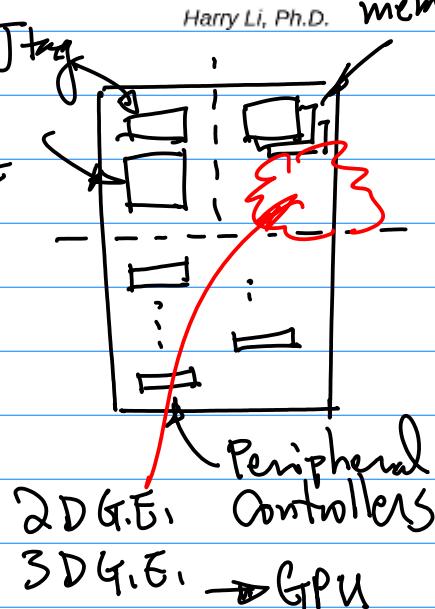


Fig2b



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

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

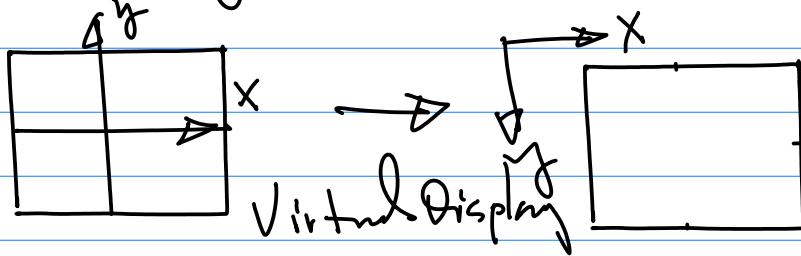
Coordinate on your  
LPC1768 Display  
Device.

Note:

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

Virtual Display Coordinate

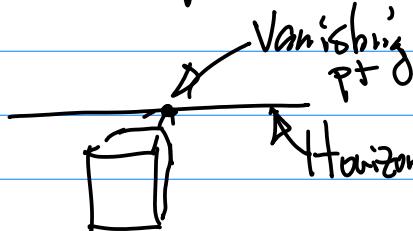
Need to perform Virtual to  
Physical Transformation in order  
to plot your result.



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

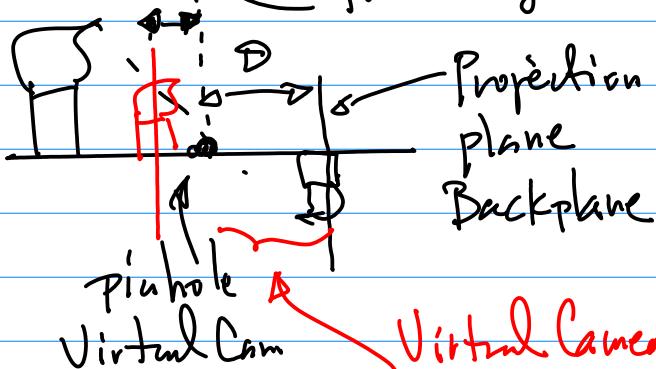
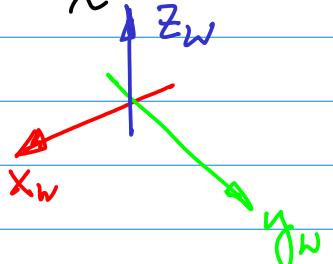


Fig1.

In Homework,  
 $D=20$



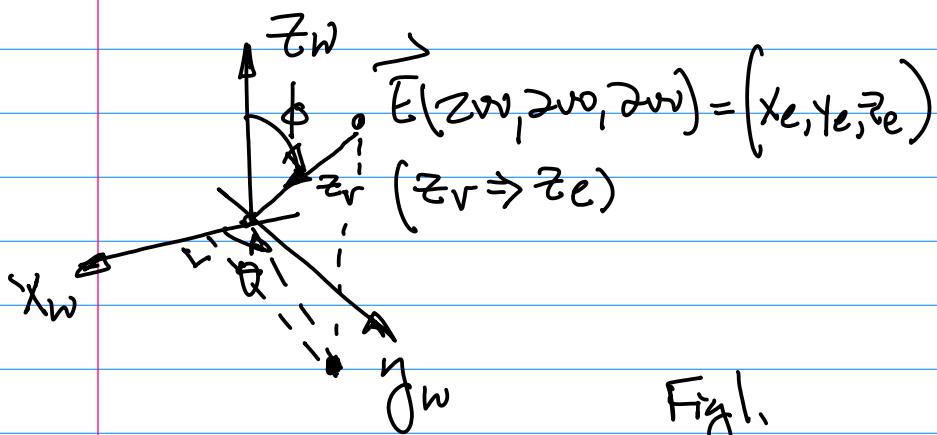


Fig. 1.

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

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  $\phi$  (phi)

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

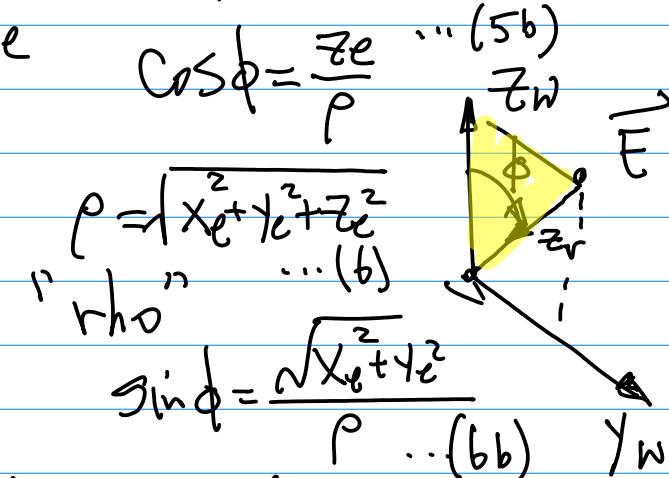
Note: Angle  $\phi$  (phi) on  $Z_v - Z_w$   
plane,

Now, find each entry on  $4 \times 4$  matrix, April 7 (Wed)

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

for Angle  $\theta$  (Theta)

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

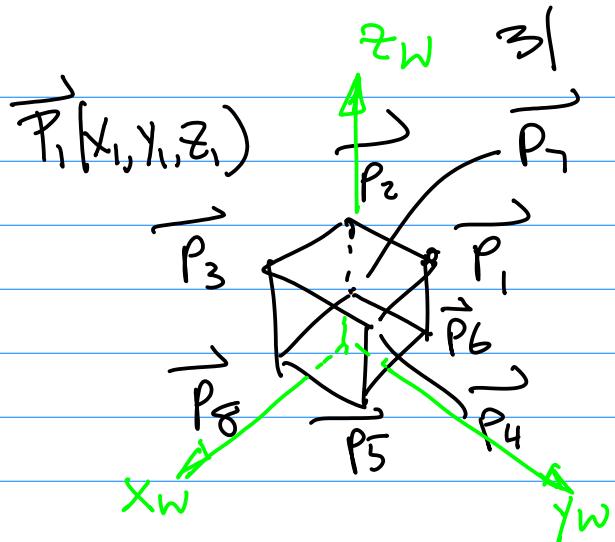


$$\cos \phi = \frac{z_e}{\rho} = \frac{200}{\sqrt{200^2 + 200^2 + 200^2}}$$

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

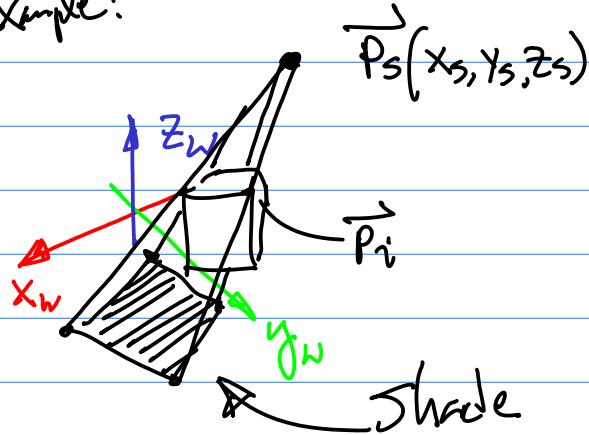
Similarly

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



Consider "Shade" Calculation.

Example:



1.  $x_w - y_w - z_w$ .

Assignment (Monday)

Note: 1<sup>o</sup> Homework Submission (E-mail)

$x_w - y_w - z_w$  Drawing.

