

Sept. 7

Sept. 7.

Note: 1° LPC1768 from 2022S

Semester, Waiting List.

CANVAS. Scott.

2° LPC1768 Pin-to-pin
Comparable. (Mbed)a. Step1. MCUXpresso IDE
1768 Binary Code.Step2. "Firmware" Upload
the binary file to the
Flash. Need a prog

Step3. Interactive Debugging.

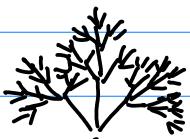
3° LPC11C24 Digi-Key in Stock.

LPC11U4

GPP/SPI, FLASH (On-Chip)
Size

1/8 of the size

Comparing to LPC1768/9



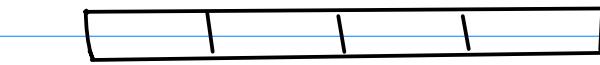
Homework (0 pt)

1. Form A Team By Wednesday.
4-Person2. Select/Finalize your target
platform. By the end of the week.

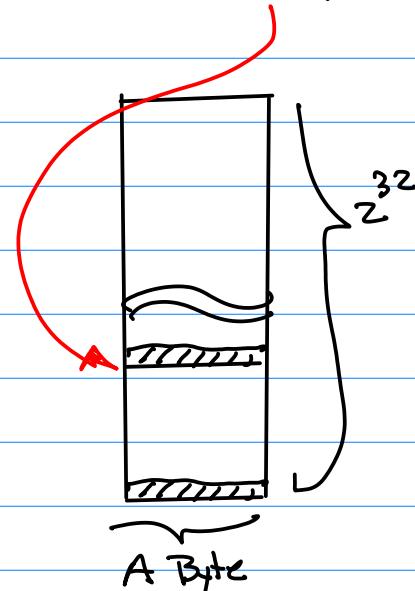
Example: RegisterFile

Special Purpose Registers
General Purpose Register

GP_x C_{DN}
T_{Root}
Prefix 3 Letters
for Port "x", x=0,1,2,3



GP_x C_{DN} its address is 32 Bits,
it maps to the memory
map.



Note: The Task of Init & Config
Can be realized by using HLL
(High Level Language), C/C++,
to deposit A Binary Pattern to that
Memory Location (Addr. is a pointer)

Sept. 7

For Example for Samsung ARm-11.

Consider:

11

PowerUp Address + PowerTakes.

Booting.

GPACON[11:8] GPACON[7:4]



GP_xCON its address is 32 Bits,
it maps to the memory
map.

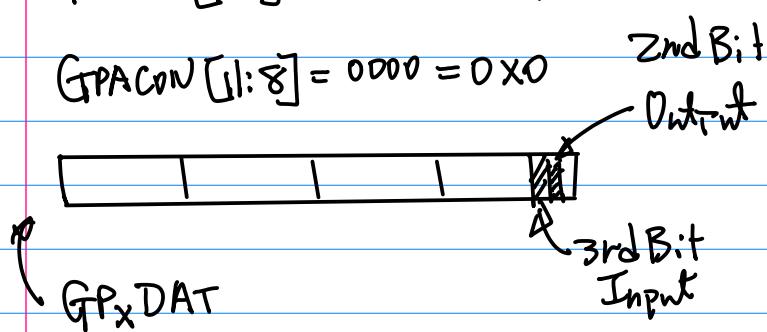
Design Requirements (Spec.)

