Note: Define x_w-y_w-z_w.

```
//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

//define projection plane
```

Font Design

```
//----------letter--------------------*
letterL.X[0] = 10.0; letterL.Y[0] = 10.0;
letterL.X[1] = 20.0; letterL.Y[1] = 10.0;
letterL.X[2] = 20.0; letterL.Y[2] = 40.0;
letterL.X[3] = 40.0; letterL.Y[3] = 10.0;
letterL.X[4] = 50.0; letterL.Y[4] = 10.0;
letterL.X[5] = 30.0; letterL.Y[5] = 50.0;
```

For $\sin\theta$, $\cos\theta$, $\sin\phi$, $\cos\phi$ in the matrix of the transformation pipeline.

```
//sin and cosine computation for world-to-viewer
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;
```

Note: Define $\vec{P_s}(x_s, y_s, z_s)$

```
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 plane
```

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

CMPE240

Spring 2023

for $\vec{R}$ Ray Equation's

```
//----------lambda for Intersection pt on xw-yw plane-----------
float temp = (world.X[47]*(world.X[46]-world.X[45]))
            +(world.Y[47]*(world.Y[46]-world.Y[45]))
            +(world.Z[47]*(world.Z[46]-world.Z[45]));
float lambda = temp / ((world.X[47]*(world.X[45]-world.X[7]))
                     +(world.Y[47]*(world.Y[45]-world.Y[7]))
                     +(world.Z[47]*(world.Z[45]-world.Z[7])));
float lambda_2 = temp / ((world.X[47]*(world.X[45]-world.X[6]))
                       +(world.Y[47]*(world.Y[45]-world.Y[6]))
                       +(world.Z[47]*(world.Z[45]-world.Z[6])));
```

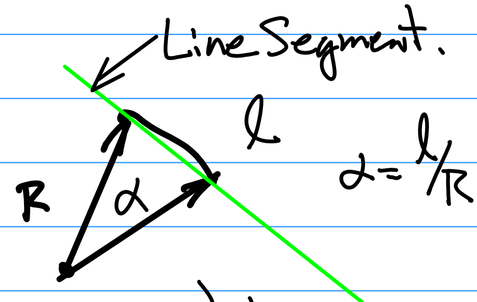
Find the intersection Points.

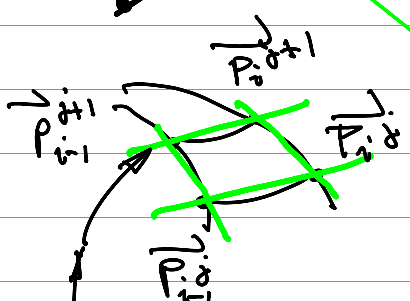
```
//-----------ray equation to find intersection pts-----------*
world.X[48] = world.X[45] + lambda*(world.X[45] - world.X[7]); // Ir
world.Y[48] = world.Y[45] + lambda*(world.Y[45] - world.Y[7]); // Ir
world.Z[48] = 0.0;
```

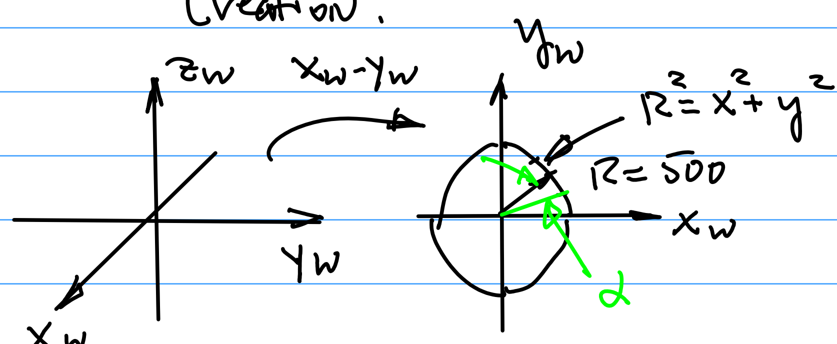
April 7 (Monday).

Note 1: Project in 3D is Due in 2 Weeks. See the previous Announcement (CANVAS Posting By the end of the Day Today).

Q & A. Spherical Surface Creation.



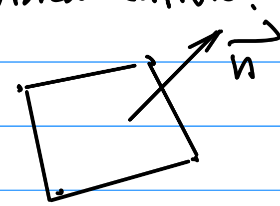




R Reduced By Predefined Proportion. Make at least 10 Layers for Better Visualization.

Define incremental α

make smaller $\alpha \approx 5°$



In Summary, we'll Create a Collection of Points

$$\{\vec{P_i}(x_i, y_i, z_i)\Big|_{z_i=0} ; i=0,1,\cdots N-1\}$$

Example:

Ref: Previous Project

2018F-115-lab-DiffuseReflection-Ru...

2018F-116-11diffuse20181114.cpp Sample code

Digital Differential Algorithm

2018F-117-12dda.cpp $y=ax+b$

2018F-118-13diffuseInterpolation20...

1. Definition.

Directions.

Fig.2

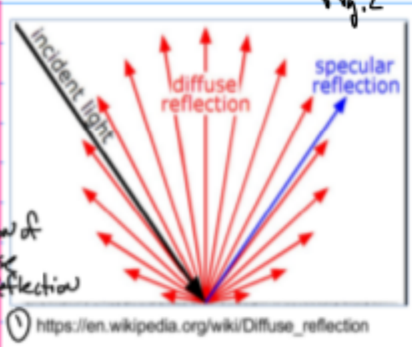


Illustration of Diffuse Reflection

2° Intensity of the Diffuse Reflection, The Intensity of

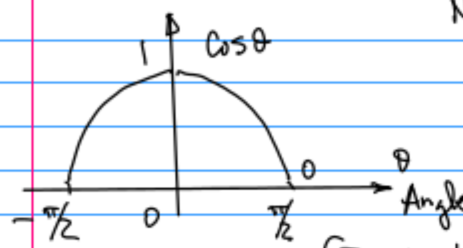
$I(x,y) = (r(x,y), g(x,y), b(x,y))$

red green Blue

depends on the incoming angle of the Ray Equation.

From Ref

Nov.21, 22



(Ray Eqn. & Normal Vector)

Fig.3

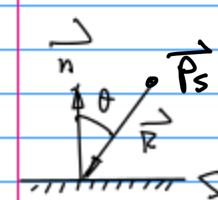


Fig.4

3° Intensity $\sim \frac{1}{(\text{distance})^2}$

Note: $\vec{n}$ Normal Vector of the Surface

$\vec{R}$ Ray Equation from the Light Source $\vec{P_s}(x_s, y_s, z_s)$ to the point of Interests

From pp. 48

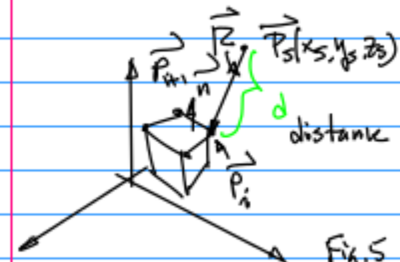


Fig.5

Note: Ray Equations:

$\vec{r_i}$ from $\vec{P_s}, \vec{P_i}$

$\vec{r_{i+1}}$ " $\vec{P_s}, \vec{P_{i+1}}$

$\vdots$

$\vec{r_{i+3}}$ ... $\vec{P_s}, \vec{P_{i+3}}$

From the Ray Equation.