Source Code — Photo

(Exported Project) By Wednesday.

2<sup>o</sup> Bring your Prototype

Board to each Session for  
Inspection, Show & Tell.

Discussion on 3D Shade

Computation

Conditions for this discussion

1<sup>o</sup> 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.

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

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

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

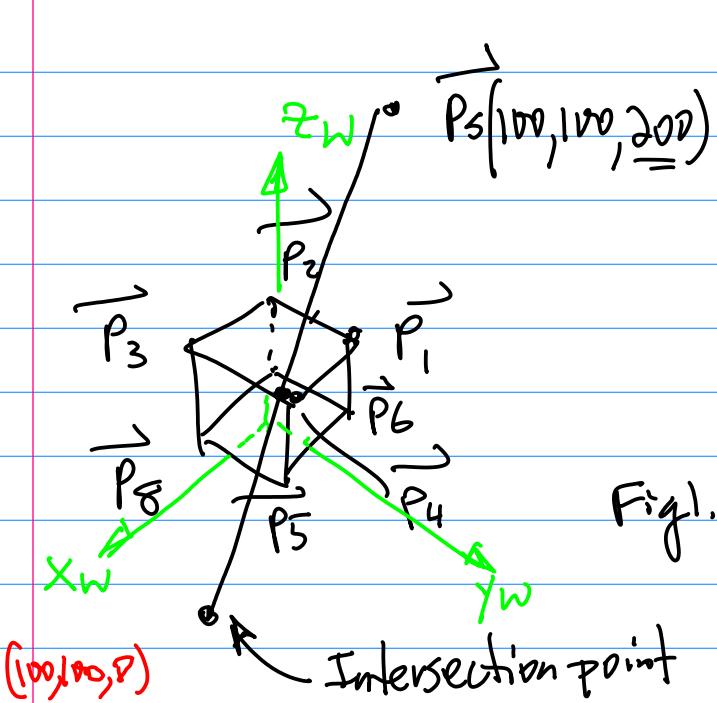


Fig.1.

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

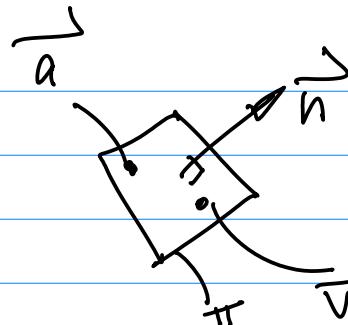


Fig.2

4b. plane  $\pi$ .

An Known, Arbitrary pt  
On  $\pi$  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  $\lambda$  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  $\lambda$  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

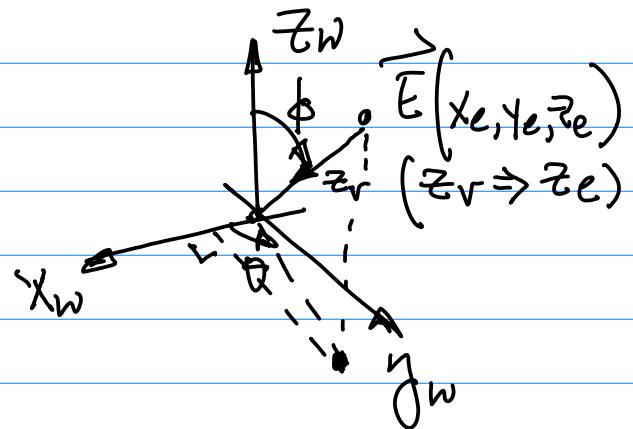
```
typedef struct{
    float X[UpperBD];
    float Y[UpperBD];
} pperspective;
```

Declaration of Each Coordinate System

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

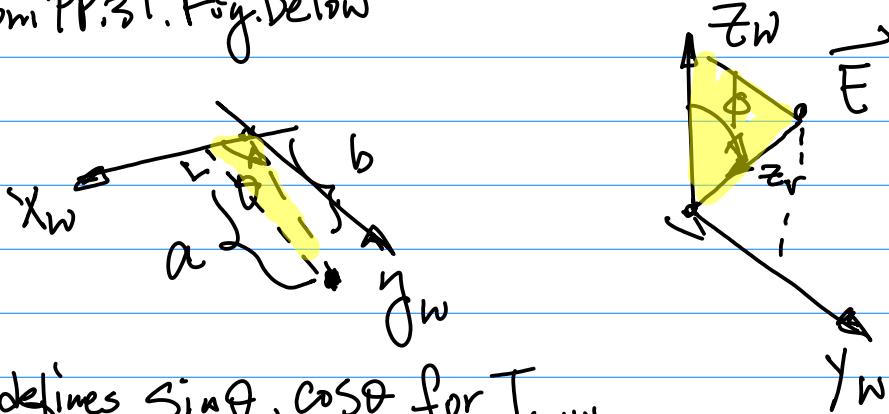
Initialization By Defining 4 vectors

(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;
```

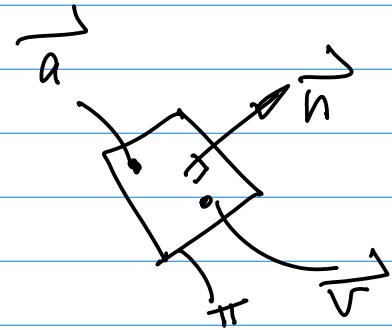
From PP. 32

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] = -sPhieta * world.X[i] + cPhieta * world.Y[i];
    viewer.Y[i] = -cPhieta * cPhi * world.X[i]
    - cPhi * sPhieta * world.Y[i]
    + sPhi * world.Z[i];
    viewer.Z[i] = -sPhi * cPhieta * world.X[i]
    - sPhi * cPhieta * world.Y[i]
    - cPhieta * world.Z[i] + Rho;
}
```

from TP.29, T<sub>4x4</sub>

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

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

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

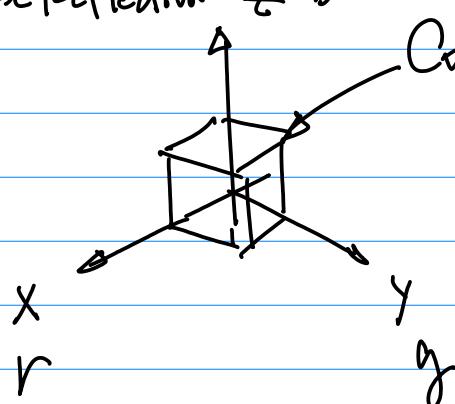
r, g, b. is Common.

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  $\vec{P}_S$ )

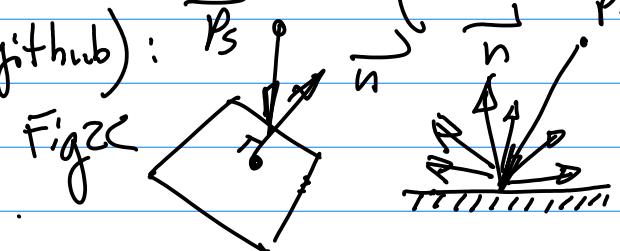


Fig 2C

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.

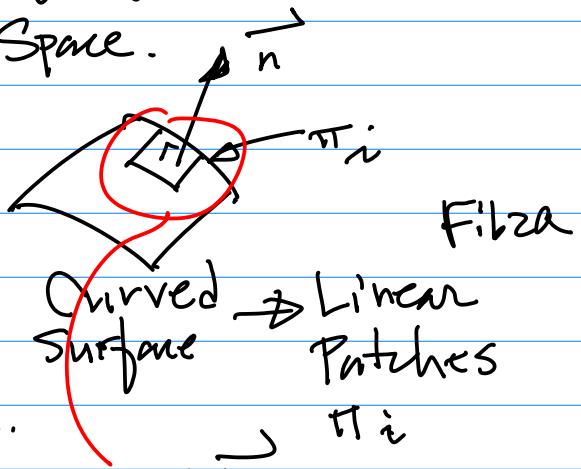
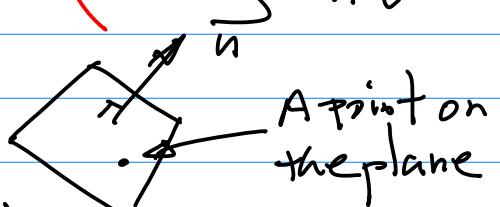


Fig 2b



A point on the plane

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

# Comptex40

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.  
 $I_r(x, y, z), I_g(x, y, z),$   
 $I_b(x, y, z)$

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.

