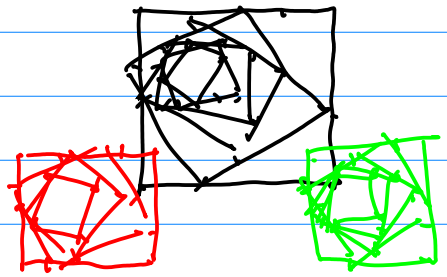
a A set of Squares Rotation with One Same Color

b Change location/Color/size

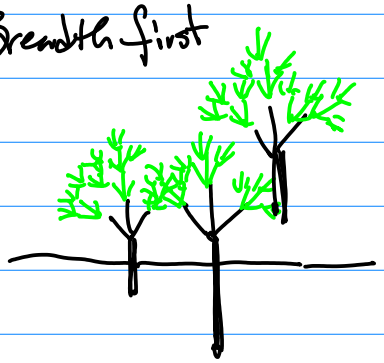
c



Keep the patterns, please don't erase them!

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

a Breadth first



b $L \geq 7$ c Create A patch of Forest Changing Location/size

Note: Please Don't Erase the Drawing Till All the trees are Constructed.

Note: Report Writting.

3D G.E. (Graphics Engine)

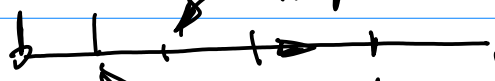
GPU Architecture

Jtag

Cortex M3

GPP SPRs, SPI SPRs.

Solid Obj HL, HS. Removal



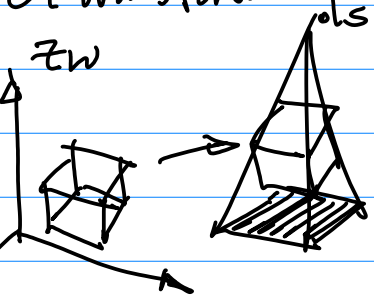
3D Concepts + Algorithm

(2) Wireframe

Shade

State-of-the-Art

(1) World-To-Viewer Transform



Lighting model. Diffuse Reflection

March 15 (monday)

Note: 1° midterm A week from this Wednesday (march 24th)

2° Smartphone (CAM Capability) A piece of Paper to the Class. Test the Environment

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

Virtual Camera Location

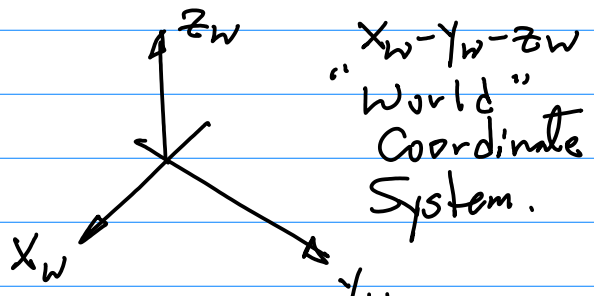
1° 

Fig. 1

Right-Hand System,

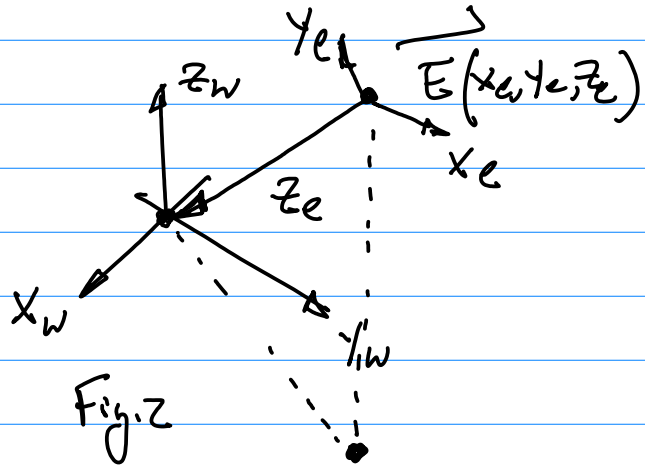


Fig. 2

Projection of $\overrightarrow{E}(x_e, y_e, z_e)$ on to x_w-y_w plane in the x_w-y_w-z_w world Coordinate System

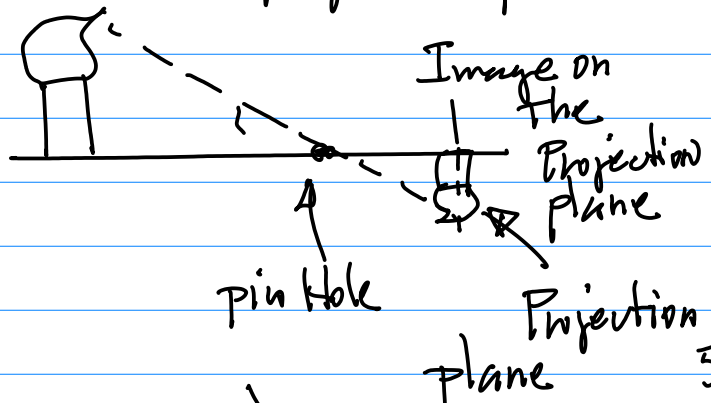
2° Virtual Camera

a. Virtual Enclosure

b. Optic Lens (Pin Hole)

$\phi_d \simeq \phi$ 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 $\overrightarrow{E}(x_e, y_e, z_e)$ as

4. Viewer Coordinate System

Viewer ~ Virtual Camera

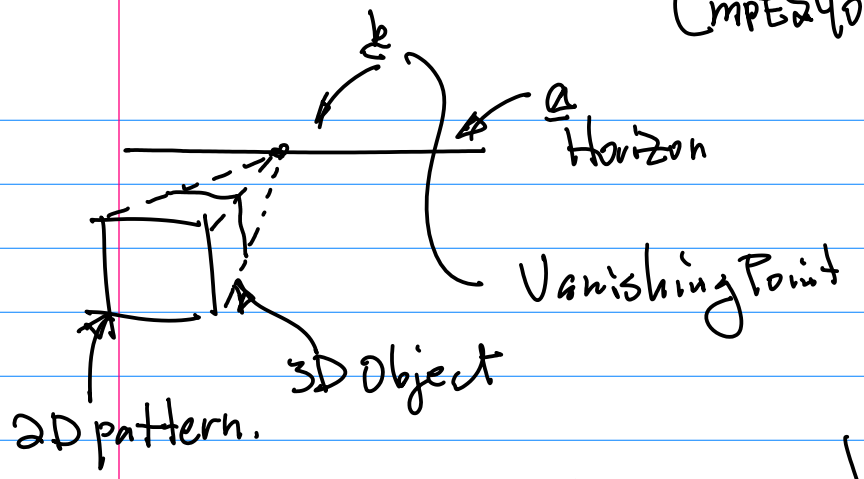
x_e-y_e-z_e Viewer Coordinate System. "Eye"

a. $\overrightarrow{E}(x_e, y_e, z_e)$ As the origin.

b. Left-Hand System.

c. z_e-axis points to the origin.