$\vec{r} = \vec{P_i} + \lambda(\vec{P_s} - \vec{P_i})$ ...(1)

pp57.

$$\vec{n} \cdot \vec{r} = \|\vec{n}\| \|\vec{r}\| \cos\theta \quad \cdots (2)$$

$$\therefore \cos\theta = \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} \quad \cdots (3)$$

$I_{diff}(x,y)$ OR $I_d(x,y,z)$

"World"

$$I_d(x,y,z) \cong \cos\theta = \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} \quad \cdots (4)$$

$\downarrow$

Next, Consider the distance (Squared)

$$\|\vec{r}\|_2^2 = (x_s - x_i)^2 + (y_s - y_i)^2 + (z_s - z_i)^2$$

mode

then, update Eqn (4),

$$I_d(x,y,z) \cong \frac{1}{\|\vec{r}\|_2^2} \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} \quad \cdots (5)$$

$\downarrow$

Now, Let's Consider Reflectivity.

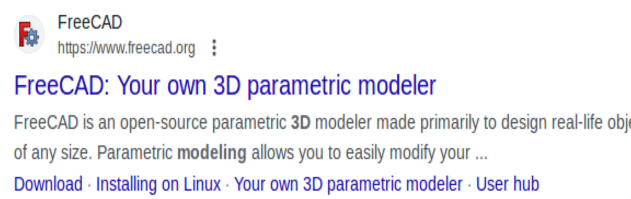
$$\text{Reflectivity } \vec{K_d} = (K_{dr}, K_{dg}, K_{db}) \quad \cdots (6)$$

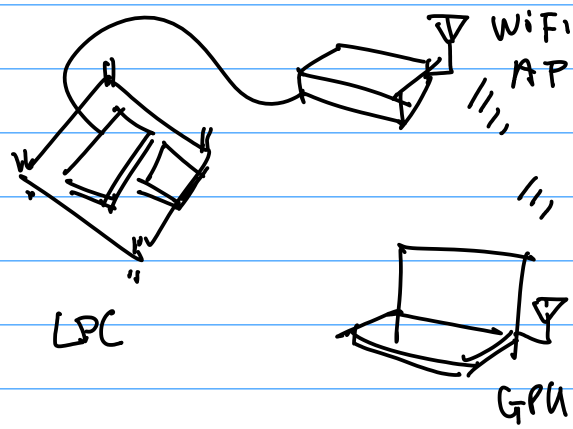
(Red, Green, Blue)

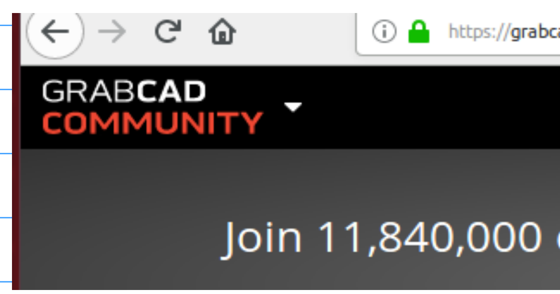
Update Eqn (5) with Reflectivity.
with Simplification, for Each Primitive Color.

$$\vec{I}_{dr}(x,y) = K_{dr} \frac{1}{\|\vec{r}\|^2} \frac{\vec{r} \cdot \vec{n}}{\|\vec{r}\| \|\vec{n}\|} \quad \cdots (7)$$

April 12 (Wed).
Note 1. Project Assignment is posted on CANVAS.
2. 5% Bonus for Using/Importing Real 3D CAD Data.







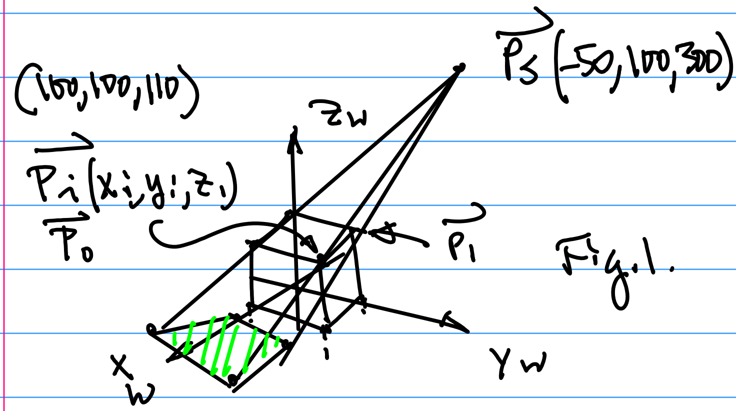


Fig.1.

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

$$= 150^2+0^2+190^2$$

Hence, $\dfrac{1}{\|\vec{r}\|^2} \ll \delta$ ...(1)

which makes $\vec{I_d}(x_i,y_i) \ll \delta$

Therefore, Suppose 8 bits per pixel

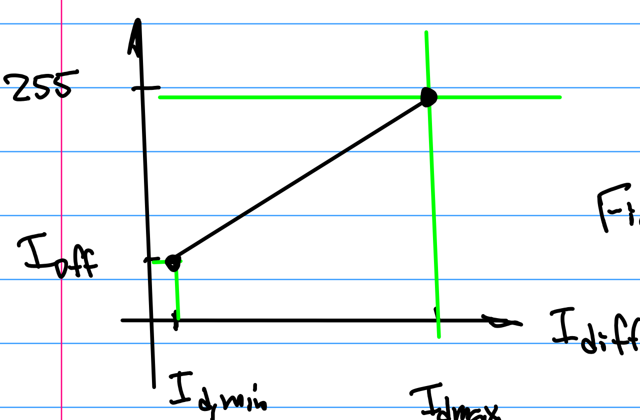


Fig.2

Where $I_{off}=20$.

$(I_{d\text{-}min}, I_{off})$ is a point on Fig. 2.

$(I_{dmax}, 255)$ is the other point

Let's define a Linear mapping function

Let

$$(I_{d\text{-}min}, I_{off}) = (x_1, y_1) \quad \ldots(2a)$$

$$(I_{dmax}, 255) = (x_2, y_2) \quad \ldots(2b)$$

then,

$$\frac{y_2-y_1}{x_2-x_1} = \frac{y-y_1}{x-x_1} \quad \ldots(3)$$

$$y = bx+c \quad \ldots(4)$$

Now, Suppose we want to display diffuse Reflection for a pixel location (x_i, y_i)