1. 2nd Bit As An Output
2. 3rd Bit As An Input

To perform Init & Config.

$$\text{GPACON}[7:4] = 000 = 0x0$$

$$\text{GPACON}[11:8] = 0000 = 0x0$$



$$\text{GPACON}[3:0] = 0x10$$

Sept. 12 (Monday)

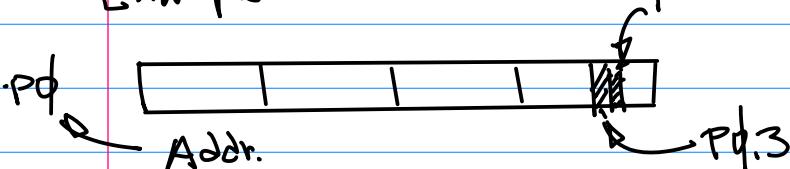
1. Homework (2pts)
2. Special Purpose Register

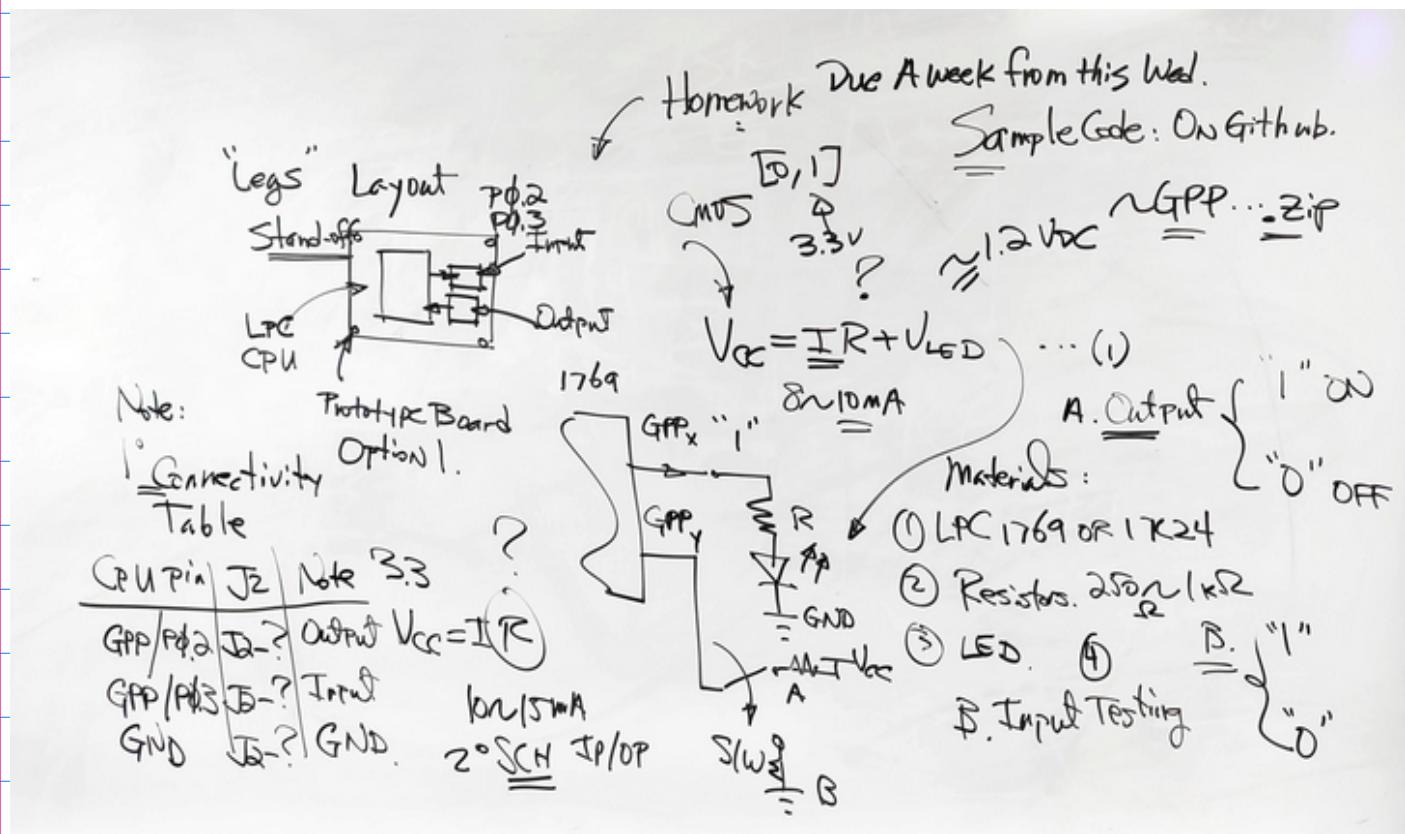
Note: Target platform.