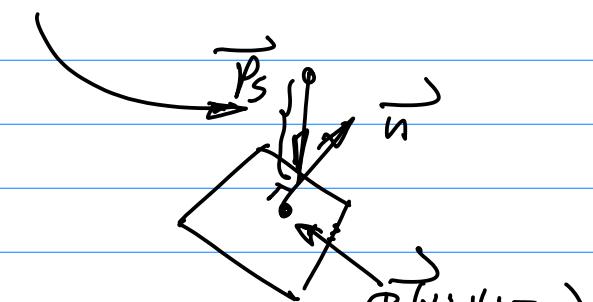
$I_a$ : Ambient light, from indirect light source, can be simulated by adding constant values to  $r, g, b$ .

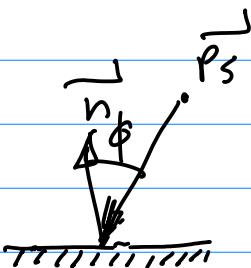
Fig 3

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

Ray Equation  $\rightarrow$

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





$\phi$ : 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

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

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

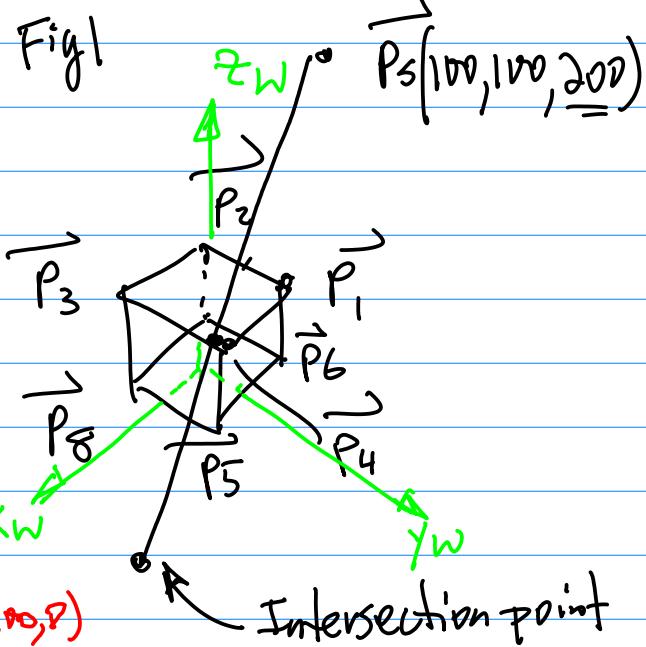
Step 2. Define Point Light

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

$$\vec{r} \stackrel{\Delta}{=} \vec{P}_i - \vec{P}_s \quad |_{n=1} = \vec{P}_4 - \vec{P}_s$$

$$= (x_4 - x_s, y_4 - y_s, z_4 - z_s) \quad \dots (1)$$

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

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

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

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

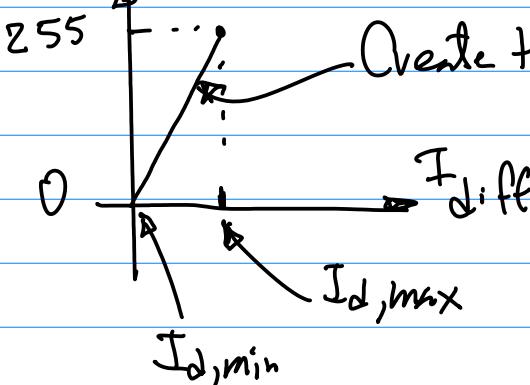
$$\frac{1}{100^2 + 100^2 + 90^2}$$

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

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?

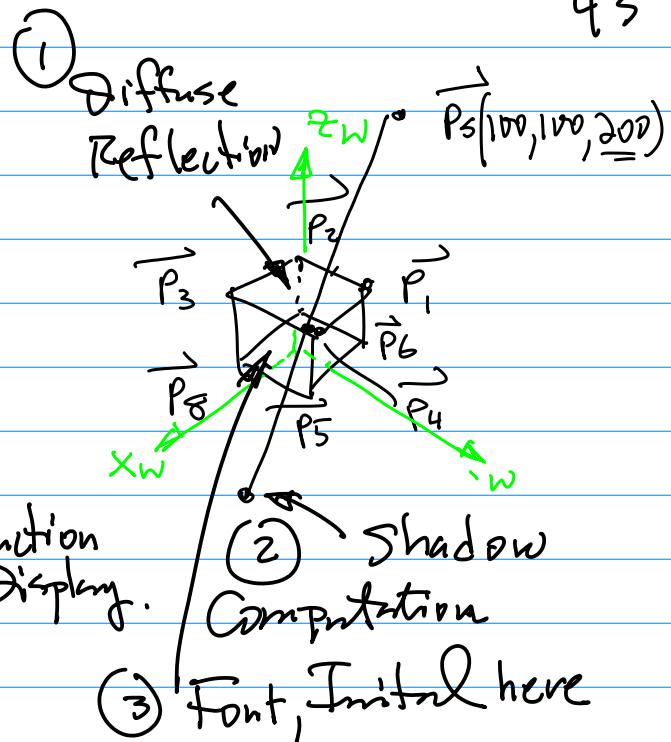
April 26 (mon). Final Exam.

May 19 (wed) [215-1430]

Project Extended to May 12 (wed)

11:59 pm. Written Requirements

Will be posted on CANVAS and on github.



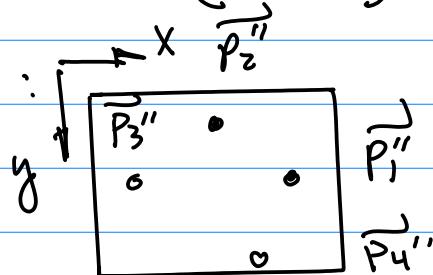
Example:

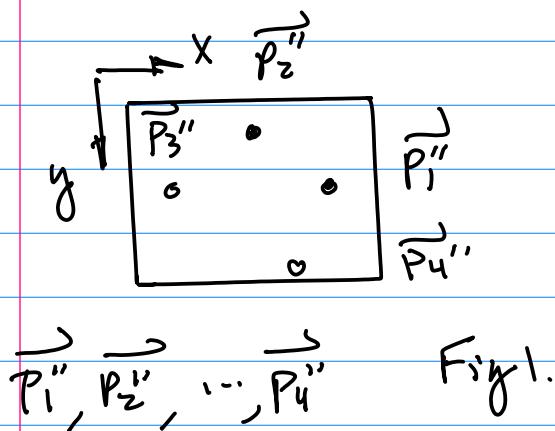
Given the 4 points  $(\vec{P}_1, \dots, \vec{P}_4)$   
Diffuse Reflection Already  
Computed, ( $x_w - y_w - z_w$ )

Compute the Boundary Diffuse  
Reflection (Part I)

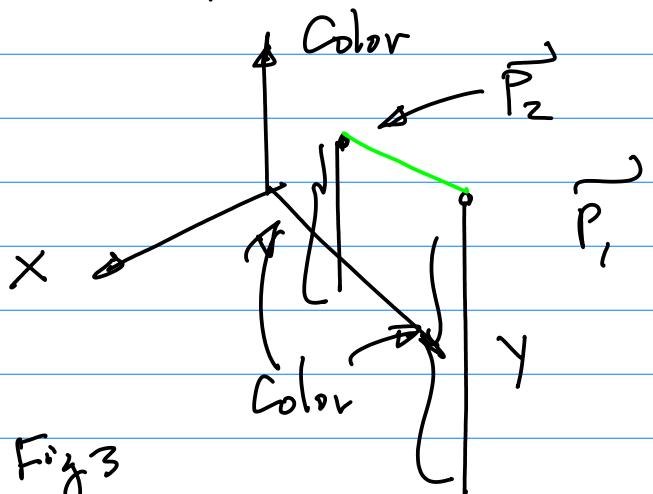
Compute Interior Points of the  
Surface defined by the  
Boundaries. (Part II)

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;