Step 1. Use Eqn (7), pp 39, to find $I_{diff}(x_i,y_i)$

Step 2. substitute $I_{diff}(x_i,y_i)$ into this Eqn (4)

$$y = bx+c \Big|_{x=I_{diff}(x_i,y_i)}$$

$$= b \cdot I_{diff}(x_i,y_i)+c$$

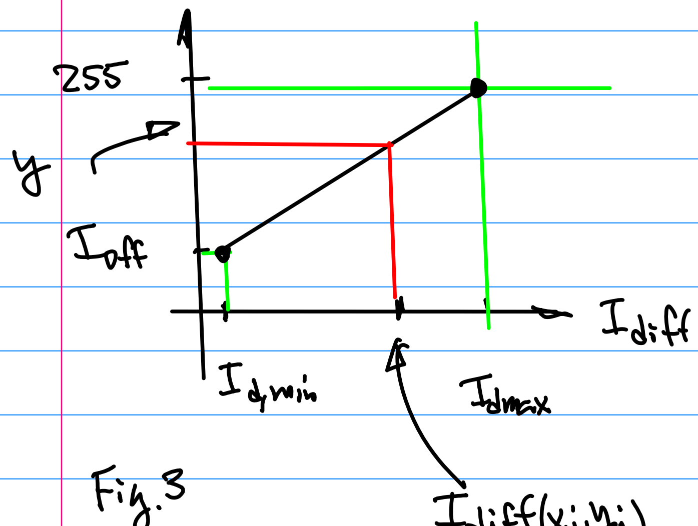
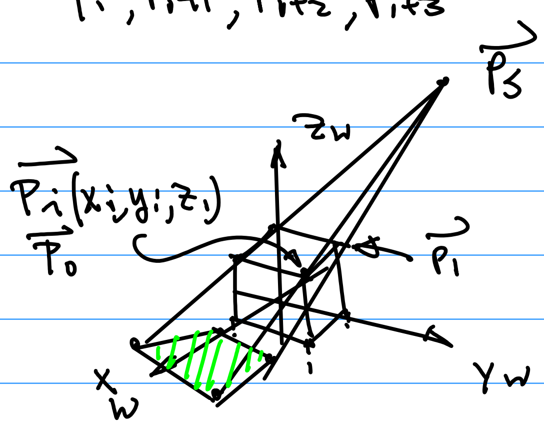


Fig.3

y is the final intensity Level for the color.

Now, we have 4 vertices with Diffuse Reflection Result.

$\vec{P_i}$, $\vec{P_{i+1}}$, $\vec{P_{i+2}}$, $\vec{P_{i+3}}$



Now, After Perspective Project

Fig.4

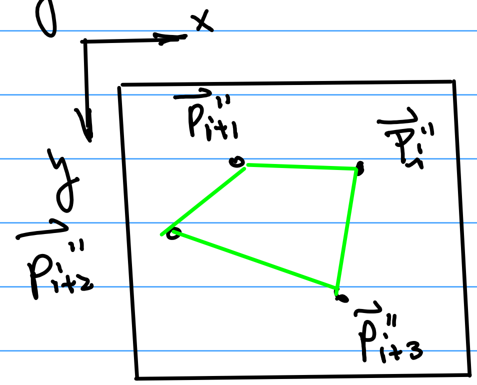


Fig.5

Find Diffuse Reflection on the Boundary Lines (Green).

April 17 (Monday).

Note 1. Check CANVAS for the Last Project Announcement.

Sample code Reading for Diffuse Reflection Computing.

2018F-116-11diffuse20181114.cpp

Example.

Preliminary 1. Intersection Pts. from the Ray Equations

```
//-----------ray equation to find intersection pts-----------*
world.X[48] = world.X[45] + lambda*(world.X[45] - world.X[7]);
world.Y[48] = world.Y[45] + lambda*(world.Y[45] - world.Y[7]);
world.Z[48] = 0.0;
```

Note: 1. Define Reflectivity,

```
//------------diffuse reflection----------*
pt_diffuse  diffuse;   //diffuse.r[3]

//-------reflectivity coefficient----------*
#define     Kdr     0.8
#define     Kdg     0.0
#define     Kdb     0.0
```

for Red color.

Note 2. Distance. To Speed up the Computation, No. Sqrt Needed.

```
//--------compute distance----------------*
float distance[UpperBD];
for (int i=48; i<=49; i++) {
distance[i] = sqrt(pow((world.X[i]-world.X[45]),2)+
                   pow((world.Y[i]-world.Y[45]),2)+
                   pow((world.X[i]-world.X[45]),2) );
//std::cout << "distance[i]  " << distance[i] << std::
```

Note 3. Compute $\cos\theta$ for Diffuse Reflection.

```
tmp_dotProd[i] = world.Z[i]-world.Z[45];
std::cout << " tmp_dotProd[i]  " <<  tmp_dotProd[i] << std::endl;

tmp_mag_dotProd[i] = sqrt(pow((world.X[i]-world.X[45]),2)+
                    pow((world.Y[i]-world.Y[45]),2)+
                    pow((world.Z[i]-world.Z[45]),2) );
std::cout << " tmp_mag_dotProd[i]  1 " <<  tmp_mag_dotProd[i] << std

angle[i] = tmp_dotProd[i]/ tmp_mag_dotProd[i];
std::cout << "angle[i]  " << angle[i] << std::endl;
```

Note 4. Theoretical Part of the Diffuse Reflection. The Result is Very Small

```
diffuse.r[i] = Kdr *  angle[i] / pow(distance[i],2) ;
diffuse.g[i] = Kdg *  angle[i] / pow(distance[i],2) ;
diffuse.b[i] = Kdb *  angle[i] / pow(distance[i],2) ;
}
```

Very Big Distance

CMPE240

Spring2023

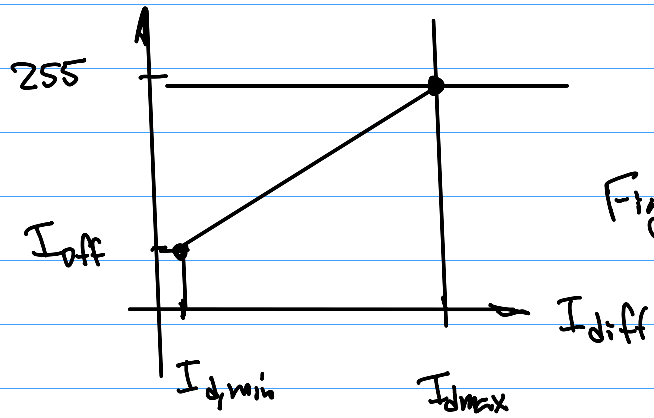


Fig.2

Sample code for the post Processing:

$\begin{cases} \text{Add offset} = 20 \\ \text{Map the diffuse Reflection} \\ [\text{offset}, 255] \end{cases}$

CMPE240-Adv-Microprocessors / 2018F / **2022S-101-notes2-cmpe240-2022-04-18.pdf.pdf.20.pdf**

Post processing function, PP13

$$\frac{x-x_2}{y-y_2} = \frac{x_1-x_2}{y_1-y_2} \quad \cdots (5)$$

```
float r, g, b;
r = display_scaling*diffuse.r[i]+display_shifting;
//r = display_scaling*diffuse.r[i];
g = diffuse.g[i]; b = diffuse.b[i] ;
glColor3f(r, g, b);
```

Example: Bi-Linear Interpolation of Diffuse Reflection.

From Eqn (5), PP.14.