LPC1768, LPC11C24

Example:

Pq.2





Sept. 14 (Wed).

Note: 1° Check Homework

Assignment on CANVAS.

Two Options | Prototype Board

Board A, Board B.

Topics today: IDE

- 1° GPP Software / Program
- 2° 2D Graphics Processing Engine Design.

Example: Set up the Expresso.

Key points:

- 1° Make sure Select Target Board LPC1769. (Ref. on github, 35 slides)

2° C/C++ Project Settings. →

"Semi-host"

3° Import LPC1769 patch.



1769 patch.zip

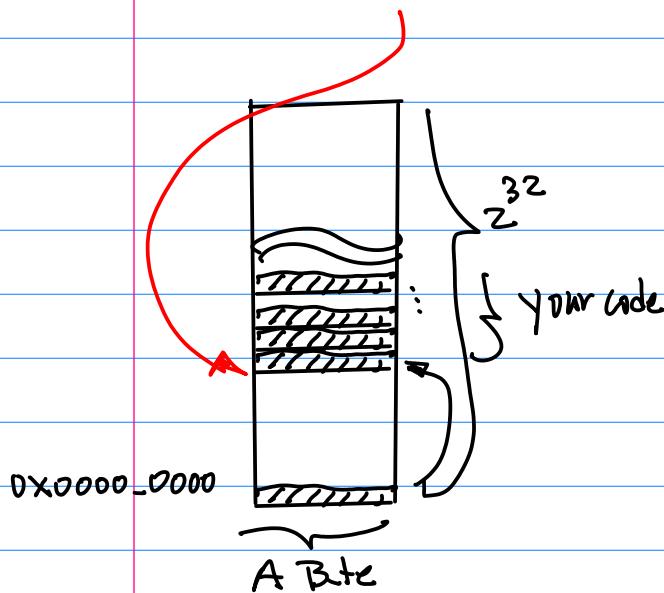
Note: 1° LPC1769 patch is already Config by NXP.

2° prob issue → fix:

Re connect DR Reboot.

Import GPIO project to your MCU Expresso.

Run "Debug", Once prob is detected then
Your Program (Binary)



Note: To uniquely define a line,
we can add a directional
vector, denoted as

Definition 1:

$$\vec{d}(x, y) \triangleq \vec{P}_{i+1}(x_{i+1}, y_{i+1}) - \vec{P}_i(x_i, y_i)$$

↓ ↓ ... (i)
 Ending pt. Starting pt.

Definition 2: (Line Eqn. in Vector Form)

$$\vec{P}(x, y) = \vec{P}_i(x_i, y_i) + \lambda \vec{d}(x, y) \dots (z)$$

$-\infty < \lambda < +\infty$

Consider G.E. Design.

Math. Formulation, for Vector Graphics.

Example:

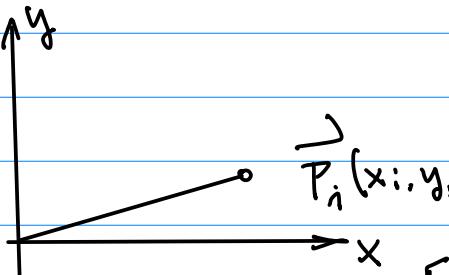


Fig. 1.