5: Perspective Projection



3D Object formed on 2D plane

7. Transformation Pipeline

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

W2V (Work_2_Viewer) Transform

Perspective Projection.

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

{ Objects in E } in X_w-Y_w-Z_w → Viewer Coordinate System X_e-Y_e-Z_e

Projection Plane form Perspective Projection

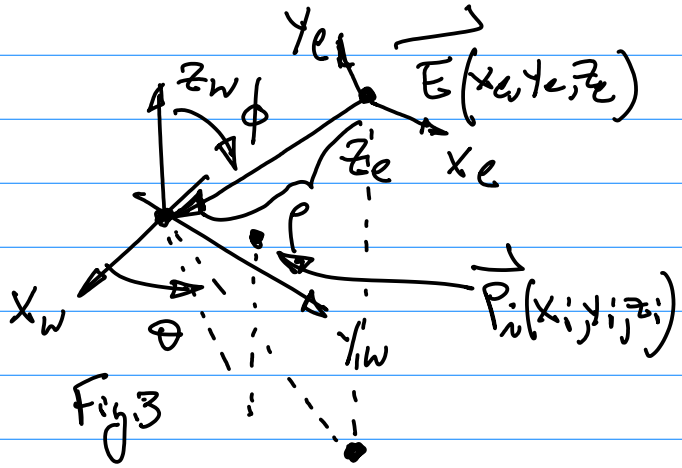


Fig 3

3 parameters:

a ≙ theta θ Angle

b ≙ Vector $\overrightarrow{E}$ to $\overrightarrow{O}$, "rohi", Distance ρ

Mathematical Formulation

1° Formula: Transformation Pipeline

2° Foundation

a ≙ Translation

b ≙ Rotations { ~ wrt X_w .. Y_w .. Z_w And wrt Arbitrary axis

a ≙ World_to_Viewer Transform

b ≙ Perspective Projection

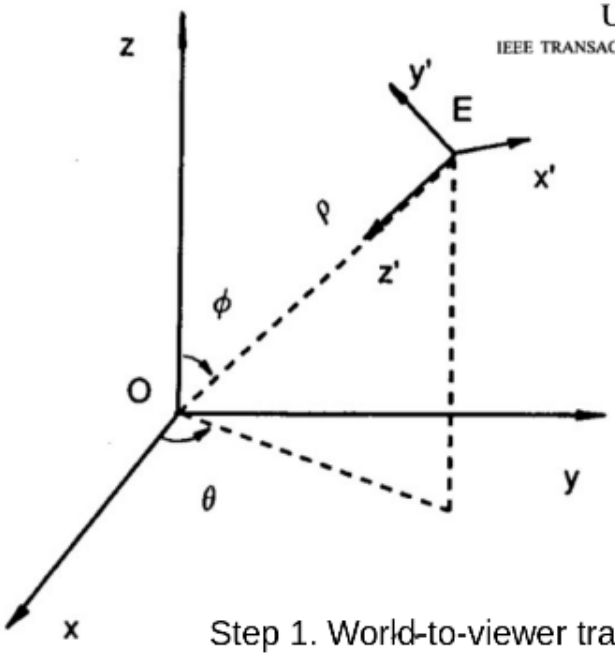
$$\rho \triangleq \sqrt{x_e^2 + y_e^2 + z_e^2} \quad \cdots (1)$$

≙ Angle Z_w and Z_e φ.

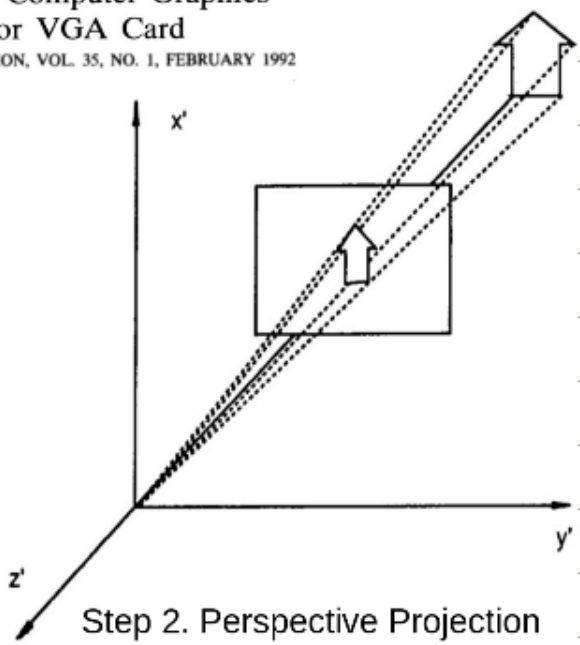
World_2_Viewer

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

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

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

March 17 (Wed)
Topics: 1° Hardware Architecture 2° Software SPRs Init & Config.

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

GPIO(GPP) SPI
A set of b ~
- System Configuration (Per. Cnt. PWR CLK multiplexing)
- * Configuration of the Peripheral Cnt.

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

$$\begin{pmatrix} x_i' \\ y_i' \\ z_i' \\ 1 \end{pmatrix} = T \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \quad \cdots (2)$$

After: x_e-y_e-z_e

Before: x_w-y_w-z_w

World to Viewer Transform.