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

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

$$y = y_2 + \frac{y_2-y_1}{x_2-x_1}(x-x_2) \qquad y = b'x + c'$$

$$\boxed{y = \frac{y_2-y_1}{x_2-x_1}x - \frac{y_2-y_1}{x_2-x_1}x_2 + y_2} \quad \cdots (1)$$

$$a\,y = b\,x + c, \quad y = \frac{b}{a}x + \frac{c}{a} \quad \cdots (2)$$

$$\frac{b}{a} = \frac{y_2-y_1}{x_2-x_1} \quad \cdots (2\text{-}b)$$

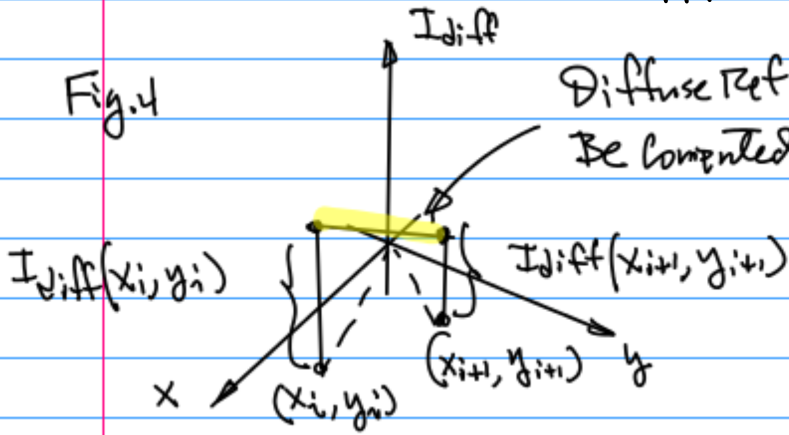
CmpE240

Spring 2023

/ 2018F / 2020S-APL29-BilinearDiff1.jpg

PP15.

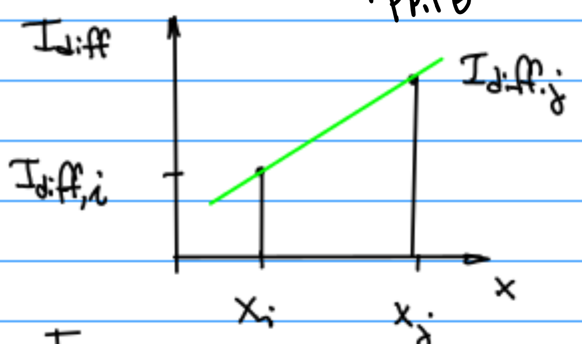
Fig.4



Diffuse Reflection to Be Computed, on the Bounding line with $\vec{P_i}(I_{diff,i})$ and $\vec{P}_{i+1}(I_{diff,i+1})$ as Starting & Ending point.

x, y are from the Display Device

PP.16



Fig(1.c)

then, derived the following Equations

$$\boxed{y = \frac{y_2 - y_1}{x_2 - x_1} x - \frac{y_2 - y_1}{x_2 - x_1} x_2 + y_2} \quad \cdots (1)$$

$$a\,y = b\,x + c, \quad y = \frac{b}{a} x + \frac{c}{a} \quad \cdots (2)$$

$$\frac{b}{a} = \frac{y_2 - y_1}{x_2 - x_1} \quad \cdots (2\text{-}b)$$

$$\frac{c}{a} = -\frac{y_2 - y_1}{x_2 - x_1} x_2 + y_2 \quad \cdots (2\text{-}c)$$

From,

$$\frac{x - x_2}{y - y_2} = \frac{x_1 - x_2}{y_1 - y_2} \quad \cdots (5)$$

PP.17.

Therefore. 2020S-APL29-BilinearDiff2.jpg

$$I_{diff,x} = \frac{I_{diff,j} - I_{diff,i}}{x_j - x_i} x - \frac{I_{diff,j} - I_{diff,i}}{x_j - x_i} x_j + I_{diff,j} \quad \cdots (3)$$

For I_{diff} w.r.t y. we have (Symmetric)

$$I_{diff,y} = \frac{I_{diff,j} - I_{diff,i}}{y_j - y_i} y - \frac{I_{diff,j} - I_{diff,i}}{y_j - y_i} y_j + I_{diff,j} \quad \cdots (4)$$

CmpE240

Spring 2023.

Hence,

$$I_{diff} = \frac{1}{2}\left[I_{diff,x} + I_{diff,y}\right] \quad \cdots (5)$$

April 19 (Wed)

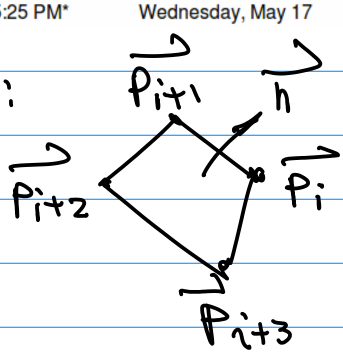
Final Exam:

Group I Classes

Group I classes are those classes which meet M, W, F, MTW, MWR, MTWF, MWRF, MTWRF, MW, WF, MWF, MF, TW, WR, MT, WS.

Regular Class Start Times	Final Examination Days	Final Examination Times
7:00 through 8:25 AM	Friday, May 19	7:15-9:30 AM
8:30 through 9:25 AM	Tuesday, May 23	7:15-9:30 AM
9:30 through 10:25 AM	Thursday, May 18	7:15-9:30 AM
10:30 through 11:25 AM	Monday, May 22	9:45 AM-12:00 PM
11:30 AM through 12:25 PM	Wednesday, May 17	9:45 AM-12:00 PM
12:30 through 1:25 PM	Friday, May 19	12:15-2:30 PM
1:30 through 2:25 PM	Tuesday, May 23	12:15-2:30 PM
2:30 through 3:25 PM	Thursday, May 18	12:15-2:30 PM
3:30 through 4:25 PM*	Monday, May 22	2:45-5:00 PM
4:30* through 5:25 PM*	Wednesday, May 17	2:45-5:00 PM

Example:



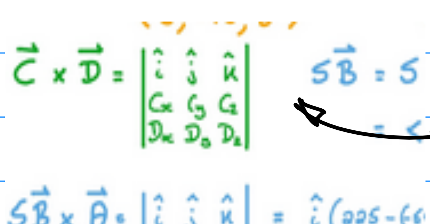
Detect the orientation

$$\{\vec{P}_i, \vec{P}_{i+1}, \cdots, \vec{P}_{i+3} \mid i = 1, 2, \cdots N\}$$

$$(\vec{P}_i - \vec{P}_{i+3}) \times (\vec{P}_{i+2} - \vec{P}_{i+3}) \quad \cdots (1)$$

Hence,

$$\vec{n}^{\,0} = \frac{(\vec{P}_i - \vec{P}_{i+3}) \times (\vec{P}_{i+2} - \vec{P}_{i+3})}{\|(\vec{P}_i - \vec{P}_{i+3}) \times (\vec{P}_{i+2} - \vec{P}_{i+3})\|_2} \quad \cdots (2)$$



Google Ref.

Consider DDA Algorithm,

Digital Differential Algorithm

$$y = bx + c \quad \cdots (1)$$

To plot Equation / Line Segment on a finite Display Device.

HD, 4K etc.

Technical challenges:

1°. "GAPS" Problem

2°. Removal of multiplication.