\vec{P} A Point $\rightarrow \vec{P}(x, y) \rightarrow \vec{P}_i(x_i, y_i)$
for $i = 0, 1, 2, \dots$

Also, a line

$$(x_i, y_i)$$

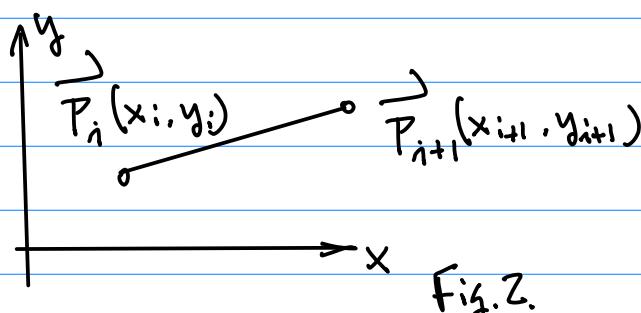


Fig. 2.

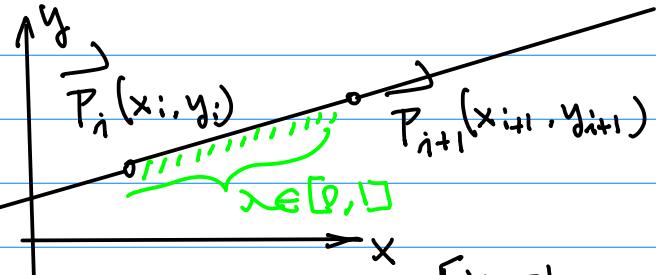


Fig. 2b.

Observation 1:

When $\lambda = 0$, Eqn (z) gives the
Starting pt. $\vec{P}_i(x_i, y_i)$

$\lambda = 1, \dots$.. Ending point
 $\vec{P}_{i+1}(x_{i+1}, y_{i+1})$

$0 < \lambda < 1$, $\vec{P}(x, y)$ Any Arbitrary
Pt Between $\vec{P}_i(x_i, y_i)$
and $\vec{P}_{i+1}(x_{i+1}, y_{i+1})$

Sept. 21. Today's Topics :

- 1° Check CNVAS for the Submission of the In-Class Exercise ;
- 2° Graphics Lib has been ported to LPC11C24, Ref. Code & PPT will be provided.

$M \times N$
 \uparrow \nwarrow \curvearrowright No. of Rows
 No. of Pixels per Row (Col).

2021S-105-CMPE240-2021-03-22-Note.pdf

$\vec{P}_0(x_0, y_0), \vec{P}_1(x_1, y_1), \dots, \vec{P}_3(x_3, y_3)$
 are defined as the same data pts
 on PPT5.

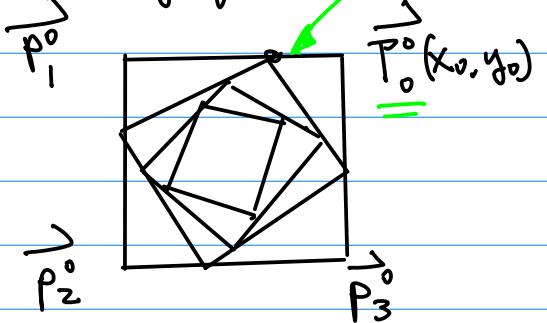
Example: Given the equation Below,

$$\vec{P}_i^j(x, y) = \vec{P}_i(x_i, y_i) + \lambda (\vec{P}_{i+1}(x_{i+1}, y_{i+1}) - \vec{P}_i(x_i, y_i)) \dots (1)$$

Let's Design A Graphic Algorithm

to Create A Set of Counter Clockwise (CCW)

Rotating Squares.



Step 1. Define the initial set of Data Points $\{\vec{P}_i(x_i, y_i) | i=0, 1, \dots, 3\}$

for Resolution of A Display

Device defined as $M \times N$

$\text{Max}(M, N) \leq 200$, Select the

Size of the Cube ≈ 50



Step 2. Get Line Equation, Based on

Eqn (1). First pt on Level 1

$$\vec{P}_0^1(x_0^1, y_0^1) = \vec{P}_0(x_0^0, y_0^0) +$$

$$\lambda (\vec{P}_1(x_1^0, y_1^0) - \vec{P}_0(x_0^0, y_0^0))$$

$$= (b_0, b_0) + \lambda ((10, b_0) - (b_0, b_0))$$

$$= (b_0, b_0) + \lambda (-50, 0)$$

From the Given Condition, CCW Rotation

Let $\lambda = 0.2$

Note: Based on the Above Calculation
 Can be conducted for the rest of the
 Pts, And Rest of the levels.

