Note: Define x_w-y_w-z_w.

```
78      //define the x-y-z world coordinate
79      world.X[0] = 0.0;   world.Y[0] =  0.0;   world.Z[0] =  0.0;    // origin
80      world.X[1] = 50.0;  world.Y[1] =  0.0;   world.Z[1] =  0.0;    // x-axis
81      world.X[2] = 0.0;   world.Y[2] =  50.0;  world.Z[2] =  0.0;    // y-axis
82      world.X[3] = 0.0;   world.Y[3] =  0.0;   world.Z[3] =  50.0;   // y-axis
83
```

Font Design

```
98      //----------letter----------*
99      letterL.X[0] = 10.0; letterL.Y[0] = 10.0;
100     letterL.X[1] = 20.0; letterL.Y[1] = 10.0;
101     letterL.X[2] = 20.0; letterL.Y[2] = 40.0;
102     letterL.X[3] = 40.0; letterL.Y[3] = 10.0;
103     letterL.X[4] = 50.0; letterL.Y[4] = 10.0;
104     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 pipeling.

```
159     //sin and cosine computation for world-to-viewer
160     float sPheta = Ye / sqrt(pow(Xe,2) + pow(Ye,2));
161     float cPheta = Xe / sqrt(pow(Xe,2) + pow(Ye,2));
162     float sPhi = sqrt(pow(Xe,2) + pow(Ye,2)) / Rho;
163     float cPhi = Ze / Rho;
164
```

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

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

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

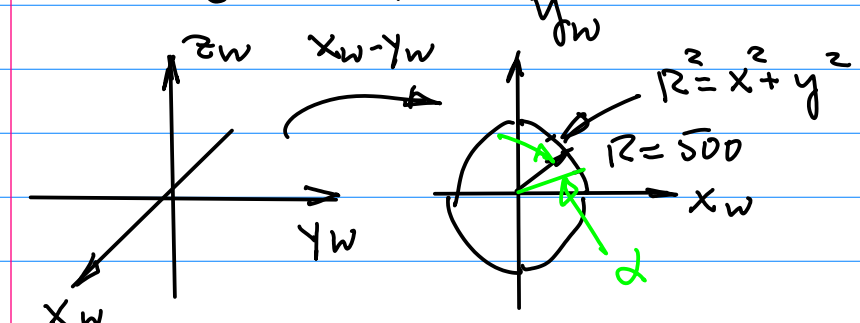
for $\vec{R}$ Ray Equation's

```
171       //----------lambda for Intersection pt on xw-yw plane----------*
172       float temp = (world.X[47]*(world.X[46]-world.X[45]))
173                  +(world.Y[47]*(world.Y[46]-world.Y[45]))
174                  +(world.Z[47]*(world.Z[46]-world.Z[45]));
175       float lambda = temp / ((world.X[47]*(world.X[45]-world.X[7]))
176                             +(world.Y[47]*(world.Y[45]-world.Y[7]))
177                             +(world.Z[47]*(world.Z[45]-world.Z[7])));
178       float lambda_2 = temp / ((world.X[47]*(world.X[45]-world.X[6]))
179                               +(world.Y[47]*(world.Y[45]-world.Y[6]))
180                               +(world.Z[47]*(world.Z[45]-world.Z[6])));
```

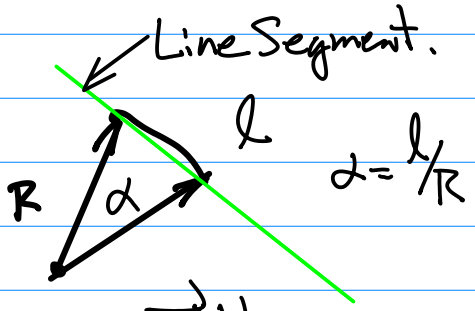
Find the intersection points.

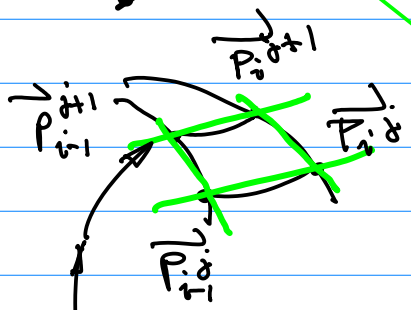
```
182       //-----------ray equation to find intersection pts------------*
183       world.X[48] = world.X[45] + lambda*(world.X[45] - world.X[7]); // Ir
184       world.Y[48] = world.Y[45] + lambda*(world.Y[45] - world.Y[7]); // Ir
185       world.Z[48] = 0.0;
186
```

April 7 (Monday).
Note1: 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.

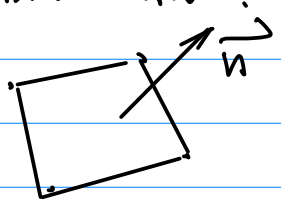


Define incremental α
make smaller α ≈ 5°





R Revised By predefined proportion. make at least 10 Layers for Better Visualization.



In Summary, we'll Create a Collection of Points

$$\left\{ \vec{P}_i(x_i, y_i, z_i) \,\middle|_{z_i=0} \; ; \; i = 0, 1, \cdots N-1 \right\}$$

Example:

Ref:

Previous Project

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

2018F-116-11diffuse20181114.cpp Sample code

Digital Differential Algorithm

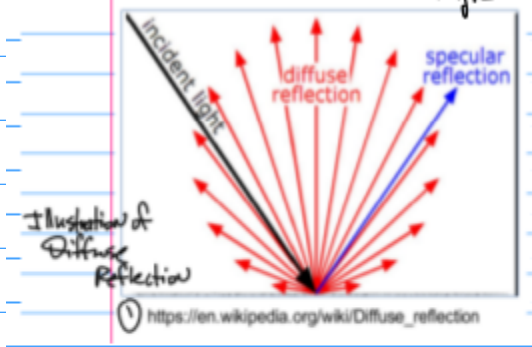
2018F-117-12dda.cpp $y = ax + b$

2018F-118-13diffuseInterpolation20...

1. Definition.

directions.