Consider Computation of y_k, y_{k+1}:

we have

for x_k, from Eqn (1).

$$y_k = bx_k + c \quad \cdots (1a)$$

for $x_{k+1} = x_k + 1$, then

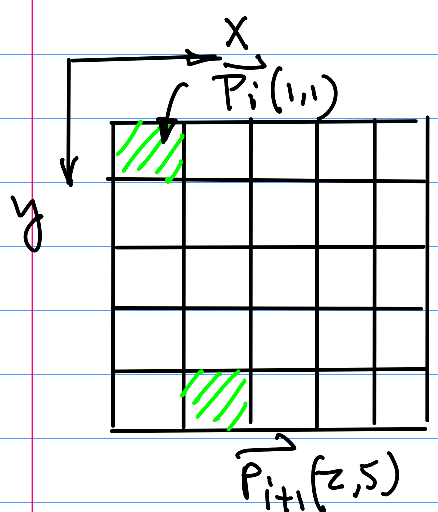
$$y_{k+1} = \underbrace{b x_{k+1}}_{\text{multiplication.}} + c$$

$$= b(x_k + 1) + c = bx_k + b + c$$

$$= y_k + b \quad \cdots (1b)$$

Example: Given $\vec{P}_i = (1,1)$, $\vec{P}_{i+1} = (1,5)$

use Eqn (1a) or (1b) to plot a line.



$$b = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = 4$$

Solve for C :

$$y = bx + c \Big|_{b=4} = 4x + C$$

$$1 = 4 + C$$

$$\therefore C = -3$$

Hence

$$y = 4x - 3 \quad \cdots (3)$$

Let $x_K = 1$, $y_K = 1$ (From Eqn(3))

$$x_{K+1} = x_K + 1 = 1 + 1 = 2$$

From Eqn (1b)

$$y_{K+1} = y_K + b = 1 + 4 = 5$$

Gap ! which is a problem.

April 24 (Monday).

Example: Continuation on DDA.

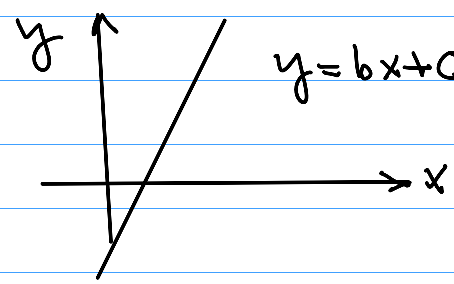
Note: when the slop $|b| > 1$ then Eqn (1-b), pp45, will land the Next

Point y_{K+1} on a pixel Location with a gap

To Solve this problem, make the Slop of the given Line is less than 1.

. Absolute value of

Consider $y = bx + c$, where $|b| > 1$. $\cdots (1)$



From Eqn(1),

$$y/b = x + c/b$$

$$x = \frac{1}{b} y - \frac{c}{b} \quad \cdots (2)$$

where $\left|\frac{1}{b}\right| < 1$

$$x_K = \frac{1}{b} y_K - \frac{c}{b}$$

$$y_{K+1} = y_K + 1 \quad \cdots (3a)$$

$$x_{K+1} = \frac{1}{b} y_{K+1} - \frac{c}{b}$$

$$= \frac{1}{b}(y_K + 1) - \frac{c}{b}$$

$$= \frac{1}{b} + \underbrace{\frac{1}{b} y_K - \frac{c}{b}}_{x_K}$$

$x_{k+1} = x_k + \frac{1}{b} \quad \ldots (3b)$

Going Back to the Same example.

for $k=1$, $y_k = 1$, $x_k = 1$.

for $k=2$.

$$y_2 = y_1 + 1 \left(= y_k + 1 \Big|_{k=1} \right) = 2$$

$$x_2 = x_1 + \frac{1}{b} = 1 + \frac{1}{4} = 1.25 \simeq 1$$

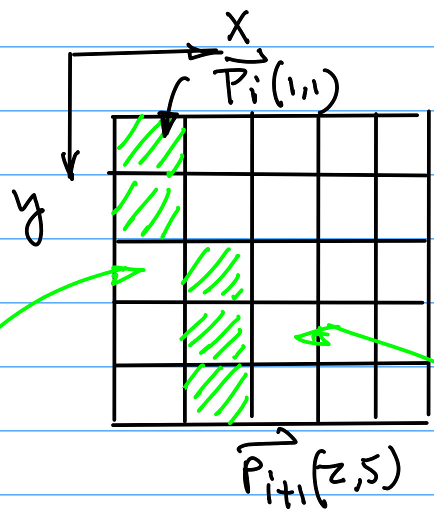


Fig. 1

for $y_3 = 3$,

$$x_3 = x_2 + \frac{1}{b} = 1.25 + 0.25 = 1.5 \simeq 2$$

for $y_4 = 4$,

$$x_4 = x_3 + \frac{1}{b} = 1.5 + 0.25 = 1.75 \simeq 2$$

Diffuse Reflection on the interior points.

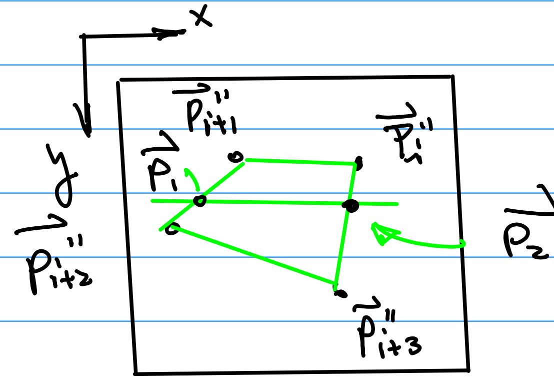


Fig. 2

$\vec{P_1}$, $\vec{P_2}$ Are Both ON the Boundary
Hence, their (1) pixel Location are computed By DDA;
(2) Diffuse Reflection are computed By Eqn (3), (4), and (5) ON TP44, 45;

Therefore. 2020S-APL29-BilinearDiff2.jpg

$$I_{diff,x} = \frac{I_{diff,j} - I_{diff,i}}{x_j - x_i}\,x - \frac{I_{diff,j} - I_{diff,i}}{x_j - x_i}\,x_j + I_{diff,j} \quad \ldots (3)$$

Example: Decoration Algorithm.

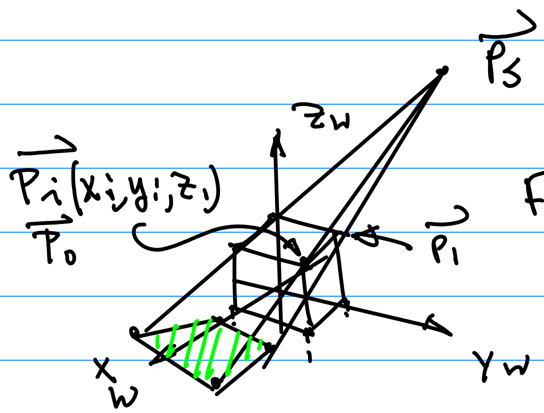


Fig. 3.

Decorate the Cube Surface.

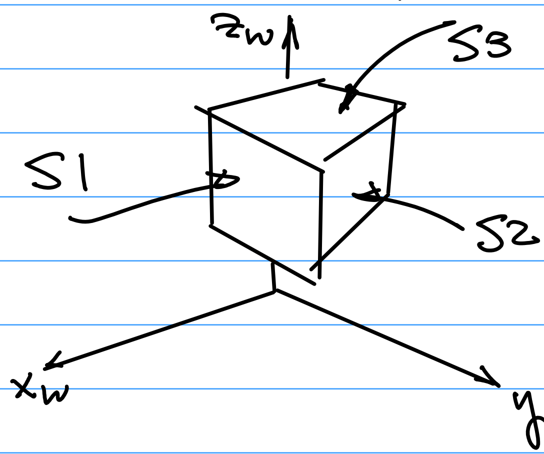


Fig.4a

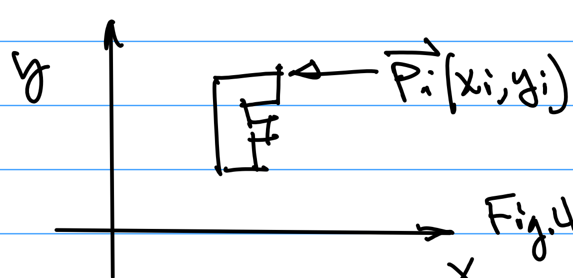


Fig.4b.