Step 3. Suppose we want to generate

10 Levels of the Rotating

Squares. Let Write C/C++
 for this Purpose.

From Eqn(1), we have

$$\begin{cases} x_i^{j+1} = x_i^j + \lambda (x_{i+1}^j - x_i^j) & \dots (za) \\ y_i^{j+1} = y_i^j + \lambda (y_{j+1}^j - y_i^j) & \dots (zb) \end{cases}$$

$$x[i][j+1] = x[i][j] + \text{lambda} * (x[i+1][j] - x[i][j]);$$

$$y[i][j+1] = y[i][j] + \text{lambda} * (y[i+1][j] - y[i][j]);$$

Consider Creating A Screen Saver

By Generating A tree.

Note: Level 0 : Tree Truck; Same Direction, Directional Vect. Same.

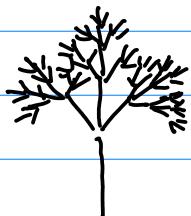
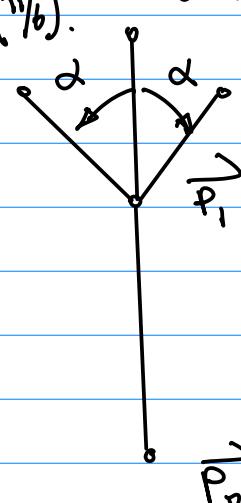
Level 1: Branch (Main) : Mag. Reduction By 70% $\lambda = 0.3$

Side Branch $\begin{cases} L(CCW), \text{Rotation by } \alpha (\pi/6). \\ R(CW), \alpha < 0 \end{cases}$

$\alpha > 0$: Positive

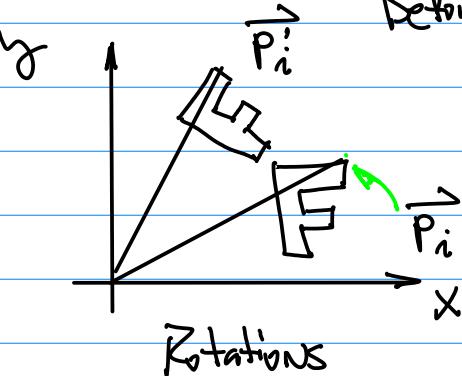
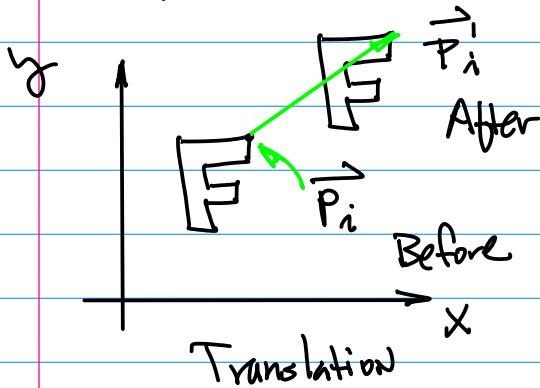
level 2, 3, ..., k.

Repeat Level 1 with Refacing Vector (p_i^+)
Updated Accordingly.



Background (2D Transformations)

Note: Mapping Between \vec{P}_i & \vec{P}_i' e.g.



Before & After.