Fig.2



2° Intensity of the Diffuse Reflection, The Intensity of

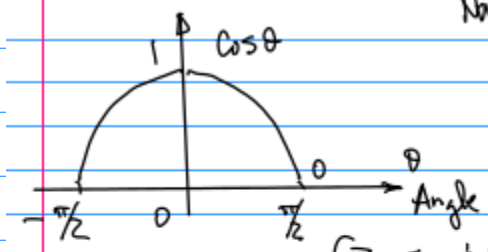
$$I(x,y) = \left(r(x,y), g(x,y), b(x,y) \right)$$

r: red, g: green, b: Blue

depends on the incoming angle of the Ray Equation.

From Ref

Nov. 21, 22



(Ray Eqn. & Normal Vector)

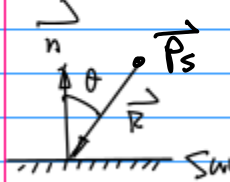


Fig.3

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

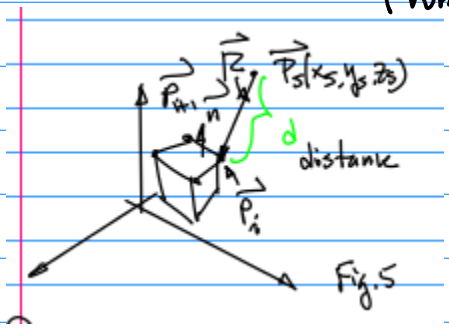


Fig.5

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.

$$\vec{r} = \vec{P}_i + \gamma(\vec{P}_s - \vec{P}_i) \quad \cdots (1)$$

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"

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

↓

Next, Consider the distance (squared)

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

then, update Eqn(4),

mode

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

↓

Now, Let's Consider Reflectivity.

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

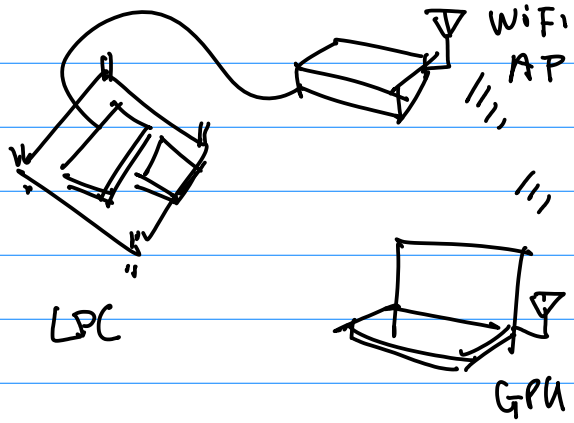
Red Green Blue

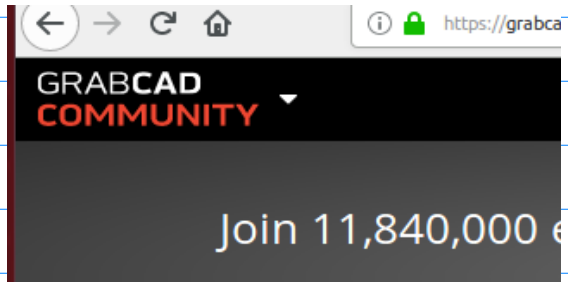
Update Eqn(5) with Reflectivity.
with Simplification, for Each Primitive Color.

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

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

FreeCAD
https://www.freecad.org
FreeCAD: Your own 3D parametric modeler
FreeCAD is an open-source parametric 3D modeler made primarily to design real-life obje of any size. Parametric modeling allows you to easily modify your ...
Download · Installing on Linux · Your own 3D parametric modeler · User hub





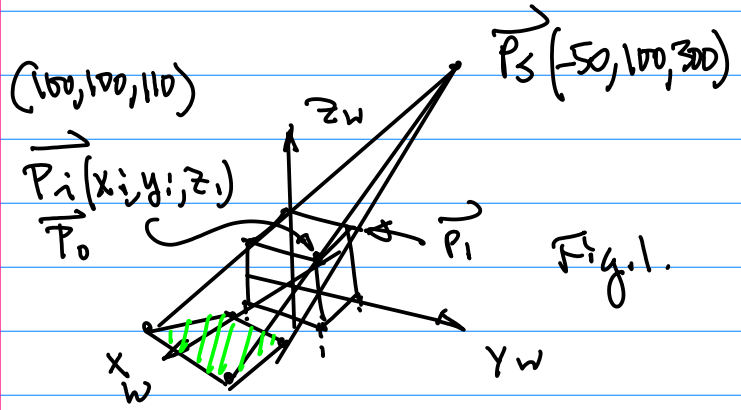


Fig. 1.

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

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

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

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

Therefore, Suppose 8 bits per pixel

Fig. 2

Where $I_{off} = Z0$.