Use 2D Pattern $\{\vec{P_i}(x_i, y_i) \mid i = 1, 2, \cdots N\}$ to Decorate 3D Surfaces.

Background: 3D Transforms.

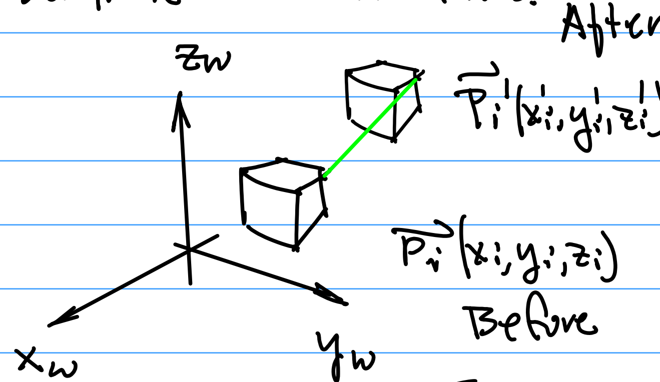


Fig. 5

$$\underset{\text{After}}{x_i'} \overset{?}{=} \underset{\text{Before}}{x_i + \Delta x}$$

$$y_i' = y_i + \Delta y, \quad z_i' = z_i + \Delta z$$

$$T = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \ldots (4)$$

$$\begin{pmatrix} x_i' \\ y_i' \\ z_i' \\ 1 \end{pmatrix} = T \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \quad \ldots (5)$$

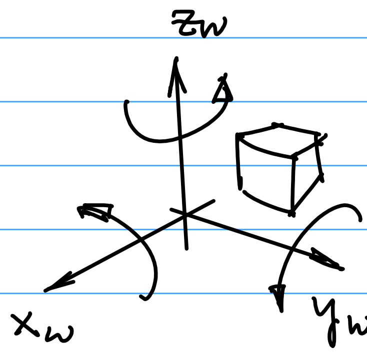


Fig.6

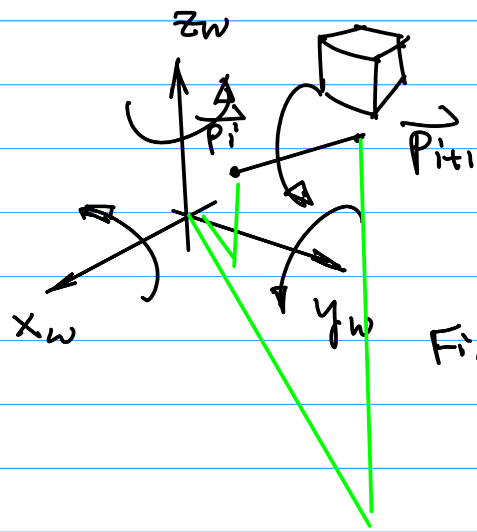


Fig.7

Rotation w.r.t. X_w-axis.

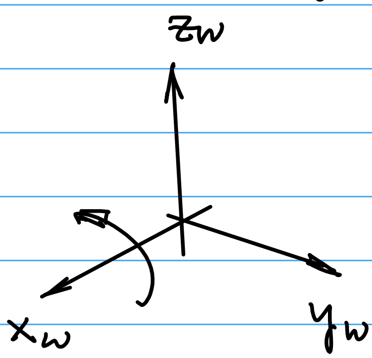


Fig.8

Methodology: Observe | Discover the independent variable which stays the constant. $x_i \longrightarrow$ Rotation on y_w-z_w plane.

From 2D Rotation Matrix

$$\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdots (6)$$

$$R_{x_w} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdots (7)$$

April 26 (Wed).

Example: Linear Decoration Algorithm.

Surface $\rightarrow$ plane

Continuation on pp48, Fig.4a ~ 4b.

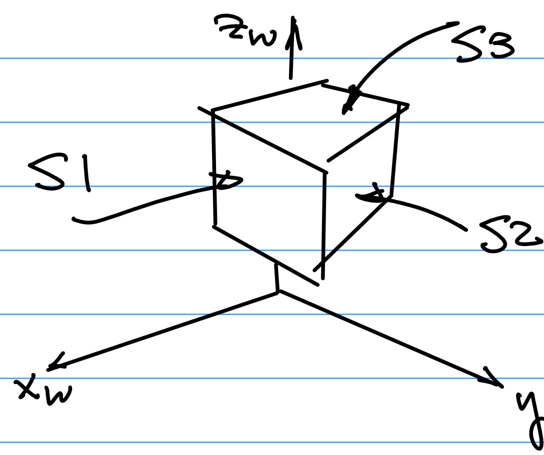


Fig.4a

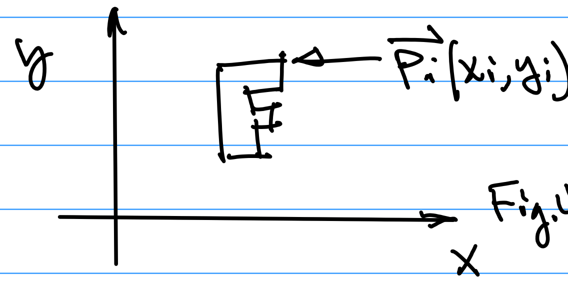


Fig.4b.

Step 1. Given $\{\vec{P_i}(x_i, y_i) \mid i = 1, 2, \cdots, K\}$

Redefine $\{\vec{P_i}(x_i, y_i)\}$ in the World Coordinate System.

$\downarrow$

$$\{\vec{P_i}(x_i, y_i, z_i)\big|_{z_i=0}\ i=1,2,\cdots,k\} \quad \cdots (1)$$

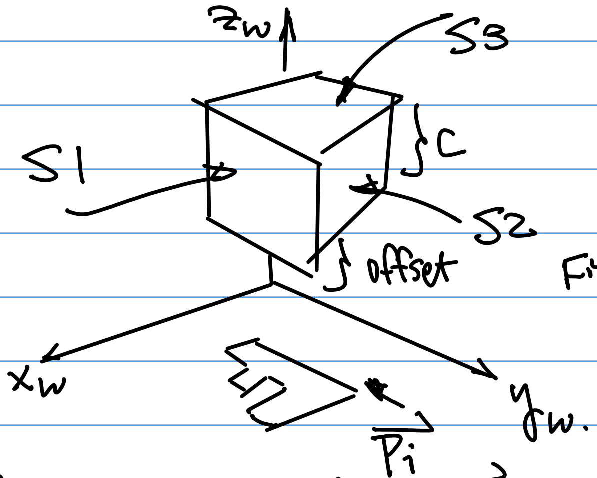


Fig. 4a

Note: Check the scale of $\{\vec{P_i}(x_i, y_i, z_i)\}$ to make sure it match the Size of the Surface (plane) to Be decorated:

Step 2. Consider Surface 3.

Remark: 1° Identify the parallel plane for the surface; $\leftarrow$ X_w-Y_w

2° Identify the indep. Variable & Function of the plane. $\leftarrow$ x_w. (Indep). V.S. y_w (Function)

After ? Before

$$\begin{cases} x_i' \overset{?}{=} x_i & \text{(Indep)} \\ y_i' = y_i & \\ z_i' = C + \text{offset} & \end{cases} \quad \cdots (1)$$

Consider S1:

Find A parallel plane y_w-z_w.