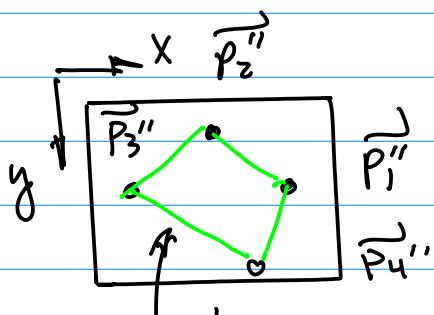


Fig 2

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

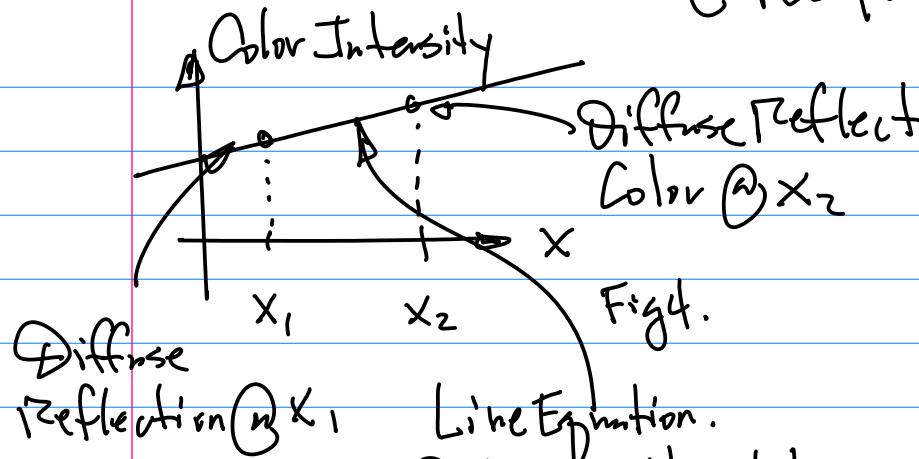
How To Compute Diffuse Reflection from 2 Known pts (Starting & Ending Pt of A Line) ?

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  $y$ .

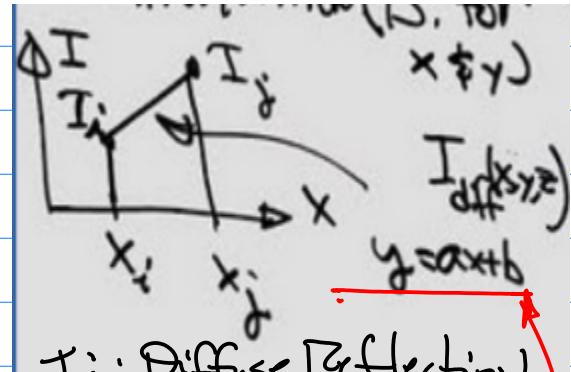
- (1) Contribution from  $x$  if color changing
- (2) Contribution of color changing from  $y$

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

Compeacto



Notation from the 45 (github) TRT.

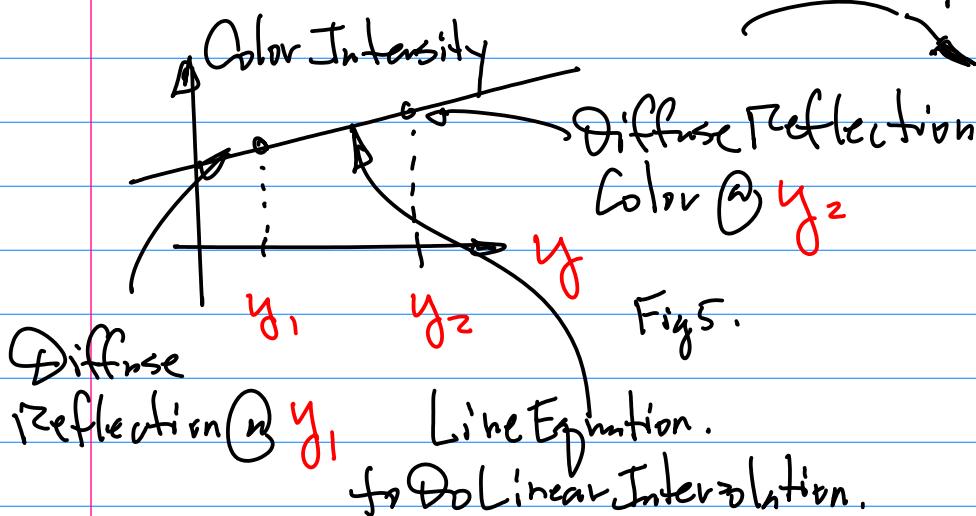


$I_i$ : Diffuse Reflection  
@  $x_i$

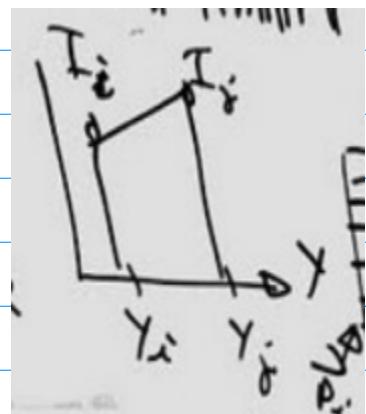
$I_j$ : --- @  $x_j$

$$\frac{x_2 - x_1}{x - x_1} = \frac{y_2 - y_1}{y - y_1} \dots (1)$$

Similarly for  $y$



From P.P.T for  $y$ .



For Fig 5. (for  $y$ )

$y \rightarrow x$  (Indep.).

$I_{diff}$  Color/Intensity  $\rightarrow y$

$$I_{diff}(y) = \frac{I_{diff}(y_i) - I_{diff}(y_j)}{y_i - y_j} y - \frac{y_j (I_{diff}(y_i) - I_{diff}(y_j)) + I_{diff}(y_i)}{y_i - y_j} \dots (2)$$

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

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

April 28.

CmPE240

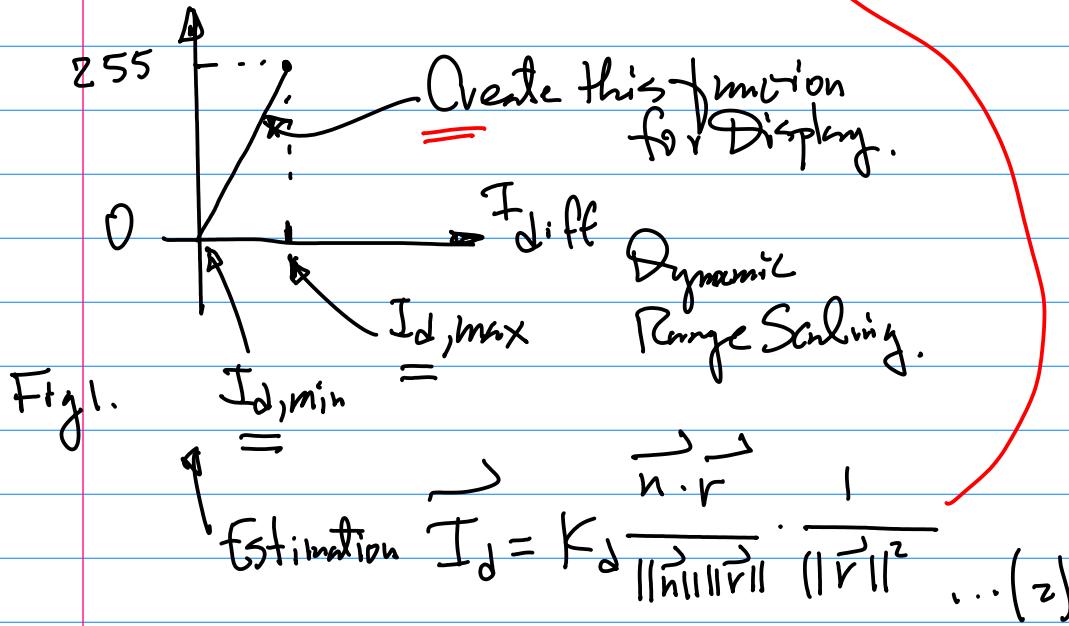
4b

Trajet SD GE. One May 1b (Sun)

Scaling up to the

Example: ① Offset ② Dynamic Range  
full

```
240 //compute color intensity
241 diffuse.r[i] = Kdr * angle[i] / pow(distance[i], 2)
242 diffuse.g[i] = Kdg * angle[i] / pow(distance[i], 2)
243 diffuse.b[i] = Kdb * angle[i] / pow(distance[i], 2)
244 }
```



$$\leq K_d \cdot \vec{n} \cdot \vec{r} = K_d (n_x, n_y, n_z) (r_x, r_y, r_z)$$

$$= K_d (1, 0, 1) (r_x, r_y, r_z) = K_d \cdot r_z \quad | \quad K_d = (0.8, 0, 0)$$