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

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

Let's define a Linear mapping function

Let

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

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

then,

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

$$y = bx+c \quad \text{...(4)}$$

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)

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

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

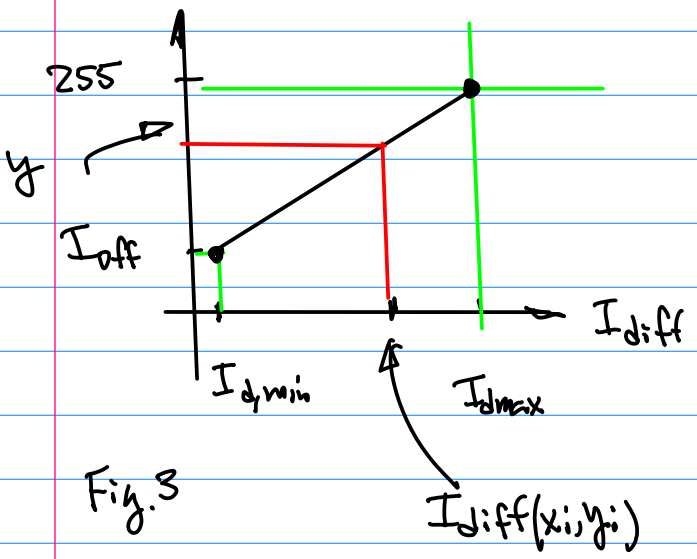
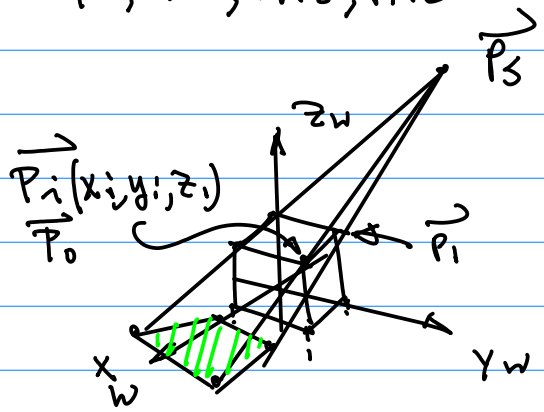


Fig.3

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

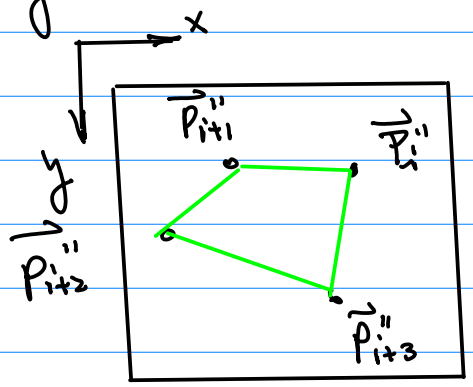


Fig 5

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

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

Note: 1. Define Reflectivity,

```
191     //-------------diffuse reflection-----------*
192     pt_diffuse  diffuse;   //diffuse.r[3]
193
194     //-------reflectivity coefficient-----------*
195     #define     Kdr     0.8
196     #define     Kdg     0.0
197     #define     Kdb     0.0
```

for Red color.

Note2. Distance. To Speed up the Computation, No. Sqrt Needed.

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

Note 3. Compute Cos θ for Diffuse Reflection.

```
229       tmp_dotProd[i] = world.Z[i]-world.Z[45];
230       std::cout << " tmp_dotProd[i]  " <<  tmp_dotProd[i] << std::endl;
231
232       tmp_mag_dotProd[i] = sqrt(pow((world.X[i]-world.X[45]),2)+
233                           pow((world.Y[i]-world.Y[45]),2)+
234                           pow((world.Z[i]-world.Z[45]),2) );
235       std::cout << " tmp_mag_dotProd[i]  1 " <<  tmp_mag_dotProd[i] << std
236
237       angle[i] = tmp_dotProd[i]/ tmp_mag_dotProd[i];
238       std::cout << "angle[i]  " << angle[i] << std::endl;
239
```

Note 4. Theoretical Part of the Diffuse Reflection. The Result is very small

```
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       }
```

Very Big Distance

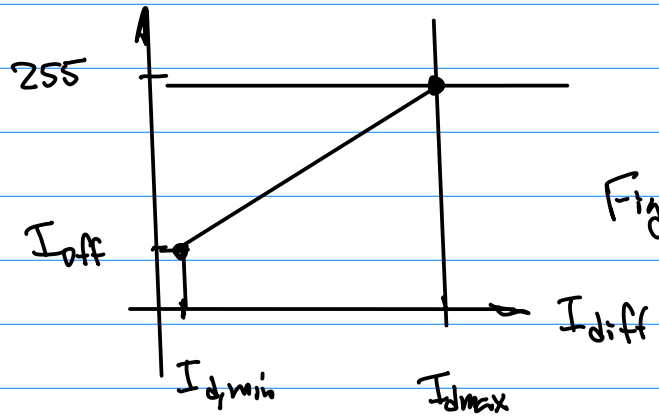


Fig.2

Sample code for the post Processing.

- Add offset = 20
- Map the diffuse Reflection [offset, 255]

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

Post processing function, PP13

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

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

Example: Bi-Linear Interpolation of Diffuse Reflection.

PP.14.

From Eqn (5),

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

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

$y = b'x + c'$

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

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

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

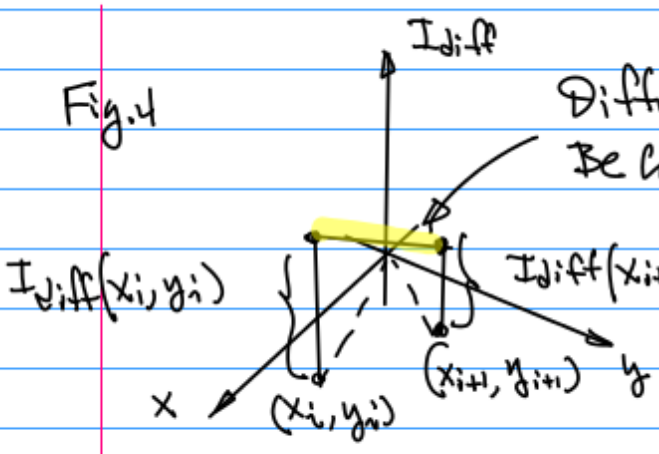
$$\frac{b}{a} = \frac{y_2-y_1}{x_2-x_1} \quad \cdots (2\text{-}b)$$

~~(x_i, y_i).~~

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

PP15.

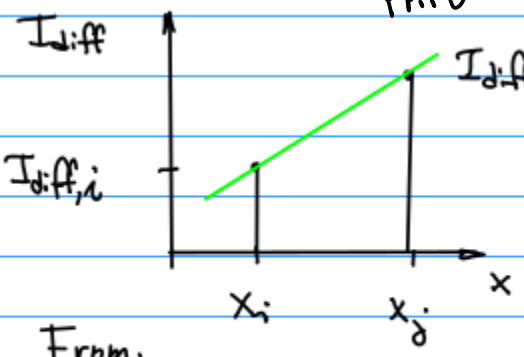
Fig.4



Diffuse Reflection to Be computed, on the Boundary line with $P_i(I_{diff,i})$ and $P_{i+1}(I_{diff,i+1})$ as Starting & Ending point.

x, y are from the Display Device

PP16



Fig(1.c)

From,

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

then, derive & the following Equations

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

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

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

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

PP17.