Compute to Rotation Matrix

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

$$\begin{pmatrix} \sin\theta & \cos\theta & 0 & 0 \\ \cos\theta & \sin\theta & \cdots & \cdots \\ \cos\theta & \cos\theta & \cdots & \cdots \\ 0 & 0 & 0 & 1 \end{pmatrix}$$ Part I. "θ"

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ \cos\phi & \cos\phi & \sin\phi & \cdots \\ \sin\phi & \sin\phi & \cos\phi & \cdots \\ 0 & 0 & 0 & 1 \end{pmatrix}$$ Part II "φ"

$$\begin{pmatrix} - & & & \\ - & - & & \\ - & - & - & P \\ 0 & 0 & 0 & 1 \end{pmatrix}$$ Part III "—"

roh =

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.

Bitwise "OR"

1 = (1<<10)

& = ~(3<<20)

"AND" "11" Negation "00"

1 = (1<<20)

"OR" "01"

Clear 2 Bits

Set 2 bits as "01"

```
&=~((3<<18)|(3<<16)|(3<<14));
```

"AND" Nega. "11" Total 6 bits <<18

Clear 6 bits

```
|= ((2<<18)|(2<<16)|(2<<14)|)
```

"OR" Set "10"

Note: CR0 CR1

LPC_SSP1 -> CR0

Control Register

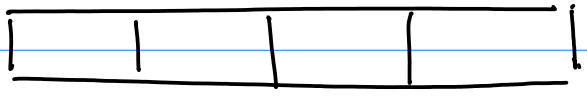


Table 571

Tech Spec.

1° 8 bit Transfer.

CR0[3:0] = 0111 = 0X7

2° SPI

CR0[5:4] = 00 (SPI)

3° Clock f_{SPI}

a CR0[15:8]

8 bits 255 = 2^8

b

$$f_{SPI} = \frac{PCLK}{(SCR+1)\,CPSDVSR} \quad \cdots (1)$$

SCR: [0, 255]

Mauch 22nd.

Today's Topics:

1° Midterm Review

2° SPRs CR0, CR1 for SPI I/F

Ref:

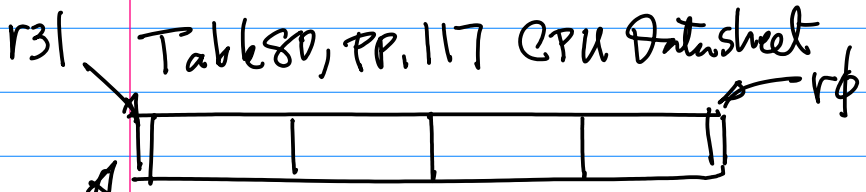
1° CPU Datasheet. PP431-433.

2° Sample code SPIinit (Draw a Line)

Example: SSP.C Source Code Walk-Through. 151-208

Line 162-165 PINSEL0

Tab680, PP. 117 CPU Datasheet



PINSEL0[31:30] = 10

Tab681. PINSEL1

= 0x2

10 from CPU Datasheet

PINSEL1[1:0], PINSEL1[3:2], PINSEL4[5:4] → SSP1

SSEL0, MOSI0, MISO0

Line 173

CR = 0x0707 → Tech Spec.

.... 0000 0111 0000 0111

16 upper Bits all zeros.

From Datasheet PP431

8 bit Transfer

CR0[15:8] SCR

CR[5:4] = 00 SPI

= Clock

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

CR0

CR0[15:8] = 0x7

CR1[2] = 0 for "Master"

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

To find f_{SPI}.

$2^8 = 256$, SCR G [0, 255]

PP. 433. CPSDVSR G [2, 254]

Midterm Review.

1° Video ON, Mandatory.

2 = Submission to CANVAS

15 min. File Uploading

No Late Submission

After the Deadline paper will be disqualified

2 If CANVAS Disrupted, then E-mail Submission

2 file in "zip"

FirstName + 4 Digits + CMPE240 mid.zip

SID

2° 30 Questions ±

Hardware { CPU Block Diagram

memory map

SFRs.

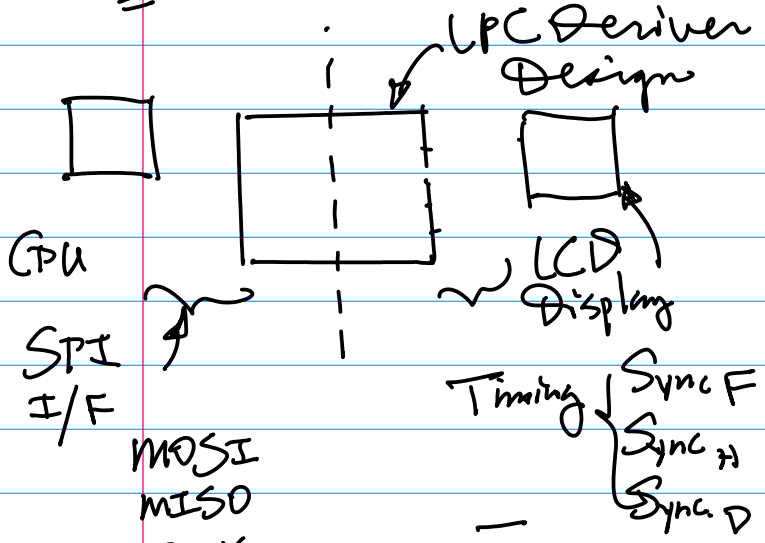
CKT, SCH Design

Software: Coding

SPR → Binary Pattern for Init & Config Debugging Purpose → Tech Spec.
Coding

Algorithm:
2D vector Graphics.
G.E.

a LPC CPU LPC Driver.



b Virtual V.S. physical Display Transform.

c $\vec{P} = \vec{P_i} + x(\vec{P_{i+1}} - \vec{P_i})$

Screen Saver. { i. Rotation ! No W/O Rotation matrix
ii 2D Transforms

Composition of 2D Transform. { R_{3x3}, T_{3x3} → Tree.

Preprocess f R_{3x3} 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 Re-design.