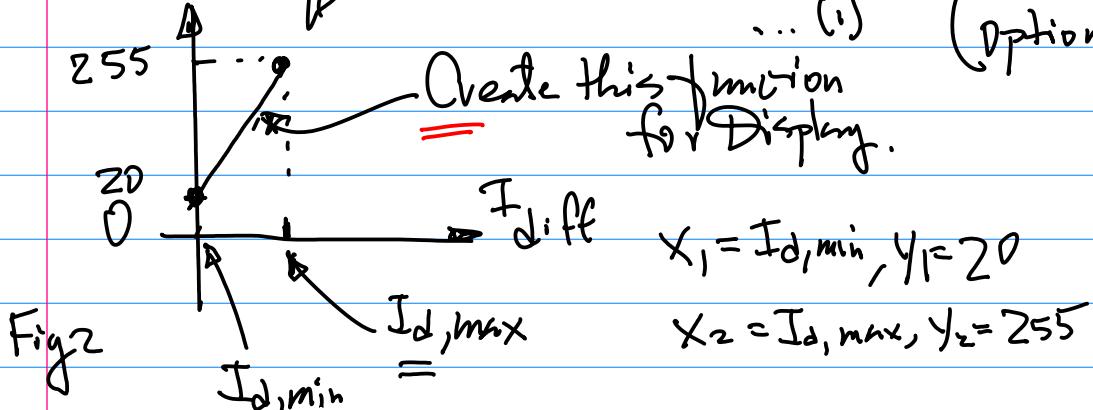
$$= (0.8r_z, 0, 0) \rightarrow I_{d,max}$$

$$I_{d,min} \approx 0$$

Add offset

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} \dots \text{linear Mapping function}$$

... (1) (optional)



May 2nd (mon)

```

509     for (int i=4; i<=5; i++) {
510         float r, g, b;
511         r = display_scaling*diffuse.r[i]+display_shifting;

```

↑ Scaling      ↑ offset

Example: Given  $\vec{P}_i(x_i, y_i) = \vec{P}_i''(100, 50)$   
on Fig 3.  $I_d(100, 50) = 100$ ;  
 $\vec{P}_j(x_j, y_j) = \vec{P}_j''(50, 25)$ ,  $I_d(50, 25) = 80$ ?

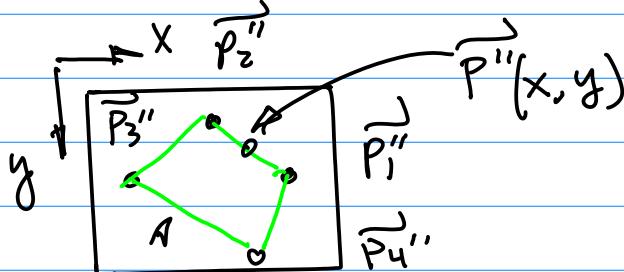


Fig 3.

Find Diffuse Reflection at the Boundary Point  $\vec{P}''(x, y)$   
 $\vec{P}_3''(75, 50)$ ,  $\vec{P}_1''(75, 30)$ ,  $\vec{P}_4''(50, 30)$ ,  $\vec{P}_2''(50, 50)$   
 $I_d(75, 30) = ?$

Note:  $I_d$  is always positive  
Then, Similarly, Calculate  
 $I_d$  from the Contribution  
of  $x$ ;

Once, Both Contribution of  
 $x$  and  $y$  are taken care of,  
then Calculate the Average  
which gives the diffuse

Reflection at this point.

Note: Check Sample code on  
the github, Implement diff.  
Reflection Computation On LFC.  
plot 4 "Small Squares".  
5x5

Sol The General Approach

$$\vec{I}_d(75, 30) = \frac{1}{2} \left( \vec{I}_d(x) + \vec{I}_d(y) \right) \quad x=75, \quad y=30,$$

$$= \frac{1}{2} \left( \vec{I}_d(75) + \vec{I}_d(30) \right) \dots (2)$$

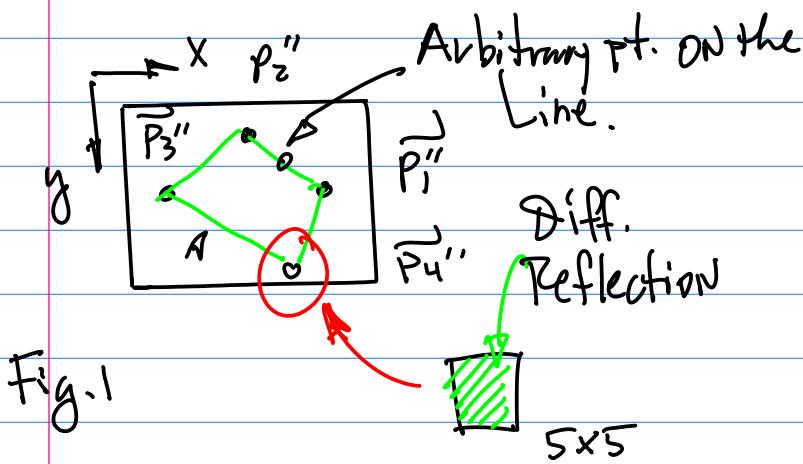
Suppose Solve for

$$\vec{I}_d(y) = I_d(y_i) = I_d(y_r) = 100, \\ I_d(y_j) = I_d(y_s) = 80$$

$$y_i = y_1 = 50, \quad y_j = y_2 = 25$$

$$\vec{I}_d(y) =$$

$$I_{diff}(y) = \frac{I_{diff}(y_i) - I_{diff}(y_j)}{y_i - y_j} y - \frac{y_j (I_{diff}(y_i) - I_{diff}(y_j)) + I_{diff}(y_j)}{y_i - y_j} \dots (2)$$



$$\begin{array}{r}
 x_3 x_{30} \dots x_1 x_0 \\
 \times a_{31} a_{30} \dots a_1 a_0 \\
 \hline
 a_0 x_3 a_0 x_{30} \dots a_0 x_0 \\
 a_1 x_3 a_1 x_{30} \dots a_1 x_0 \\
 \vdots \\
 \hline
 \dots a_{31} x_3 a_{30} x_{30} \dots a_0 x_0
 \end{array}$$

DDA Algorithm:  
(Digital Differential Algorithm)

To Deal with Finite Resolution of  
Display Devices.

Motivation: Reduce AND/OR  
Removal of multiplication

$$\left\{
 \begin{array}{l}
 \text{1 "GAPS" Problem } y = ax + b \\
 \text{2 multiplication Needed}
 \end{array}
 \right. \quad \dots (1)$$

$ax$

multiplication Unit  $\rightarrow n$  bit (32 bit)  
multiplication

H.A. (Half Adders)  $\leftarrow$  Basic Building Blocks of FA  
Carry Look-Ahead Circit.

$$ax, \quad a = a_{31} a_{30} \dots a_1 a_0$$

$$x = x_3 x_{30} \dots x_1 x_0$$

Example:  
Generate DDA Algorithm

$$\left\{
 \begin{array}{l}
 x_{k+1} = x_k + 1 \quad \dots (1) \\
 y_k = ax_k + b \quad \dots (2)
 \end{array}
 \right.$$

For  $k+1$ :

$$\begin{aligned}
 y_{k+1} &= ax_{k+1} + b \\
 &= a(x_k + 1) + b
 \end{aligned}$$

$$= ax_k + a + b \quad (\text{for Each } x \text{ increment})$$

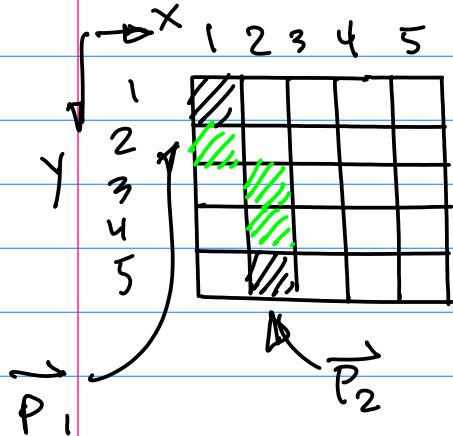
Note:  $y$  Increment by  $a$  ( $x$  increment)  
No multiplication Needed;

$$\left\{
 \begin{array}{l}
 x_{k+1} = x_k + 1 \quad \dots (za) \\
 y_{k+1} = y_k + a \quad \dots (zb)
 \end{array}
 \right.$$

49.