$\downarrow$

Indep. Variable V.S. Function are y_w V.S. z_w.

After Before

$$\begin{cases} y_i' \overset{?}{=} x_i & \text{Indep.} \\ z_i' = y_i & \text{Function} \\ x_i' = C & \end{cases} \quad \cdots (2)$$

Consider S2

Parallel plane Z_w-X_w

$\downarrow$

Indep: Z_w, V.S. Function X_w

After Before

$$\begin{cases} z_i' = x_i & \text{Indep.} \\ x_i' = y_i & \text{Function} \\ y_i' = c & \end{cases} \quad \cdots (3)$$

Optional project Topic Discussion: "Mirror" Image.

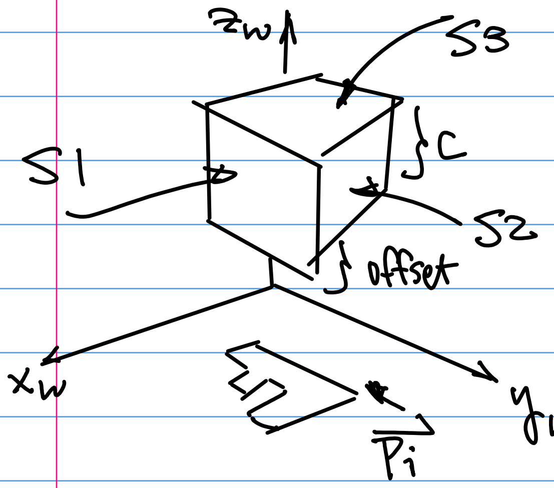
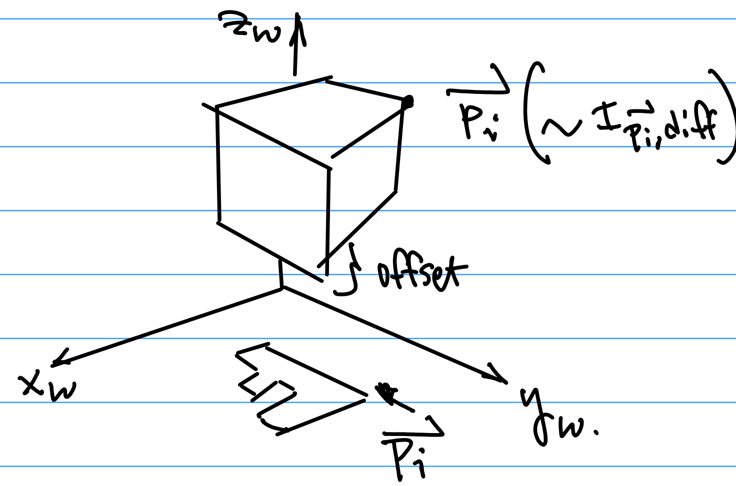
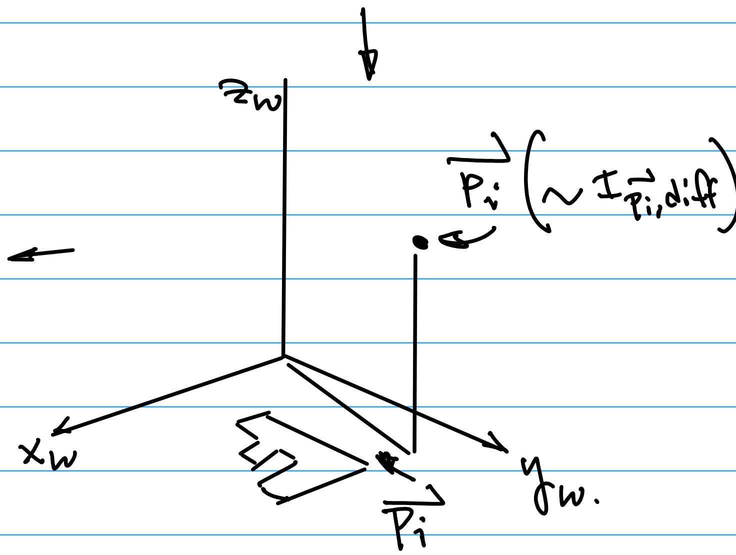
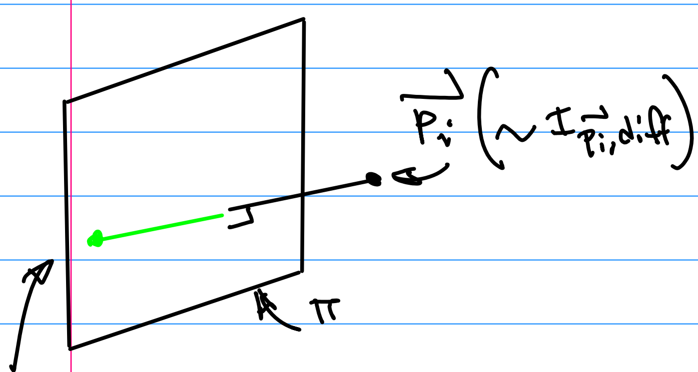


Fig.4a







$\vec{P_i}$ (Find $I_{\vec{P_i}, diff}$)

Line Eqn: $\vec{P} = \vec{P_i} + \lambda \vec{d}$ ...(4)

$\vec{d}$ is defined by the plane.
Cross product of 2 vectors
on the "π" plane

↓

λ (Defines the distance
from $\vec{P_i}$ to the π plane)

↓

2λ Reflection Point

↓

Transformation Pipeline

Point on 2D Display
(Location)
(x_i'', y_i'')

↓

Copy diffuse reflection from
$\vec{P_i}(x_i, y_i, z_i)$

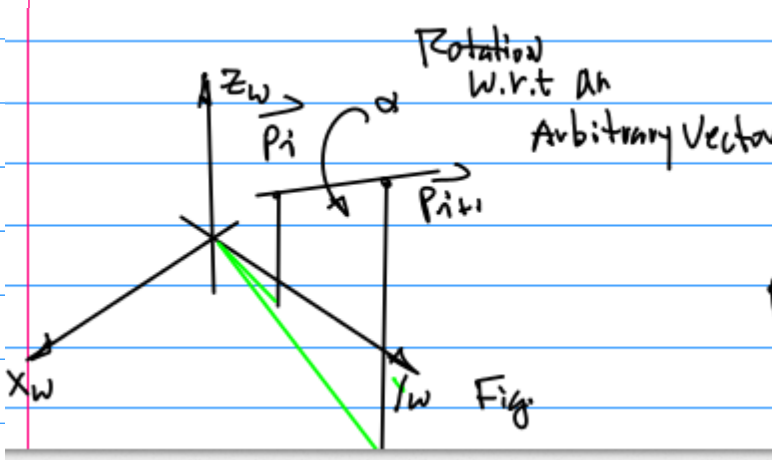
2023-5-1

2022F-101-notes-cmpe240-2022-11-30.pdf

pp.37

Example: for the arbitrary rotoations
1. Pre-processing: 3 steps;
2. Rotation w.r.t. Z_w axis;
3. Post processing: 3 steps;

Tools: translations and rotations



Pre-processing:
Step 1. Translation
Delta_X = - x_i
Delta_y = - y_i
Delta_z = - z_i .... (1)