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)

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

Hence, $I_{diff} = \frac{1}{2}[I_{diff,x} + I_{diff,y}]$ ... (5)

April 19 (Wed)

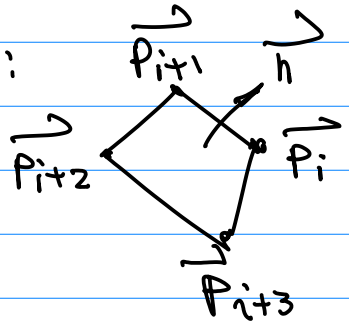
Final Exam:

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

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

$(\vec{P}_i - \vec{P}_{i+3}) \times (\vec{P}_{i+2} - \vec{P}_{i+3})$ ... (4)

Hence,

$$\vec{n} = \frac{(\vec{P}_i - \vec{P}_{i+3}) \times (\vec{P}_{i+2} - \vec{P}_{i+3})}{\|(\vec{P}_i - \vec{P}_{i+3}) \times (\vec{P}_{i+2} - \vec{P}_{i+3})\|_2}$$ ... (2)

$\vec{C} \times \vec{D} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ C_x & C_y & C_z \\ D_x & D_y & D_z \end{vmatrix}$ $s\vec{B} = s$

Google Ref.

Consider DDA Algorithm,
Digital Differential Algorithm

$y = bx + c$ ... (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).

$y_k = bx_k + c$ ... (1a)

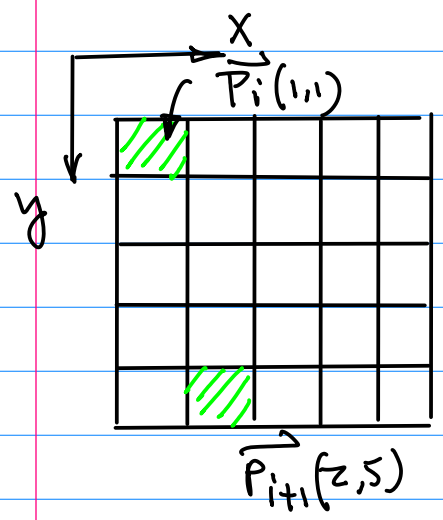
for $x_{k+1} = x_k + 1$, then

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

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

$= y_k + b$ ... (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)

$$y_{k+1} = y_k + b = 1 + 4 = 5$$

Gtt ! 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

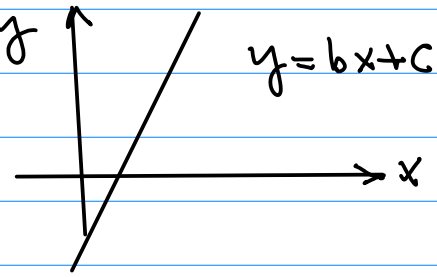
Point y_{k+1} on a pixel Location with a gap

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

. (Absolute value of

Consider $y = bx + C$, where

$$|b| > 1. \quad \cdots (1)$$



From Eqn(1),

$$y/b = x + C/b$$

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

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

$$x_k = \frac{1}{b} y_k - \frac{C}{b}$$

$$y_{k+1} = y_k + 1 \quad \cdots (3a)$$

$$x_{k+1} = \frac{1}{b} y_{k+1} - \frac{C}{b}$$

$$= \frac{1}{b}(y_k + 1) - \frac{C}{b}$$

$$= \frac{1}{b} + \underbrace{\frac{1}{b} y_k - \frac{C}{b}}_{x_k}$$

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

Diffuse Reflection on the interior points.

Going Back to the Same example.

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

for $k=2$.

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

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

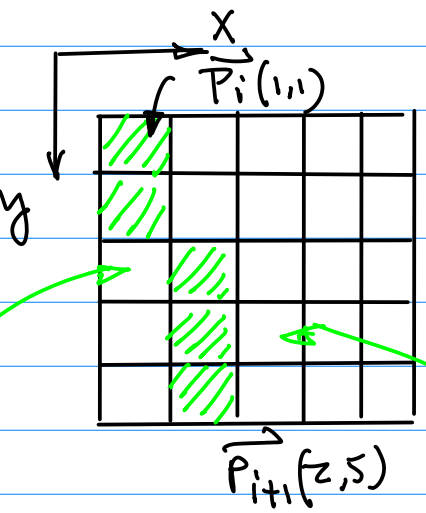


Fig. 1

for $y_3 = 3$,

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

$$= 1.5 \simeq 2$$

for $y_4 = 4$,

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

$$= 1.75 \simeq 2$$

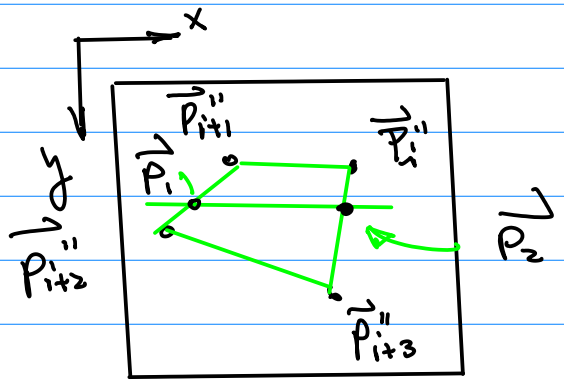


Fig. 2

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

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

Example: Decoration Algorithm.

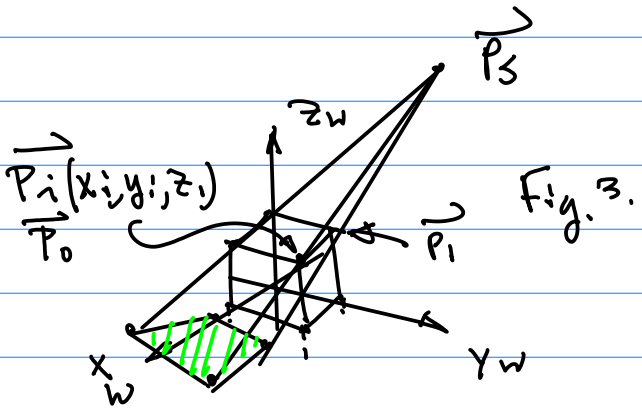


Fig. 3.

Decorate the Cube Surface.