Suppose given 2 vertices

$$\vec{P}_1(1, 1), \vec{P}_2(2, 5)$$



use Eqn (2a), (2b) to plot  
a straight line.

$$\text{Step 1. Compute Slope } a \triangleq \frac{y_E - y_S}{x_E - x_S}$$

... (3)

from the given  
Condition:

$$a = \frac{5-1}{2-1} = 4.$$

Now, from Eqn (2a), (2b)

$$\text{for } k=1, x_1 = 1, y_1 = 1$$

$$\text{for } k=2, x_{k+1} = x_k + 1$$

$$x_2 = x_1 + 1 = 2$$

$$y_{k+1} = y_1 + a = 1 + 4 = 5$$

The "gap" is due to  $a > 1$ .

DDA:

Part I: If  $|a| < 1$ , then

$$y_{k+1} = y_k + a$$

Fig 2.

Part II: if  $|a| > 1$ , then

swap x and y, y as  
independent variable,  
x function.

$$y = ax + b$$

$$\frac{1}{a}y = x + \frac{b}{a}$$

$$\therefore x = \frac{1}{a}y - \frac{b}{a}$$

Then Apply the Incremental  
Algorithm.

For  $a=4$ , Let's make  $\frac{1}{a} = \frac{1}{4}$

$$y_1 = 1, x_1 = 1 \text{ (given)}$$

$$y_2 = y_1 + 1 = 1 + 1 = 2$$

$$x_2 = x_1 + \left(\frac{1}{a}\right)$$

$$= 1 + \frac{1}{4} \approx 1.25 = 1.$$

$$y_3 = y_2 + 1 = 2 + 1 = 3$$

$$x_3 = x_2 + \left(\frac{1}{a}\right) = x_1 + 2\left(\frac{1}{a}\right)$$

$$= 1 + \frac{2}{4} = 1.5 \approx 2$$

# CmpE240

SD

$$\text{for } y_4 = y_3 + 1 = 3 + 1 = 4$$

$$x_4 = x_3 + \left(\frac{1}{a}\right) = x_3 + \frac{3}{4}$$

$$= 1 + \frac{3}{4} = 1.75 \approx 2$$

Please Check my Sample  
Code "DDA-C" on github.

Homework: (Optional)

Use Bilinear Interpolation  
plus DDA Algorithm to

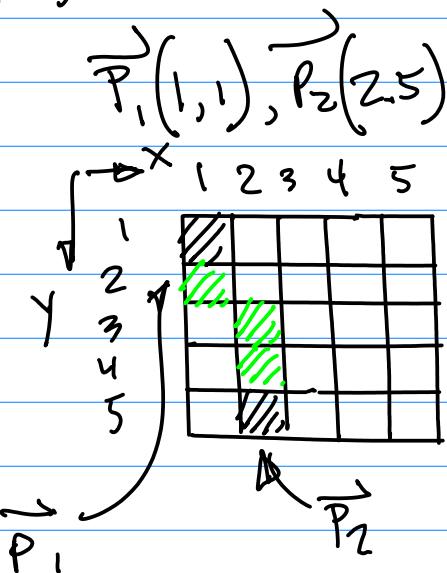
Compute/plot Diff. Reflection  
of At least one boundary line.

May 4th (Wed)

Example: Given  $P_1$  and  $P_2$ , and  
their Diffuse Reflection Value

$$I_{d1} = 100, I_{d2} = 90$$

( $T_{cd}$ )



Find Diffuse Reflection of the  
Boundary points ("green").

These points have to be

Computed By using DDA algorithm  
(See Example on PP.48-49)

First, find  $I_d(1, z) = ?$

Sol:

$$I_d(1, z) = \frac{1}{2} (I_{dx}(1, z) + I_{dy}(1, z))$$

Computation of  $I_{dy}(1, z)$  is  
given in Eqn(z) PP.47.

$$\frac{I_{di} - I_{dj}}{y_i - y_j} = \frac{I_{d1} - I_{d2}}{y_1 - y_2} = \frac{100 - 90}{1 - 2} = -10$$

$$= -10$$

$$\text{And } y = z$$

$$\frac{-y_j}{y_i - y_j} (I_{di} - I_{dj}) = -\frac{y_2}{y_1 - y_2} (100 - 90)$$

$$= -\frac{2}{1-2} \cdot 10 = -20$$

$$I_d(1, z) = -10 \cdot 2 - 20 + 90 = 50$$

Similarly,  $I_{dx}(1, z)$  can be

$I_{dx}(1,2)$  Can be Computed

Similarly,

$$\therefore I_d(1,2) = \frac{1}{2} (I_{dy}(1,2) + I_{dx}(1,2)) \\ = \frac{1}{2} (SD + I_{dx}(1,2))$$

github/finalisti/Cmpe240

"Diffuse" Sample is provided.

Note: Show + Tell the Prototype Board is Mandatory ! No Timely Showing / Completion will have Negative Impact to Entire Course Performance As Stated throughout the Semester.

Consider Diffuse Reflection Inside Boundary Lines.

Example

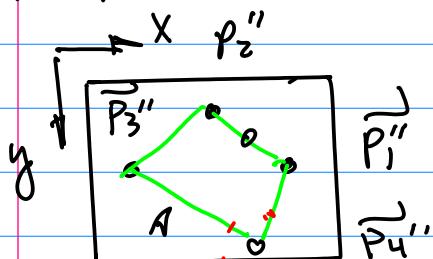


Fig 1.

Conditions:

a Scan the Display Conceptually from the top left corner left to Right, top to Bottom.

b Scanning Line "L" has 2 intersection points (Search Algorithm for Advancing index  $(i, j)$ ) to match up a point from the Set of pts from DDA)

c Diffuse Reflection on the Boundary has been Computed.

Compute "Green" Line (pts) inside the Boundary Line

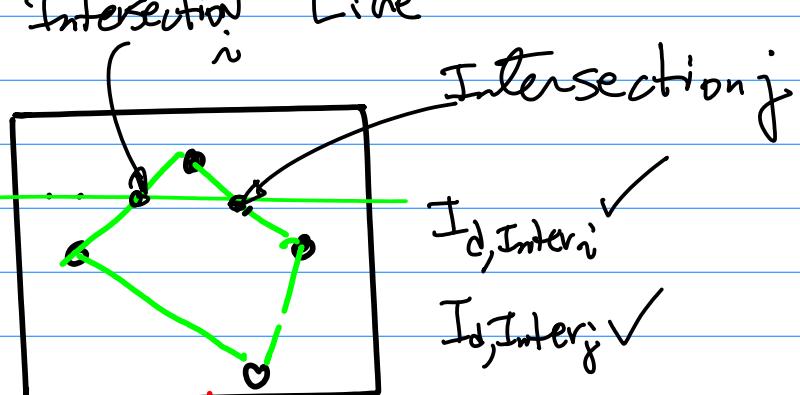


Fig 2

## Decorate Any Surfaces (planes)

Algorithm:

Step 1. Define 2D Pattern

$$\{\vec{P}_i(x_i, y_i) \mid i=1, 2, \dots, N\}$$

Step 2. Re-define 2D Pattern(s) in  $X_w - Y_w - Z_w$  Space;

Step 3. 3D Transform to Decorate on plane.