Calculator is allowed;

CMPE240

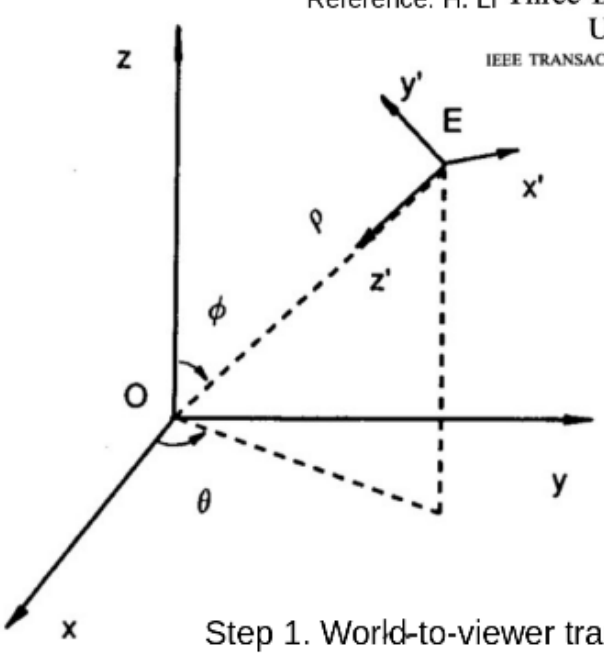
April 5 (Monday)

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

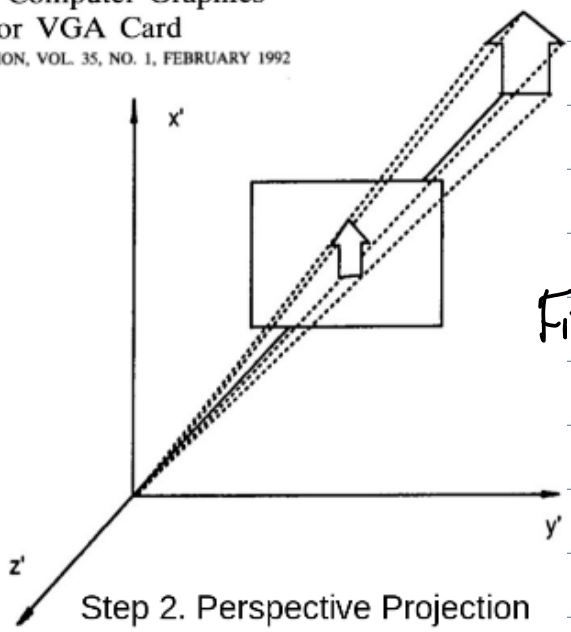
3D Transformation Pipeline Technique

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



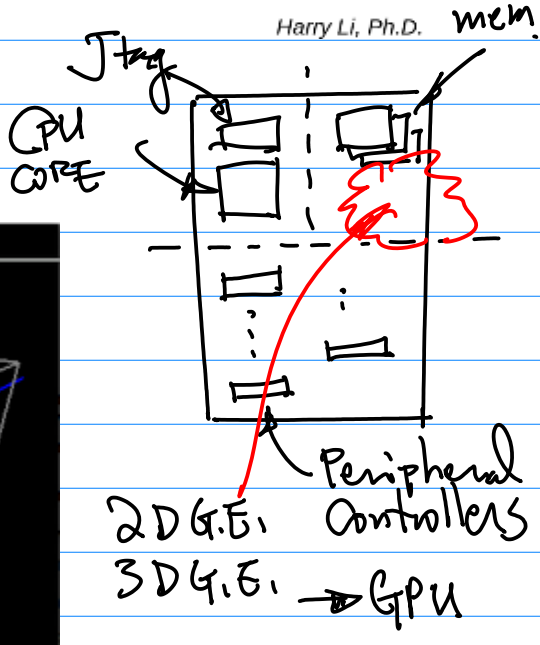
Step 2. Perspective Projection

Fig1.

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

$$x_p = x_e\left(\frac{D}{z_e}\right) \qquad y_p = y_e\left(\frac{D}{z_e}\right) \quad \cdots(2)$$

Today's Topics: 3D G.E.



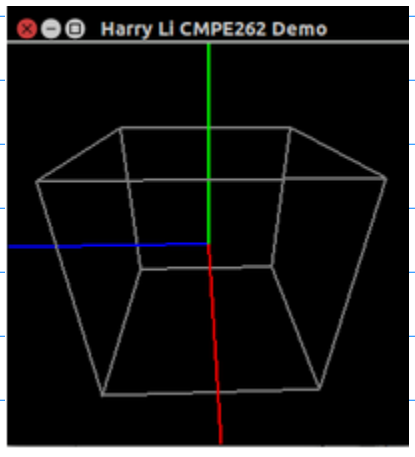


Fig2a

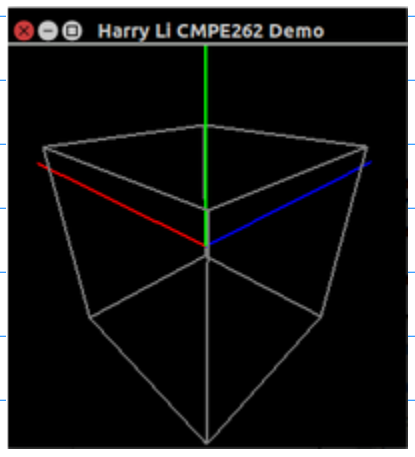


Fig2b

from github. Kong "111a," "111b," ...

Note: 2D G.E. { Vector Graphics
Transformation
Primitive Graphics { a ≙ D.D.A.
b ≙ line
c ≙ Arc/circle etc.

Transistor/Gate level.

Example: 3D Wireframe model

First, X_w-Y_w-Z_w World Coordinate System.

a ≙ Right System, r-g-b for X_w, Y_w, Z_w axis

b ≙ Transformation Pipeline

1st ≙ World-2-Viewer

$T_{4\times4}: (X_w, Y_w, Z_w) \to (X_v, Y_v, Z_v)$ ...(3)

2nd ≙ Perspective Projection

$P: (X_v, Y_v, Z_v) \to (X_p, Y_p)$ ...(3b)

Second. Design of Dataset, e.g.

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 $\vec{P}_0(X_0, Y_0, Z_0) = (0,0,0)$ ...(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) ← Atul2(m)

Homework: Draw r-g-b axis on your LPC Display.