CMPE240
Sept. 2b

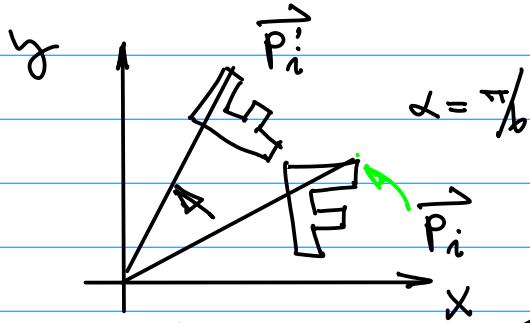
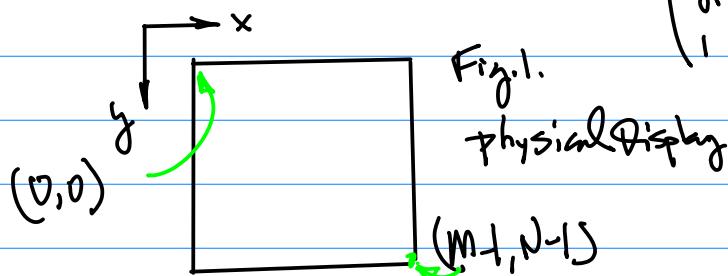
1b

Sept. 2b.

Rotation: 1° Positive Angle is defined as a Counter Clockwise Rotation;

2° Reference pt is defined as the Origin.

3° Physical Display (Coordinate System) v.s. Virtual Display (Virtual Coordinate System).



Rotations
Fig.2
Virtual Display.

Example: for the Rotation illustrated

in Fig.2.

$P_t(\text{After})$

$$\begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix}$$

? \doteq ϕ

$P^+(\text{Before}) \vec{P}_i$

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \dots (1)$$

$P_t(\text{After})$

$\vec{P}^+(\text{Before}) \vec{P}_i$

$$\begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & \dots & a_{23} \\ a_{31} & \dots & a_{33} \end{pmatrix}_{3 \times 3} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \dots (1b)$$

3D Vector, with One "Dummy" Dimension.

Rotation Matrix for Eqn (1b)

$$\begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \dots (2)$$

$$\left\{ \begin{array}{l} x'_i = x_i \cos\alpha - y_i \sin\alpha \dots (2-b) \\ y'_i = x_i \sin\alpha + y_i \cos\alpha \dots (2-c) \end{array} \right.$$

$$\left\{ \begin{array}{l} x_{\text{Prim}}[i] = x[i] * \cos(\text{alpha}) \\ - y[i] * \sin(\text{alpha}) \end{array} \right.$$

for the Reference of Doing C/C++ Coding.

Note: Reference Pt. for the Definition of Rotation. This Rotation Cannot be Simply defined Eq(2)

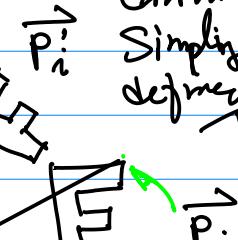


Fig.3

Sept. 26, 22

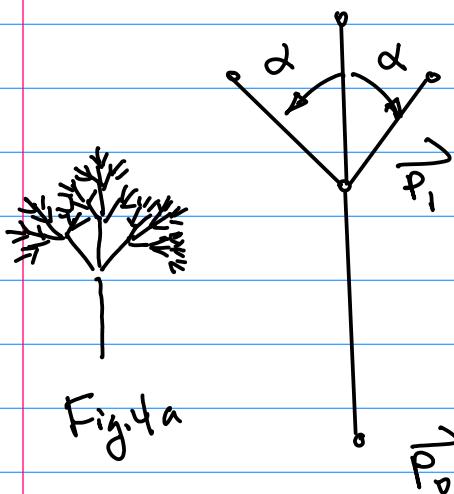


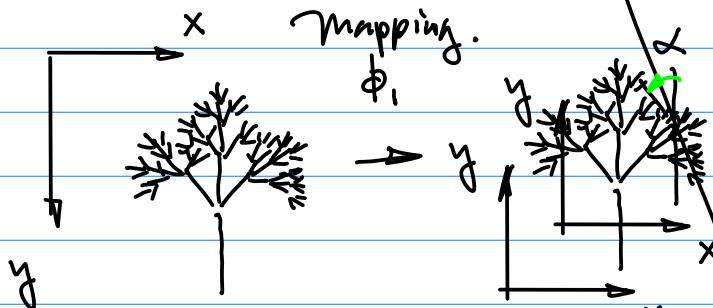
Fig. 4b.