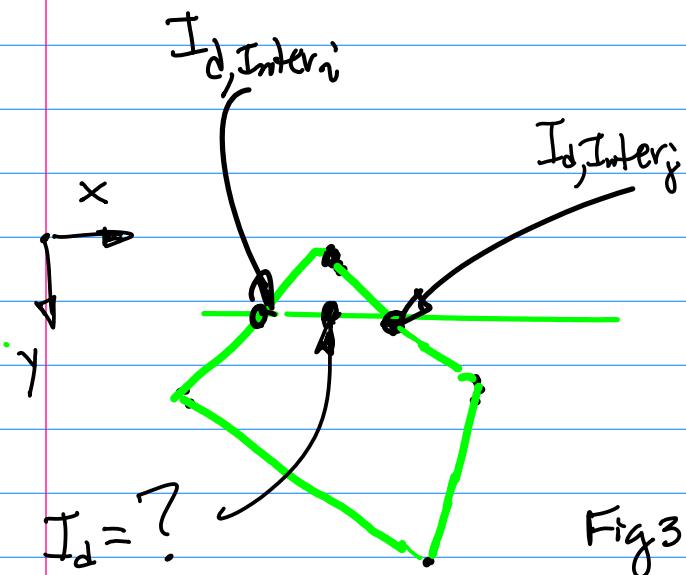
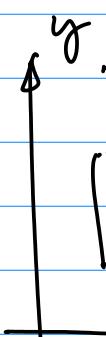
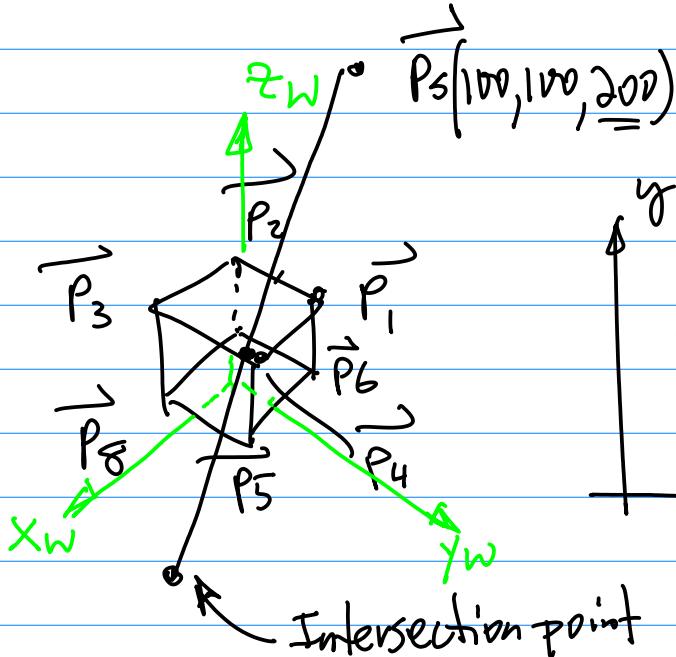


Fig.3

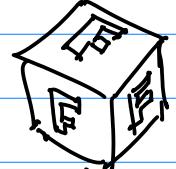
Interpolation Technique to find  
 $I_d = ?$

Interpolation w.r.t. X-variable.

Consider "Linear" Decoration



$$\vec{P}_i(x_i, y_i)$$



May. D (Mon).

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

Fig.4

Given 2D Pattern  
Above

To Decorate 3D plane (Surface)

$$\stackrel{a}{\equiv} \{\vec{P}_i(x_i, y_i) | i=1, 2, \dots, N\} \rightarrow \{\vec{P}'_i(x'_i, y'_i, z'_i) | i=1, 2, \dots, N\} \dots (0-1)$$

Decoration

$\stackrel{b}{=} \text{Conducted in } X_w Y_w Z_w.$

Meaning After the Decoration,

Transformation Pipeline will have  
to be Carried in order to display  
it.

$\stackrel{c}{=} \text{Let's consider the decoration}$   
for Surface 1, 2, and 3.

Other plane orientations can  
be generalized later. (Not  
in CMPE240)

Algorithm for Decoration.

Step 1. Define 2D pattern.

$$\{\vec{P}_i(x_i, y_i) | i=1, 2, \dots, N\}$$

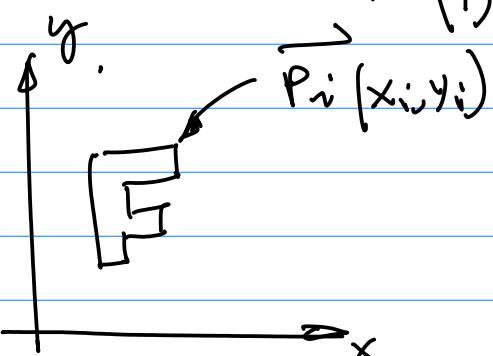


Fig 1.

Step 2. Redefine 2D

Pattern into 3D  $X_w Y_w Z_w$   
Space.

The Simplest way: Add  $z'_i$   
to Each point  $\vec{P}'_i(x'_i, y'_i)$ , such  
that New 3D point is  
Created:  
 $\vec{P}'_i(x'_i, y'_i, z'_i)$   
where

$$\begin{aligned} x'_i &= x_i, y'_i = y_i, \\ &\text{Add } z'_i; \text{ Or in Eqn (0-1)} \\ x'_i &= x_i, y'_i = y_i, z'_i = z_i \\ &\text{where } z'_i = 0 \dots (z) \end{aligned}$$

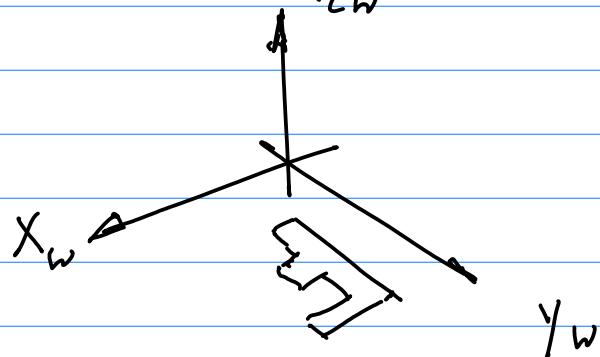


Fig 2

Step 3. Combination of 3D Transforms,

e.g. Translation, Rotation, Scaling,  
to move/attach the pattern to the  
desired plane.

$y_w - z_w : y_w - \text{Ind}, z_w - \text{Func}$

2D Pattern, 3D Pattern  
have to be matched

Ind - Ind, Function - Function

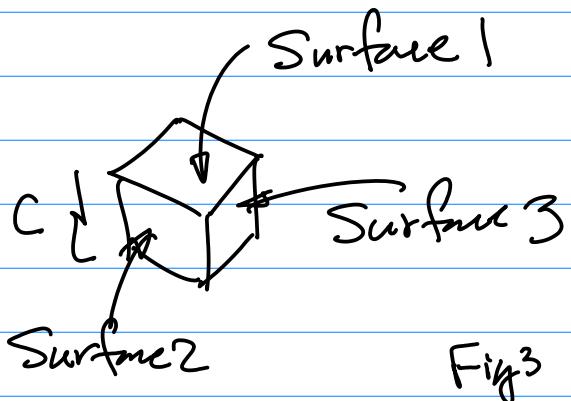


Fig 3

Surface 1 is the best plane  
to decorate with minimum  
operation.

$$\begin{cases} z_i^1 = C & \dots (3-1) \\ x_i^1 = x_i & \dots (3-2) \\ y_i^1 = y_i & \dots (3-3) \end{cases}$$

func	3D (After)	2D (Before)
Ind.	$x_i^1 = y_i$	(4-1)
	$y_i^1 = C$	(4-2)
	$z_i^1 = x_i$	(4-3)

Step 5, Surface 2

$y_w - \text{Ind}; z_w - \text{Func}$ .

$$\begin{cases} x_i^1 = C & \dots (5-1) \\ y_i^1 = x_i & \dots (5-2) \\ z_i^1 = y_i & \dots (5-3) \end{cases}$$

C/C++ Implementation.

Step 4. Decoration of Surface 3.

Note: (1) 3 planes in  $x_w - y_w - z_w$ .

$x_w - y_w$  plane:  $x_w - \text{Ind}, y_w - \text{Func}$   
(Z.H. Principal)

$z_w - x_w$  plane:  $z_w - \text{Ind}, x_w - \text{Func}$ .

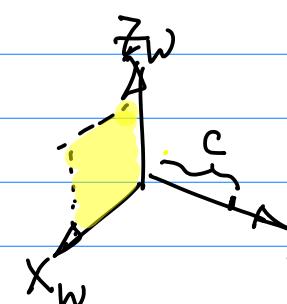
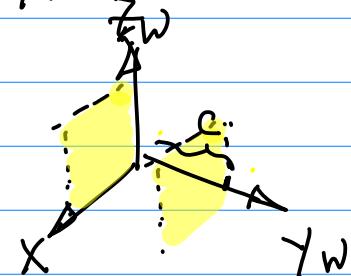


Fig 4.



Example: Given 2D Pattern  $\vec{P}_i(10, 15)$

May 12 (Wed)

A Cube in 3D with  $C=50$

Final Exam: May 19 Wed

12:15 - 14:30

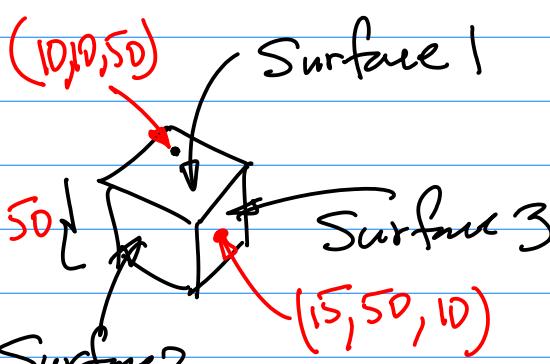
Decorate Surface 1, 2, 3 by  
 $\vec{P}_i(10, 10)$ .