Bring your Program/Board to the Next Class.

Step2. World_2_Viewer

$T_{4\times4}: (X_w, Y_w, Z_w) \in \mathbb{R}^3 \to (X_v, Y_v, Z_v) \in \mathbb{R}^3$

$$\begin{pmatrix} X_v \\ Y_v \\ Z_v \\ 1 \end{pmatrix} = T_{4\times4} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \quad \text{...(5)}$$

"After" "Before"

Now, find the X_w-axis in Viewer Coordinate

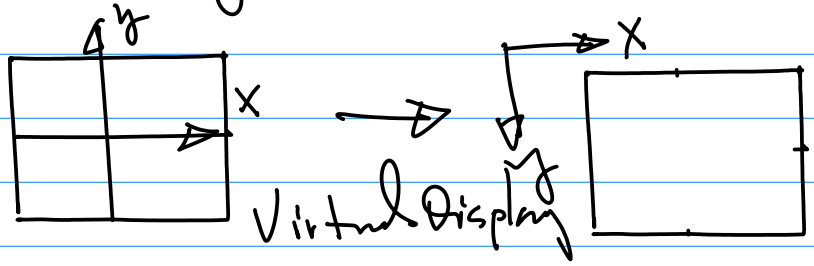
CmPE340

Step3. Perspective Projection

$P: (x_e, y_e, z_e) \in \mathbb{R}^3 \to (x_i'', y_i'') \in \mathbb{R}^2$

(x_i'', y_i''): Coordinate on your LPC1769 Display Device.

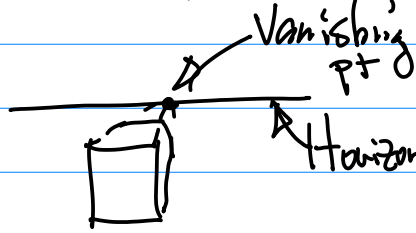
Note:
(x_i'', y_i'') is defined on 2D Virtual Display Coordinate
Need to perform Virtual to Physical Transformation in order to plot your result.



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".



$$\begin{cases} x_p = \frac{D}{z_v} x_v \quad \cdots (a1) \\ y_p = \frac{D}{z_v} \cdot y_v \quad \cdots (a2) \end{cases} \quad (x_v, y_v, z_v) \in \mathbb{R}^3$$

D: Virtual Cam focal length

In Homework, D = 2D

(x_p, y_p) On your 2D Display

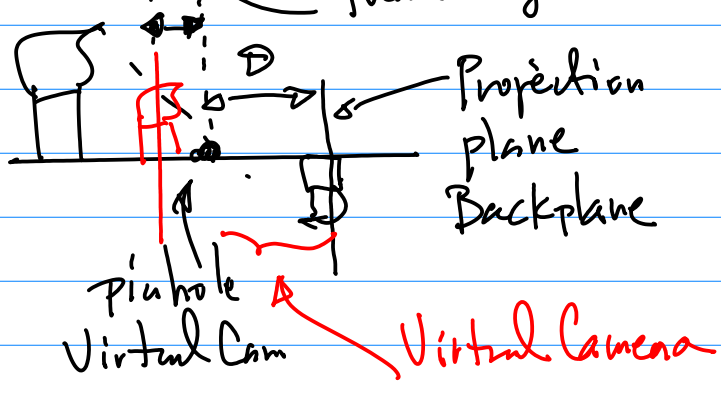
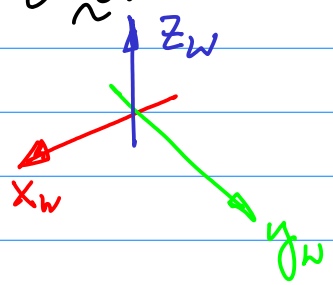


Fig1.



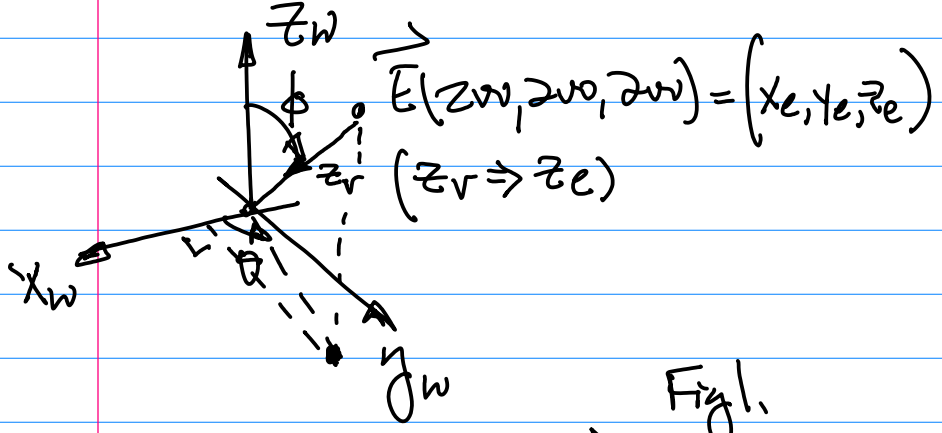
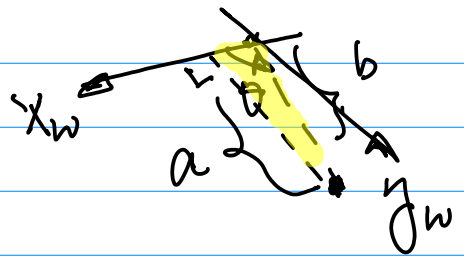


Fig 1.



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

b Vector Passing through the intersection pt on X_w-Y_w, perpendicular to X_w axis.

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 T 4x4 matrix, So World-2-Viewer Transform Can be Performed.