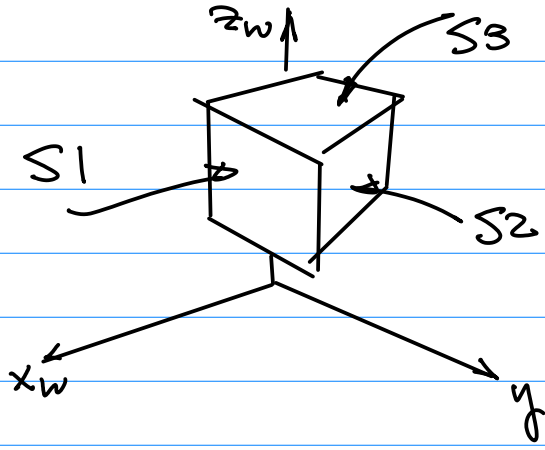


Fig. 4a

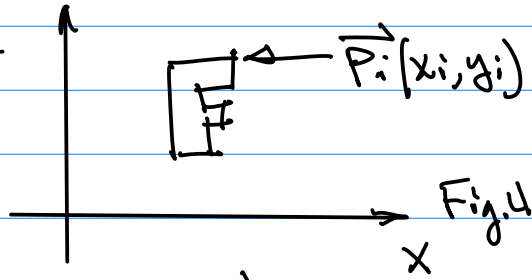


Fig. 4b.

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

Background: 3D Transforms.

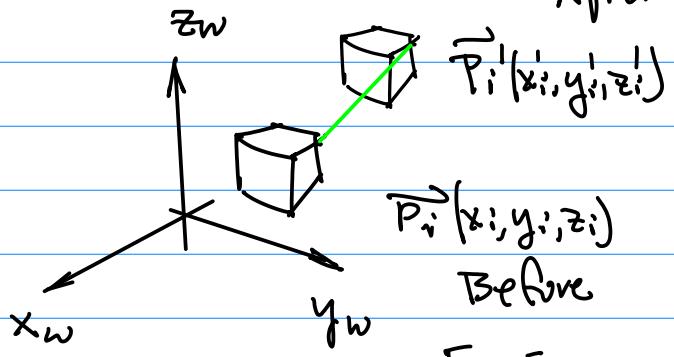


Fig. 5

$x_i' \stackrel{?}{=} x_i + \Delta x$

After Before

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

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

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

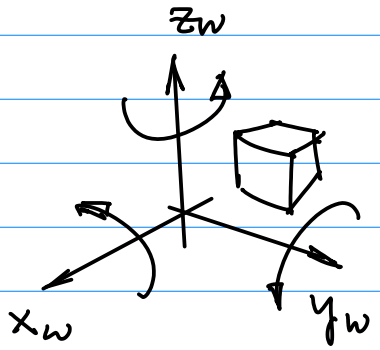


Fig.6

From 2D Rotation Matrix

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

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

Apply 26 (Wed).

Example: Linear Decoration Algorithm.

↑ Surface → plane

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

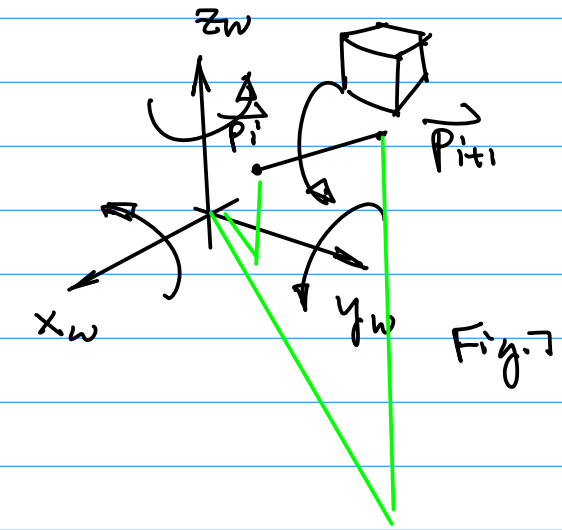


Fig.7

Rotation w.r.t. X_w-axis.

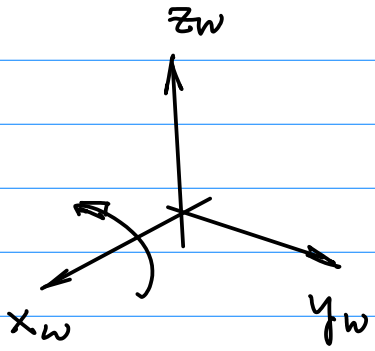


Fig.8

Methodology: Observe/Discover the independent variable which stays the constant. x_i →

Rotation on y_w-z_w plane.

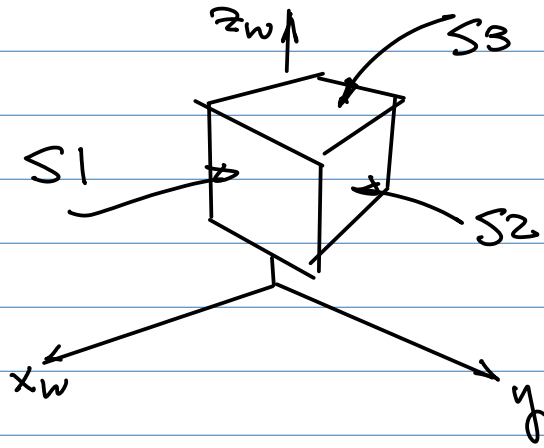


Fig.4a

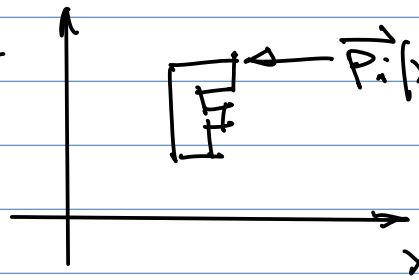


Fig.4b.

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.