 P_0

These are different Rotations than that in Eqn(z).

Consider the Composition of 2D Transformations.

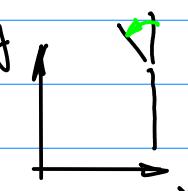
Example: Build/Design to Realize a 2D Tree Pattern in Fig. 4.



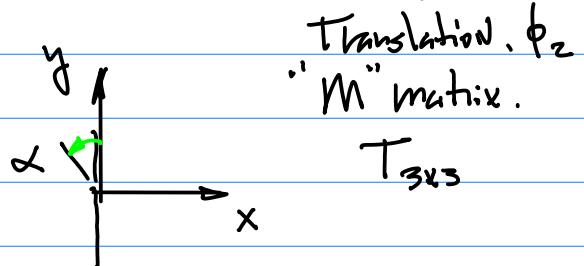
Physical Display

Fig. 5c. Rotation By α , CCW.

Fig. 5b. Virtual Display



mapping: Preprocessing —



Figs. d. Preprocessing By Translation.



Rotation Egn. (z).

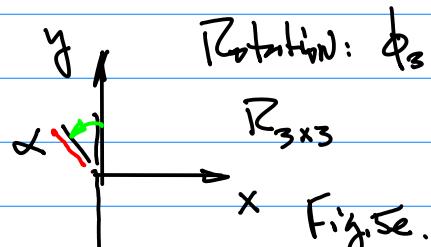


Fig. 5e.

Rotation: Done

Post Processing To move Back to its original Position.

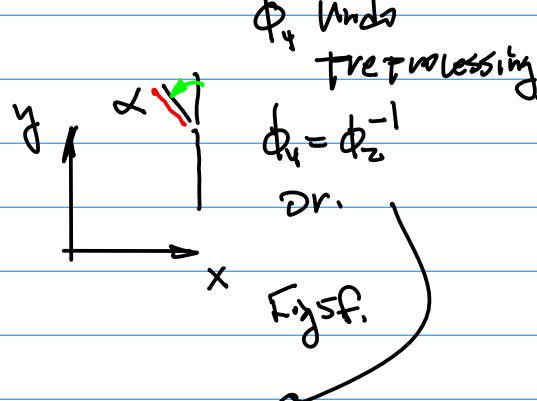


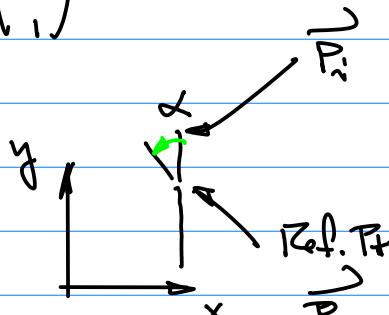
Fig. 5f.

$$\phi_4 = \phi_2 \Big|_{\phi_2 = T_{3 \times 3}} = T_{3 \times 3}^{-1} \dots (3)$$

Based on Step by Step Analysis.
(Analyze "Before" and "After"
Relationship).