For Angle θ (Theta)

Sinθ and Cosθ

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

$$a = y_e = 200$$

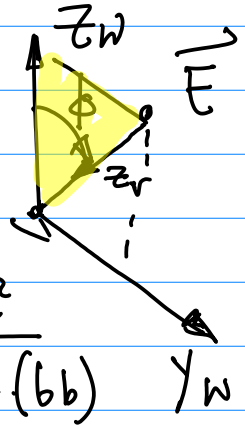
$$b = \sqrt{x_e^2 + y_e^2} = 200\sqrt{2}$$

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

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

for ϕ (phi)

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



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

"rho"

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

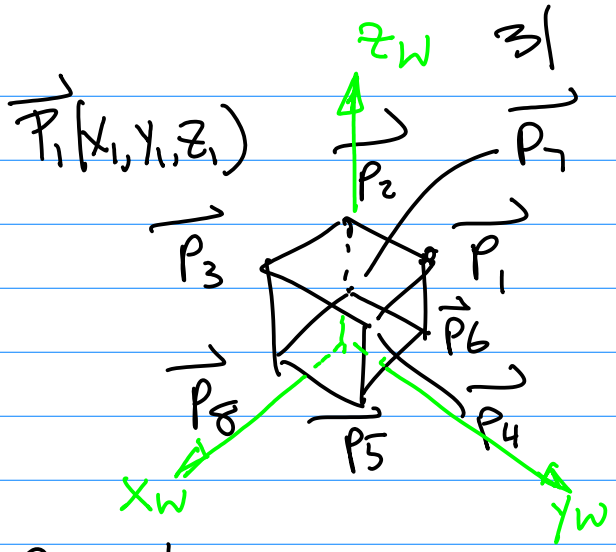
April 7 (Wed)

Note: X_v-Y_v-Z_v Left Hand System

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

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

CMPE240

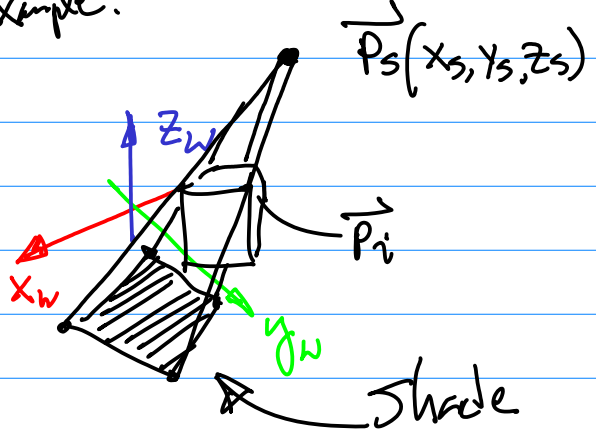


Similarly

$$\sin\phi = \sqrt{X_C^2 + Y_C^2}/\rho = \frac{20\sqrt{2}}{20\sqrt{3}} = \sqrt{2/3}$$

Consider "Shade" Calculation.

Example:



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: $P_1, P_2 \ldots$, are arranged Counter clockwise

1. X_W-Y_W-Z_W.

April 12 (Monday)

Note: 1° Homework Submission (E-mail)

X_W-Y_W-Z_W Drawing.

Source code — photo

(Exported Project) By Wednesday.

2° Bring your Prototype Board to Each Session for Inspection, Show & Tell.

$\vec{P_5}(100,100,10)$, $\vec{P_6}(0,100,10)$

$\vec{P_7}(0,0,10)$, $\vec{P_8}(100,0,10)$

Discussion on 3D Shade Computation

2. Point Light Source

$\vec{P_S}(100,100,200)$

Conditions for this discussion

1° Define $\{\vec{P_i}(x_i,y_i,z_i) \mid i = 1,2,\cdots,8\}$

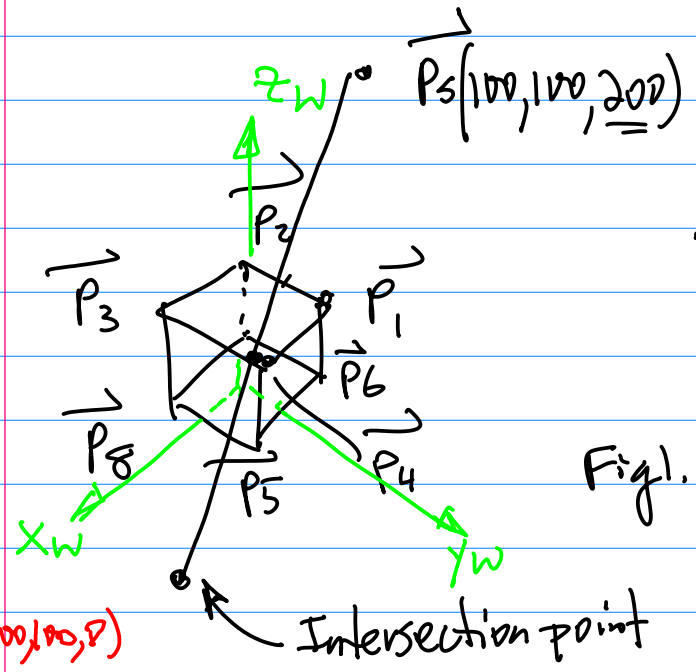


Fig1.

3. Ray Equation, Connecting $\vec{P_3}$ to $\vec{P_4}$,

$$\vec{R} = \vec{P_5} + \lambda(\vec{P_i} - \vec{P_5}) \quad \dots (1)$$

$$\vec{P_i} = \vec{P_4}$$

OR,

$$\vec{P} = \vec{P_5} + \lambda(\vec{P_i} - \vec{P_5}) \quad \dots (1b)$$

Find intersection point on x_w-y_w plane.

4. plane Equation

4a Normal Vector $\vec{n}(n_x, n_y, n_z)$ perpendicular to the given plane, x_w-y_w plane

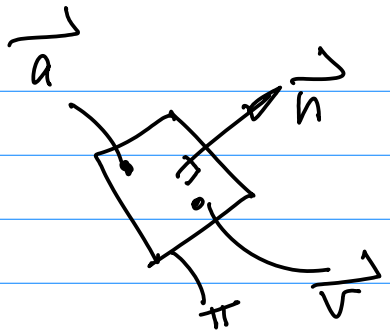


Fig.2

4b. plane π.