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

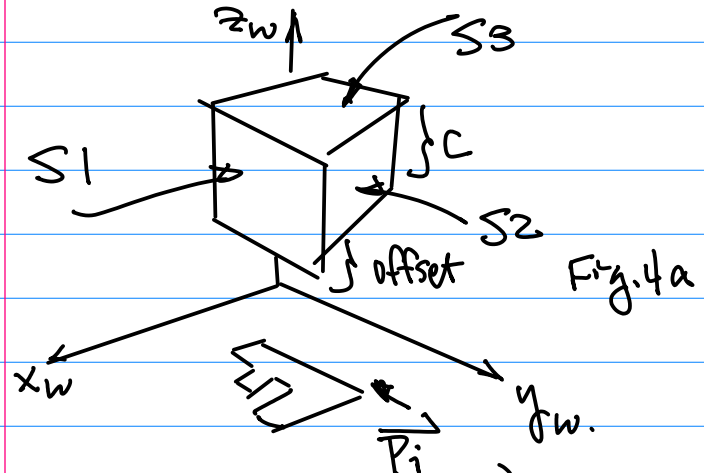


Fig. 4a

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;

2° Identify the indep. Variable & Function of the plane.

x_w-y_w

x_w (Indep.) V.S. y_w (Function)

After ? Before

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

Consider S1:

Find A parallel plane y_w-z_w.

Indep. V.S. Function

Variable are y_w V.S. z_w.

After ? Before

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

Consider S2

Parallel plane z_w-x_w

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.

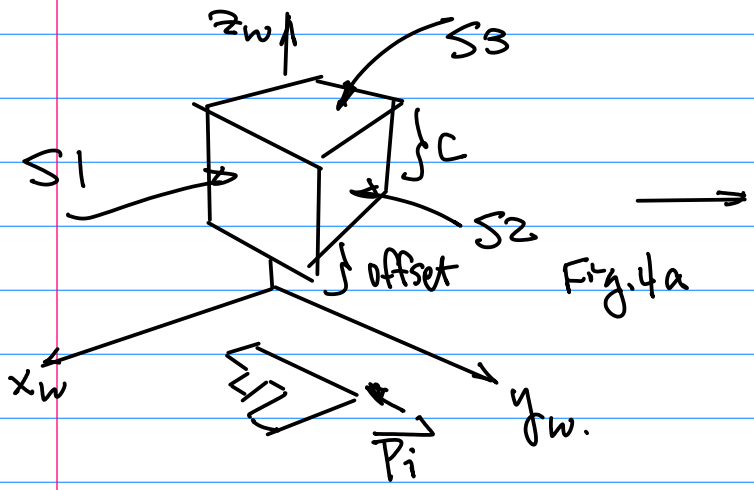
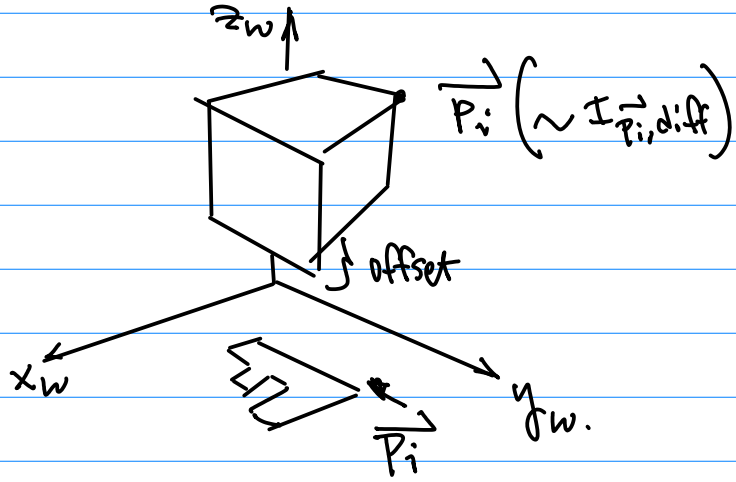
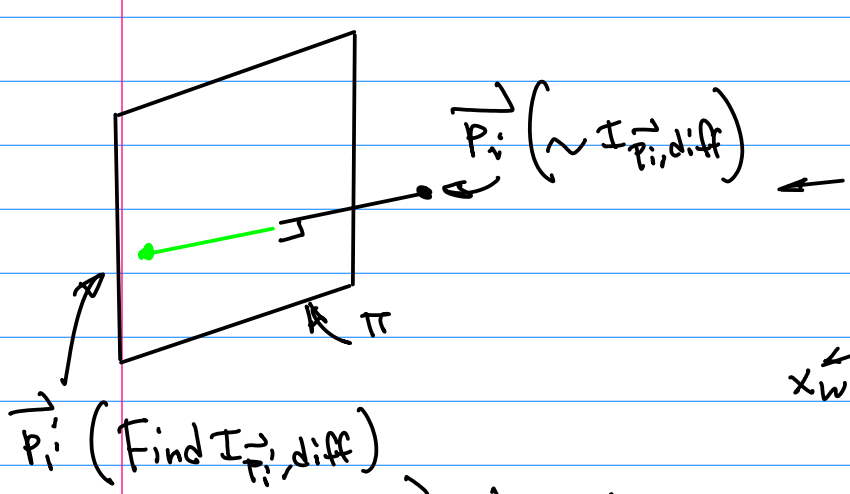
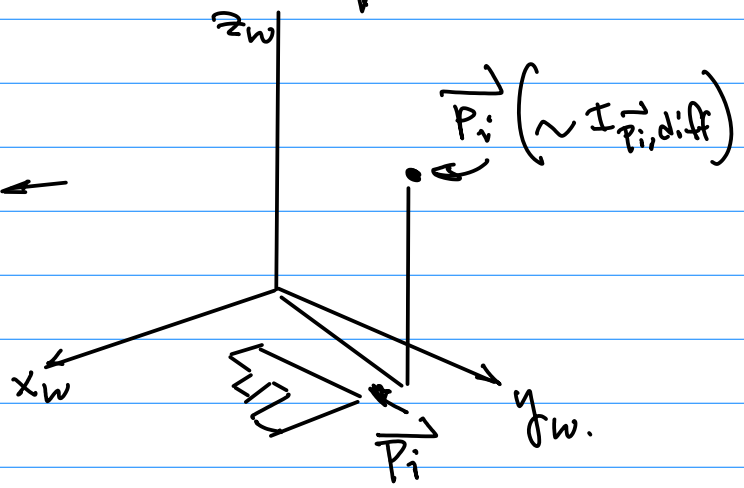


Fig.4a