May 17 (Monday) Review Final.

Sol: For Surface 1: (Top)

$$\begin{cases} x'_i = x_i \\ y'_i = y_i \\ z'_i = c \end{cases} \quad |_{c=50}$$

then,  $x'_i = 10, y'_i = 15, z'_i = 50$

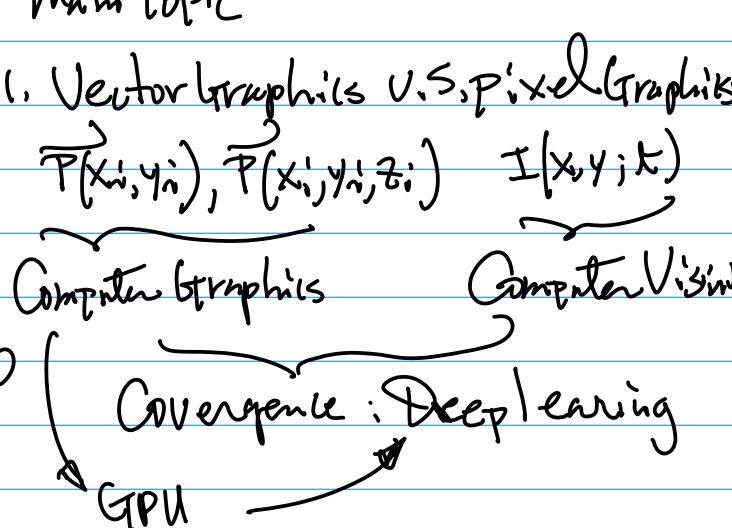


For Surface 3.

$$\begin{cases} x'_i = y_i \\ y'_i = 50 \\ z'_i = x_i \end{cases} \quad \text{Hence}$$

$z'_i = 10$  Ind.  $x'_i = 15$  Fanc.

$y'_i = 50$ .



2. 3D Graphics Engine

Vector Graphics + "Patch" of Vector Graphics

Lighting models + Triangulation Process.

"Triangles"

For Surface 2

$x'_i = 50$

$$\begin{cases} y'_i = 10 \text{ (Ind)} \\ z'_i = 15 \text{ (Fanc)} \end{cases}$$

3. Lighting model  
Condition: Shortest Distance Surfaces

Diffuse Reflection

Step 1. In  $X_w - Y_w - Z_w$  World Coordinate System, Compute  $I_d$  @ Corner Points of A patch;

# Computation

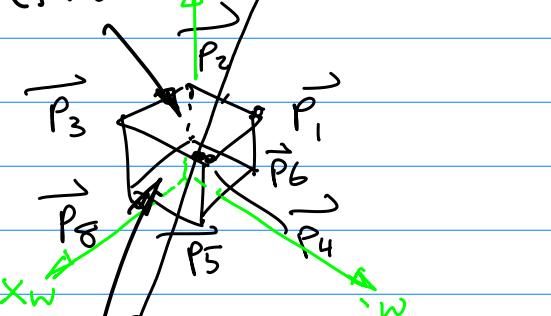
57

for Triangle Patch: 3 pts Diffuse  
Reflection

for A Cube Patch: 4 pts Diff. Reflection

(1)

Diffuse  
Reflection



(2) Shadow  
Computation

(3) Font, Initial here

Fig. 1 (PP. 41)

Step 2. After Transformation  
Pipeline, e.g. in 2D Space.  
(Virtual Coordinate System)

First, Compute Boundary  
Points, By DDA

then, Compute  $I_d$  on Each  
of these Boundary Points

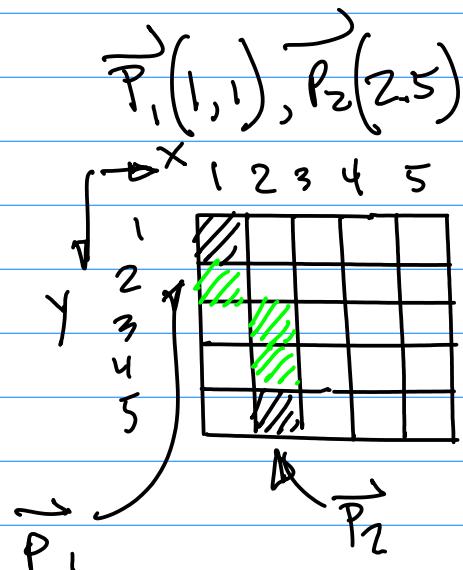


Fig. 2 (PP. 50)

Bi-Linear Interpolation  
for  $I_d$  on the Boundary pts

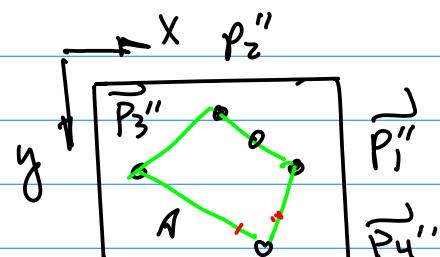
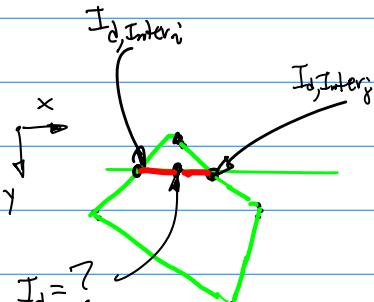


Fig. 3. (PP. 52)

Step 3. 2D Virtual Coordinate  
System (After  
Linear Interpolation)  
transformation Pipeline)

Fig. 4  
(PP. 53)



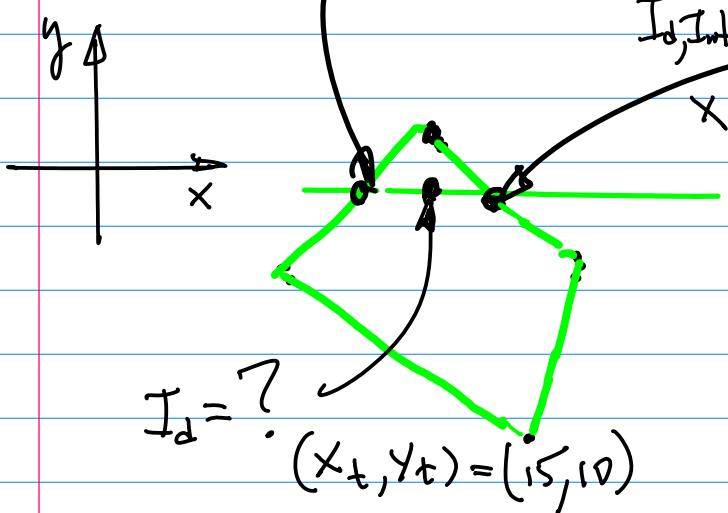
Example (for Step 3) Compute

$I_d$  for the interior pts.  $P(x_t, y_t) = \vec{P}(15, 10)$ ,  $I_d(15, 10) = ?$

Given the following condition:

$$I_{d, \text{Inter}_i}(x_i, y_i) = 80, x_i = 10, y_i = 10$$

$$I_{d, \text{Inter}_j}(x_j, y_j) = 90 \text{ (closer to } P_s, \text{ pt. Light Source)} \\ x_j = 20, y_j = 10; \text{ Note: The variable } x$$



Contributes to the Change of Diffuse Refl.

Therefore, Linear Interpolation with respect to X variable

$$I_d(15, 10) = \frac{80 + 90}{2} = 85$$

Hence, from Eqn (2),

$$I_{\text{diff}}(x_i) \quad I_{\text{diff}}(x_j) \quad x \quad x' \quad x_i \quad x'_j \quad x_j$$

$$I_{\text{diff}}(y) = \frac{I_{\text{diff}}(y_i) - I_{\text{diff}}(y_j)}{y_j - y_i} y - \frac{y_j (I_{\text{diff}}(y_i) - I_{\text{diff}}(y_j)) + I_{\text{diff}}(y_j)}{y_j - y_i} \quad (2)$$

Note: Derivation from:  $\frac{x - x_2}{x_1 - x_2} \cdot \frac{y - y_2}{y_1 - y_2}$