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

An arbitrary point $\vec{v}(v_x, v_y, v_z)$ on π

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

OR

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

$\vec{n}$: normal Vector; $(\vec{v} - \vec{a})$: line Segment

5. Intersection pt.

$$\begin{cases} \vec{R} = \vec{P_5} + \lambda(\vec{P_i} - \vec{P_5}) & \dots (3a) \\ \vec{n} \cdot (\vec{v} - \vec{a}) = 0 & \dots (3b) \end{cases}$$

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)

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

$$\vec{n} \cdot (\vec{R} - \vec{a})\Big|_{\vec{R} = \vec{P_s} + \lambda(\vec{P_i} - \vec{P_s})} = 0$$

$$\vec{n} \cdot \left(\vec{P_s} + \lambda(\vec{P_i} - \vec{P_s}) - \vec{a}\right) = 0$$

$$\vec{n} \cdot \vec{P_s} + \lambda \vec{n} \cdot (\vec{P_i} - \vec{P_s}) - \vec{n} \cdot \vec{a} = 0$$

$$\lambda \vec{n} \cdot (\vec{P_i} - \vec{P_s}) = \vec{n} \cdot \vec{a} - \vec{n} \cdot \vec{P_s}$$

$$\lambda = \frac{\vec{n} \cdot (\vec{a} - \vec{P_s})}{\vec{n} \cdot (\vec{P_i} - \vec{P_s})} \quad \cdots (4)$$

---

from Eqn (3a), Ray Eqn.

With λ we can find the intersection Point.

In C/C++ Coding.

$$\lambda = \frac{\vec{n} \cdot (\vec{a} - \vec{P_s})}{\vec{n} \cdot (\vec{P_i} - \vec{P_s})} = \frac{(n_x, n_y, n_z) \cdot (a_x - x_s, a_y - y_s, a_z - z_s)}{(n_x, n_y, n_z) \cdot (x_i - x_s, y_i - y_s, z_i - z_s)}$$

$$= \frac{n_x(a_x - x_s) + n_y(a_y - y_s) + n_z(a_z - z_s)}{n_x(x_i - x_s) + n_y(y_i - y_s) + n_z(z_i - z_s)} \quad \cdots (4a)$$

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 Eqn.

$$\vec{R} = \vec{P_s} + \lambda(\vec{P_4} - \vec{P_s})\Big|_{\lambda = 20/q}$$

$$= (100, 100, 200) + \frac{20}{q}(x_4 - x_s, y_4 - y_s, z_4 - z_s)$$

$$= (100, 100, 200) + \frac{20}{q}(100-100, 100-100, 110-200)$$

$$= (100, 100, 200) + \frac{20}{q}\cdot(0, 0, -90)$$

$$= (100, 100, 200) + (0, 0, -\frac{20}{q} \times 90)$$

$$= (100, 100, 200) + (0, 0, -200) = (100, 100, 0)$$

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

2° Compute/Implement this Algorithm, to Calculate (Hand) Each of Every 4 pts of top Surface of the Cube.

Note: Homework Submission Extended to 18th Sunday, 11:59pm.

a e-mail ; Subject: FirstName + SID(4Digits) + CmpE240 + HW2

April 14 (Th).

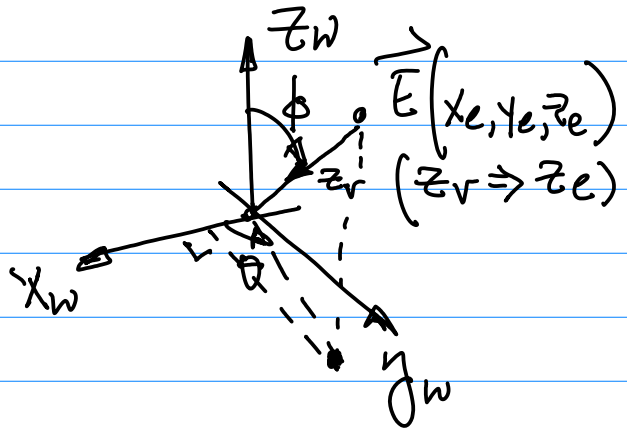
See the figure Next page

```
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;   // y-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;
```

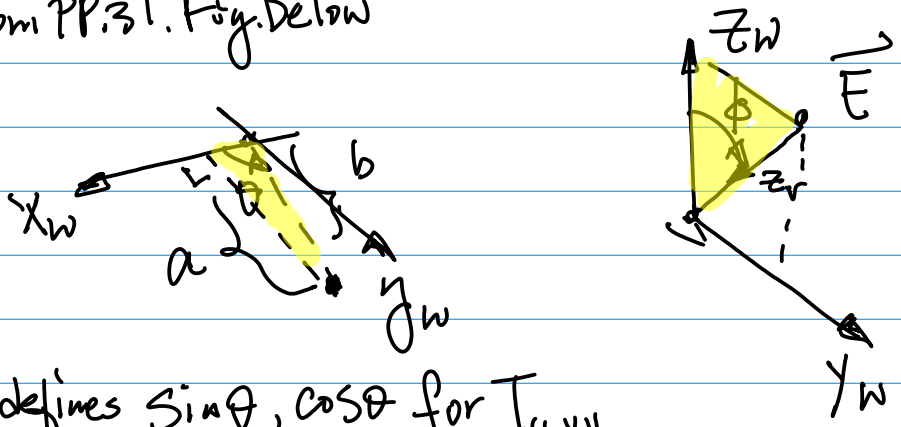
Declaration of Each Coordinate System

```
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;   // y-axis
```

From PP.31. Fig. Below

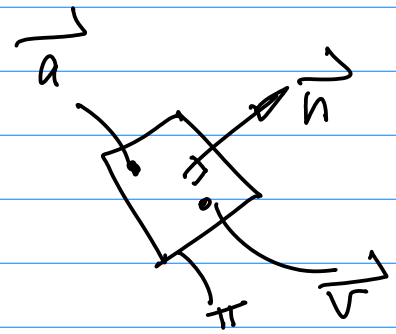


We defines $\sin\theta$, $\cos\theta$ for $T_{4\times4}$

World-To-Viewer Transform.