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'')

↓

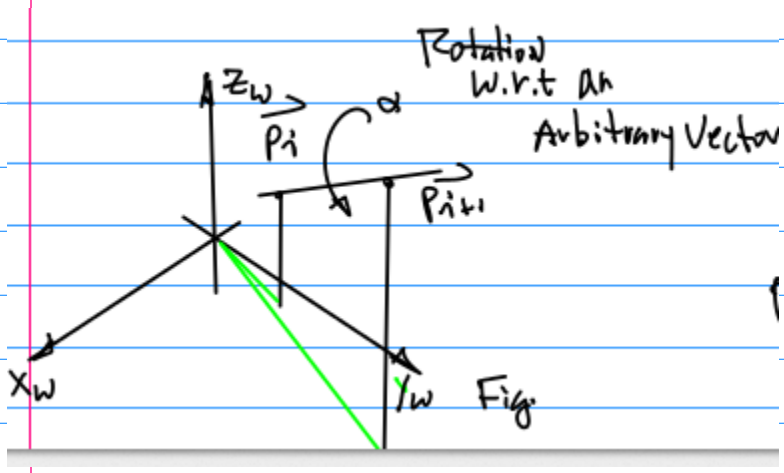
Copy diffuse reflection from $\vec{P_i}(x_i, y_i, z_i)$

2023-5-1

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

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

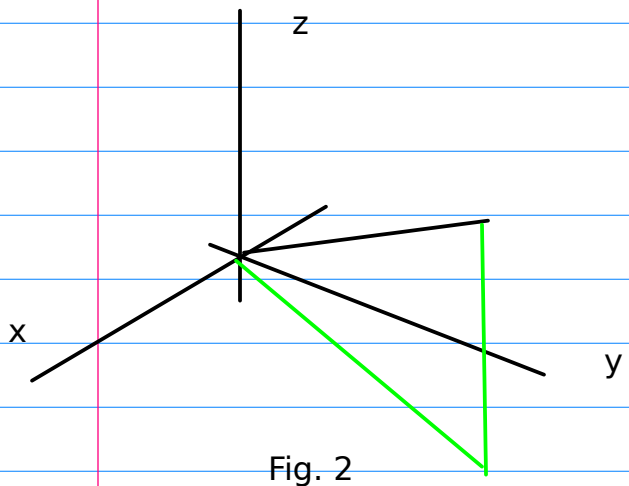


Fig. 2

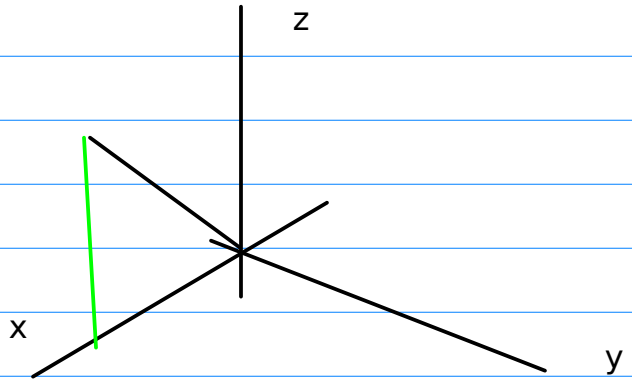


Fig. 3

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.

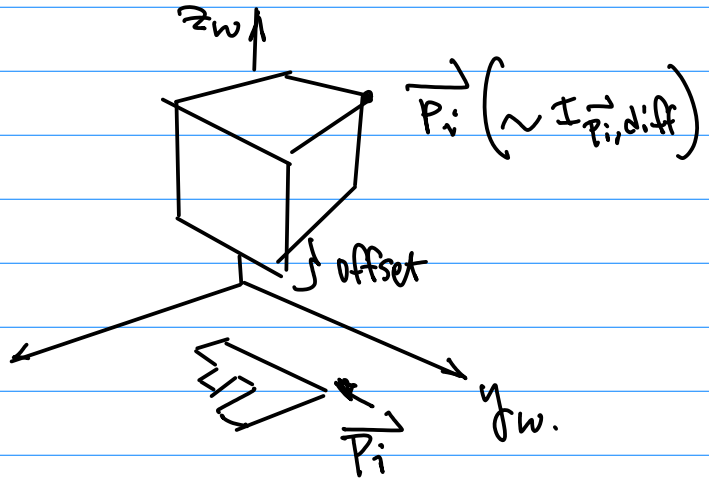
For the optional proejct (Bonus),
please use the following vector:

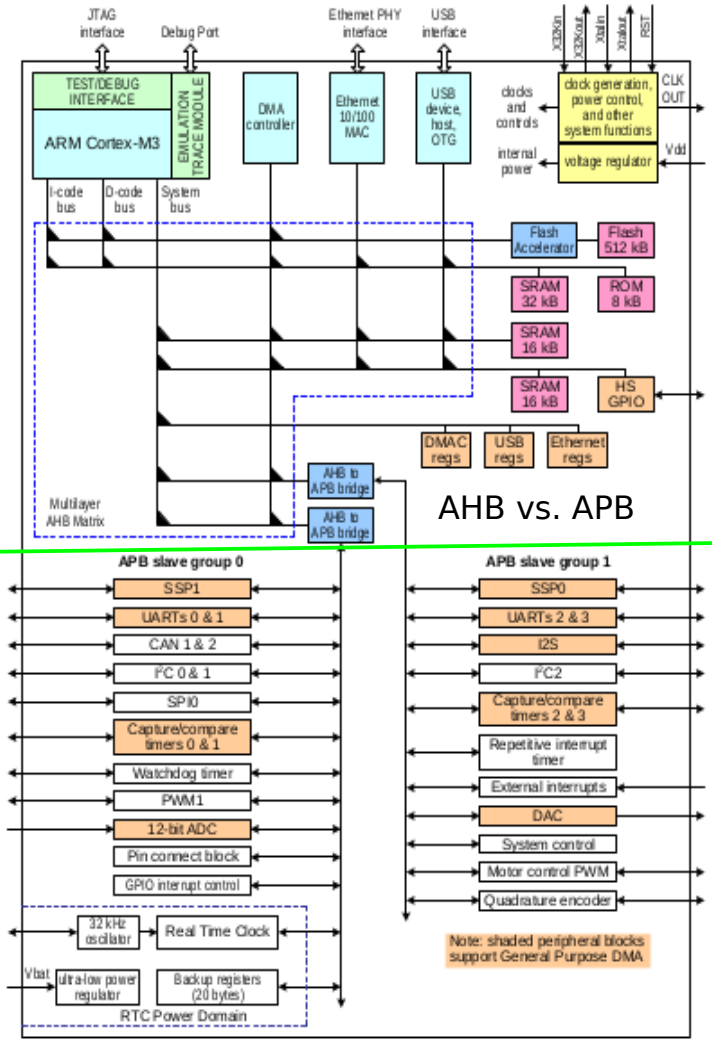
P_i(10, 10, 10) and
P_(i+1) (200, 200, 300)

Rotation by 5 degree;

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

May 8 (Monday)

2023S-101-note-part2b-benchmarking-cmp...