Step 2. Rotation w.r.t z-axis

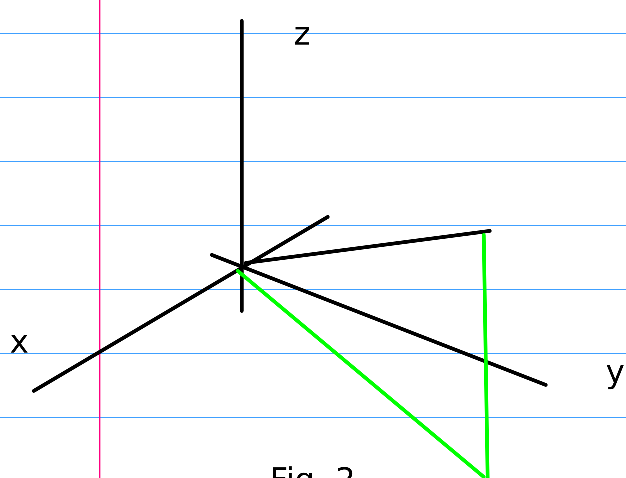


Fig. 2

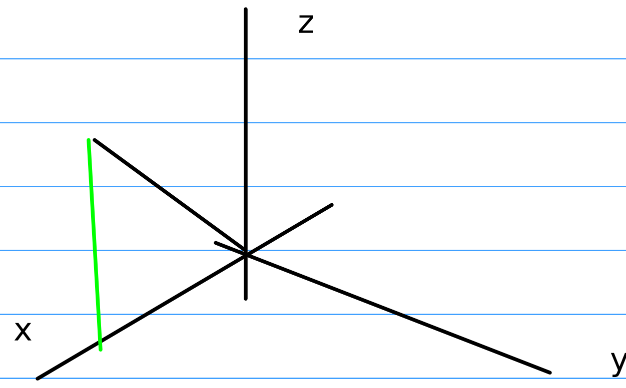


Fig. 3

rotation matrix R_z
clockwise rotation
Find the rotation matrix;

Step 3. Rotation w.r.t y_w axis

Find the rotation matrix R_y

Note: 3 steps together:

R_y R__z T ... (2)

Coding in C/C++, we will need
3 lines of code for x, y, and z;

Step 4. Rotation wrt Z-axis
per the requirement

Step 5. Undo rotation y

R_y^(-1) ... (3)

Note just need to change the sign
of the rotation angle;

Step 6. Undo rotation wrt to Z

R_z^(-1) ... (4)

Step 7. Undo the translation

T^(-1)

Put all the euqations together

T^(-1) R_z^(-1) R_y^(-1) R_z   R_y R__z T

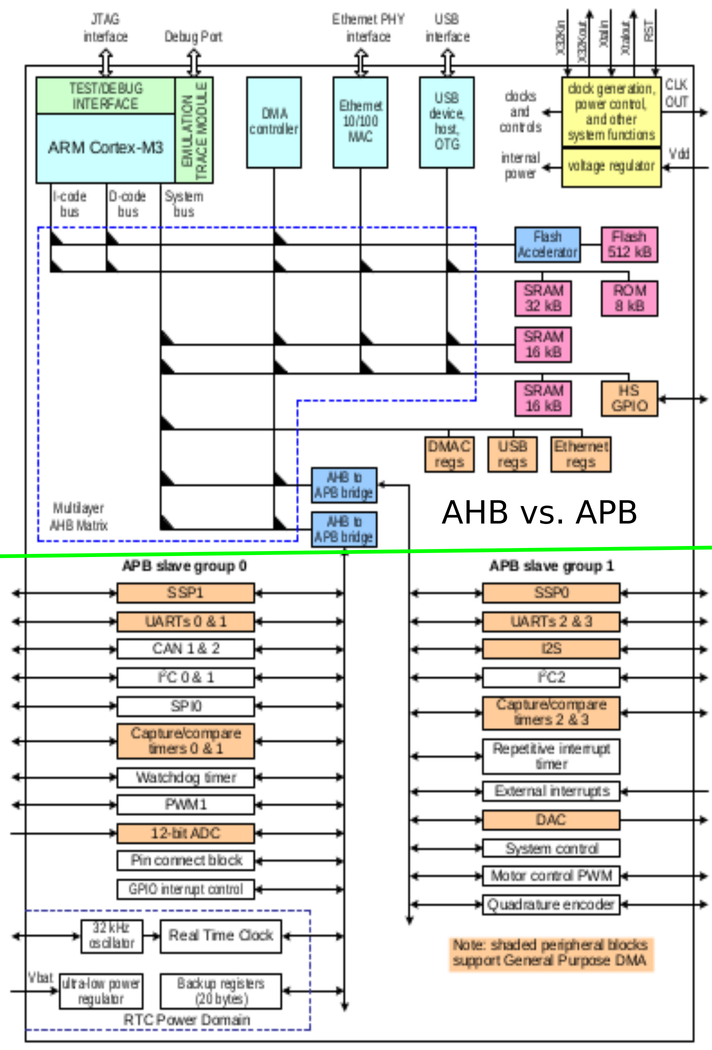
... (5)

Note: to write C/C++ code,
we will need to have 3 lines of code,
one for x, one for y, and one for z.

For the optional proejct (Bonus),
please use the following vector:

P_i(10, 10, 10) and
P_(i+1) (200, 200, 300)

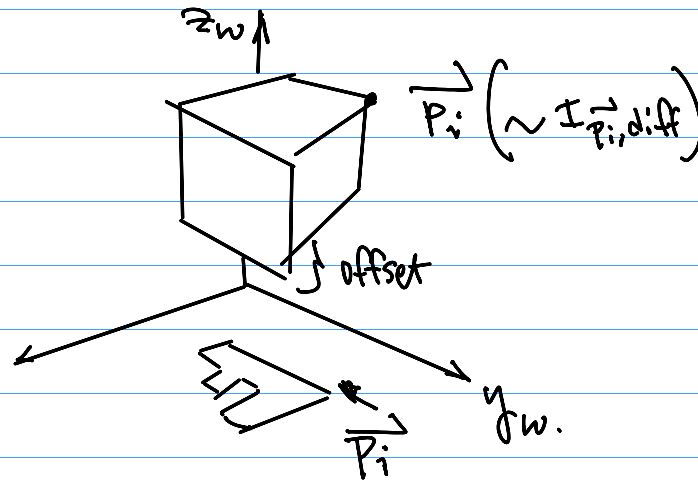
Rotation by 5 degree;



AHB vs. APB

Example:

Conditions:
1. After the transformation pipeline;



2 . Diffuse reflection can be added for further analysis;

3. Note: DDA has to be a part of it.

Analysis:
Step 1. World to Viewer transform (Assuming the sines, cos, and roh have been given)

For x: mul 2; additions: 1;
For y: mul: 3; additions: 2;
For z: mul: 3; additions: 3 (rho)

Step 2. Perspective Projection:
For x: mul: 2;
For y: mul: 2;

Step 3. Virtual to physical
For x: addition: 1;
For y: addition: 1;

In summary:

for each vertex:
Mul: 12
addition: 8

Consider the ARM Cortex 3
1 clock for 1 addition (pipeline is filled)
1 clock for 1 multiplication

Clock rate of the CPU: 200 Mhz;

No. of poly per second =

Clock/(mul+add) = 200/(12+8)
= 10 Million Poly / Second