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

From PP.32



Define plane Equation By

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

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

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

```
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 $T_{4\times4}$

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

from TP,29, $T_{4\times4}$

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

$$\begin{pmatrix} X_v \\ Y_v \\ Z_v \\ 1 \end{pmatrix} = T_{4\times4} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

"After" "Before"

World-to-Viewer Transform

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 Transformation pipeline than Done!

Then, Computer Virtual Display (2D) to physical Display! Then plot the points!

CmpE240

Diffuse Reflection.

Background/Basic Concepts

1. Objective: To generate realistic Looking 3D Graphics

Lighting Models (After Trans. Pipeline)

$$\vec{I}(x,y) = (r(x,y), g(x,y), b(x,y)) \quad \cdots (1)$$

Primitive Colors.

Graphics (Vector) Graphics

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 2 b

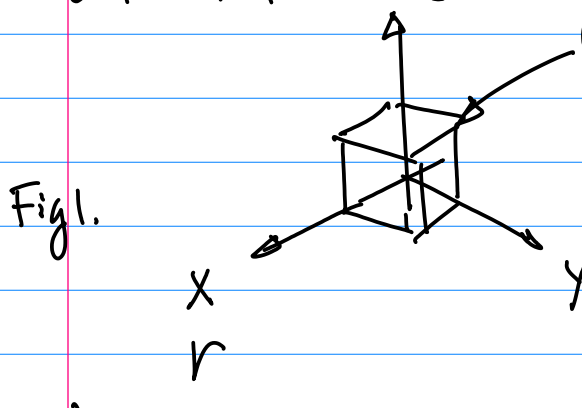


Fig1.

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) Brightest White

38

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

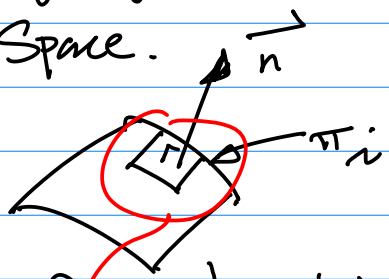


Fig2a

Curved surface → Linear Patches π_i

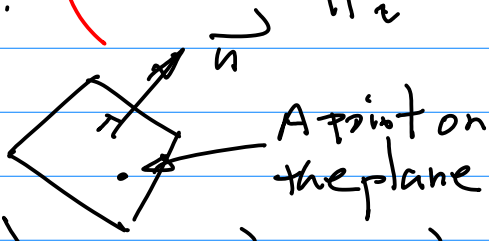


Fig2b

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

3D Space is X_w-Y_w-Z_w

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

Diffuse Reflection (Ref. on github):

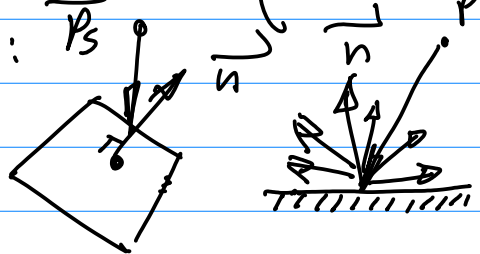


Fig2c

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

4. Reflectivity for all 3 Different light models (I_1, I_2, I_3)

Define $\vec{R} = (r_r, r_g, r_b)$ ... (3)

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

$r_r = r_g = r_b = 0$ Black

$r_r = r_g = r_b = 1$ white

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

Physical Characteristics of the given Surface

Now, Specular Reflection

I_2: Reflection Produces high light, it is a function of $\vec{E}$ position.

I_3: Ambient light, from indirect light Source, can be Simulated by adding Constant values to r, g, b.

Example: Diffuse Reflection model.

Step 1. $\vec{I}(x,y,z)$ Color on a pt. of a Surface in 3D Space.

$\vec{I}(x,y,z) = (\vec{I_r}(x,y,z), \vec{I_g}(x,y,z), \vec{I_b}(x,y,z))$

Simplify the discussion by focusing on One primitive Color

$\vec{I_r}(x,y,z)$

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

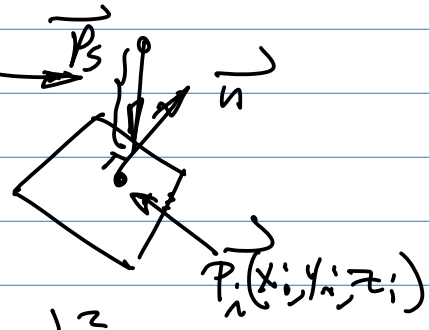
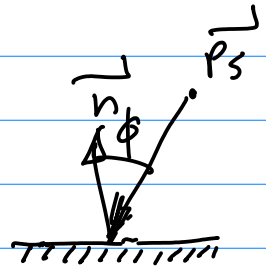


Fig 3

$\|\vec{P_s} - \vec{P_i}\|^2 = \|\vec{r}\|^2$ ... (4)

Ray Equation $\vec{r}$

$\vec{I_r}(x,y,z) \sim \frac{1}{\|r\|^2}$ ... (5)



ϕ. ph: Angle of the incoming light

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

$\phi = \pi/2$, photons miss the pt. on the surface → Normalize it → "0"

Cosϕ Rule. Dot product

$$\vec{n} \cdot (-\vec{P_s}) = \|\vec{n}\| \, \|-\vec{P_s}\| \cos\phi \quad \ldots (5)$$

$$\cos\phi = \frac{\vec{n} \cdot (-\vec{P_s})}{\|\vec{n}\| \, \|-\vec{P_s}\|} = -\frac{\vec{n} \cdot \vec{P_s}}{\|\vec{n}\| \, \|\vec{P_s}\|} \quad \ldots (6)$$

From Eqn (5), (9), (10)

define $-\vec{P_s}$ as Ray Vector

$$\vec{I_r}(x,y,z) = K_r \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} \frac{1}{\|r\|^2} \quad \ldots (11)$$

$$\vec{r} \triangleq \vec{P_i} - \vec{P_s} \quad \ldots (7)$$

Hence

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

where

$$\|\vec{r}\|^2 = \|\vec{P_i} - \vec{P_s}\|^2 = (x_i - x_s)^2 + (y_i - y_s)^2 + (z_i - z_s)^2$$

$$\vec{I_r}(x,y,z) \sim \cos\phi \left(= \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{v}\|} \right) \quad \ldots (9)$$

$$\vec{I_r}(x,y,z) \sim K_r \quad \ldots (10)$$

Reflectivity for r

Preparation for