Note:
1. Bring the submitted project with the prototype system to the class for veriication and inspection.

2. Review session on the next lecture.

Today: Finish the controller (Display) design and finish the bench marking discussion.

Example: ref:

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

PP. 25, 26

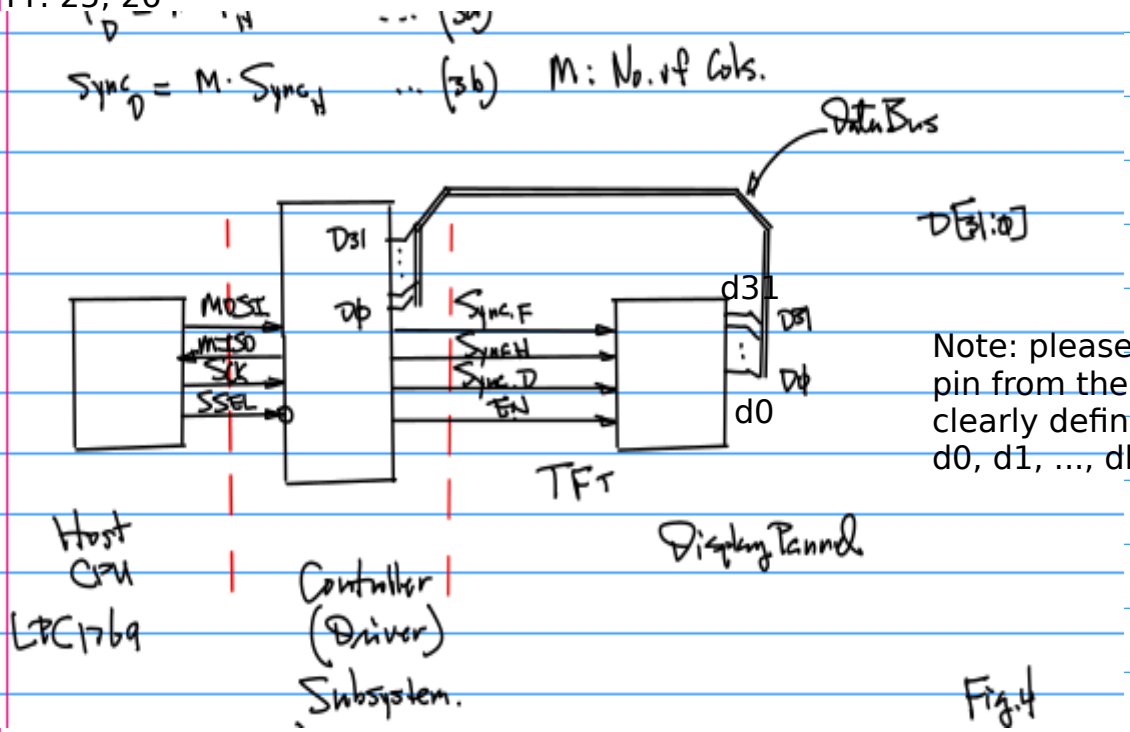
Frame rate: FPS
See the lecture note pp. 25-26
Eqn (1) on pp. 25;

Sync_F or f_F

Next, the horizontal sync or horizontal frequency, see eqn(2);
Sync_H or f_H

Then, define the timing for each each pixel
Sync_D or f_D
See eqn (3) on pp. 25-26

Example of using these equantions to perform analysis and evaluation of the engineering design



Note: please make sure each pin from the data bus is clearly defined with a label. d0, d1, ..., dk

The schematic design is based on the scanning principle of the display device.

See the lecture notes on pp. 25-26;

Suppose given a display device:
MxN = 160 x 120
assum full color display, e.g., 24 bit per pixel (pixel depth), BPP;
frame rate: 30 FPS;
Suppose we want to plot
a pixel at (123,25), find the
total time for this process.

Sol:
1. Analysis the design requirements
I(123,25)
the first argument is x for the col.
the 2nd one, y, must be for the row.

Time needed to reach to one row
before row 25

(a) the 1st is row 0;
(b) row 24 requires scanning 25 rows
before reaching to row 25;

T_segma_H = 25 * 1/Sync_H ... (4)

where

Sync_H = 120 * 30 (= N Sync_F)

Now, at the begining of the right row

T_segma_D = 123 1/Sync_D + 1/Sync_D
... (5)

Since the FPS = 30 Hz;
then we can solve for the above
equation.

Note: the schematics (see pp. 25-26)
which is required.

Example: Bench marking the system
for GE design to evaluate the CPU and
the entire system performance

Step 1. Inside CPU, with mem interface;

Step 2. Internal bus systsem to connect the peripheral device (SPI);

Step 3. Display device, Sync_F, Sync_H, Sync_D

Find the number of bits per second we can display

160 x 120 x 24 x 30

= 127 x 127 x 32 x 32

= 2^7 x 2^7 x 2^5 x 2^5

= 2^14 x 2 ^10

= 2^4 x 2^ 20

= 16 Mbps ... (1)

comparing 500K poly per second from the previous evaluation, to find out
if (1) is adquate to support 500K poly / second.

1 poly = 32 bits

then
500K x 32 (bit per second)

= 1 Mbps x 1/2 x 2^5

= 2^19 x 2^5 bps

= 2^4 x 2^20 bps

= 16 Mbps