Start at given
pt. to Be Rotated

$$\vec{P}_i = \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \dots (4)$$



Next. Preprocessing ϕ_2 , $T_{3 \times 3}$

To make $P_0(x_0, y_0)$ to overlap
with the origin (0,0).

$$T = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \dots (5)$$

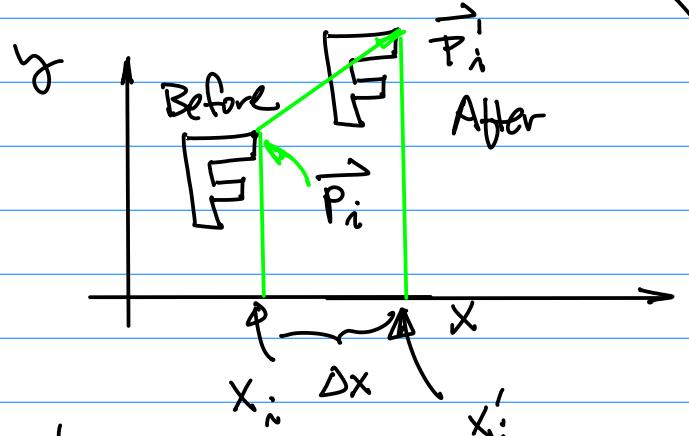
$$\rightarrow \begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} = T_{3 \times 3} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \dots (b)$$

Before After

$$= \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

$$x'_i = a_{11}x_i + a_{12}y_i + a_{13} \dots (bb)$$

Where $a_{13} = \Delta X$



Since, $a_{12}=0$, $a_{11}=1$, then

$$x'_i = x_i + 0 + \Delta X = x_i + \Delta X \dots (bc)$$

$$y'_i = y_i + \Delta Y \dots (bd)$$

Therefore

$$T_{3 \times 3} = \begin{pmatrix} 1 & 0 & \Delta X \\ 0 & 1 & \Delta Y \\ 0 & 0 & 1 \end{pmatrix} \dots (7)$$

Note: $\Delta X = -(x_i - x_0)$,

$$\Delta Y = -(y_i - y_0)$$

in Our Fig 5. Series.

Now, Rotation, $R_{3 \times 3}$.

Then, Post Processing. ϕ_4

$$M_{\phi_3} = T_{3 \times 3}^{-1} = \begin{pmatrix} 1 & 0 & -\Delta X \\ 0 & 1 & -\Delta Y \\ 0 & 0 & 1 \end{pmatrix} \dots (8)$$

Put All these together, we have

$$\begin{pmatrix} x_i'' \\ y_i'' \\ 1 \end{pmatrix} = T_{3 \times 3}^{-1} R_{3 \times 3} T_{3 \times 3} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \dots (9)$$

Rotated Post-processed Point.

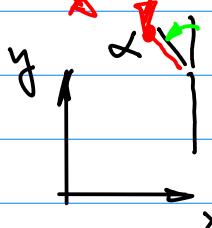


Fig. 7

Sept. 28 Wed.

Note: Conclusion on 2D Transforms

is given Composition of
in the Above Eqn. (9), Coding/Implementation
in C/C++.

T_Σ

$$\begin{pmatrix} x_i'' \\ y_i'' \\ 1 \end{pmatrix} = \overbrace{T_{3 \times 3}^{-1} R_{3 \times 3} T_{3 \times 3}}^{T_\Sigma} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

T_Σ is 3×3 matrix



Last Row of $T_\Sigma (3 \times 3)$ is
 $(0, 0, 1)$ No Need for
Coding.

Only the top 2 Rows from
the matrix matter.

Step by step Derivation is also
provided at the Lecture Notes.

$$x_L[i] = \cos\theta * x[i] - \sin\theta * y[i] \\ + \Delta x \cos\theta - \Delta y \sin\theta - \Delta x$$

Similarly for y_L ,

$$y_L[i] = \sin\theta * x[i] + \cos\theta * y[i] \\ + \Delta x \sin\theta + \Delta y \cos\theta - \Delta y$$

Note: Work on
Virtual-to-physical
mapping

Notation is introduced to
Keep track if the new point
is generated By CCW or CW

\vec{P}_i \rightarrow Land

$\vec{P}_{i,l}$ or $\vec{P}_{i,r}$
Enumeration of the
Points.

$\vec{P}_{i,l}$ or $\vec{P}_{i,r}$

(l - left - CCW,
r - Right - CW)

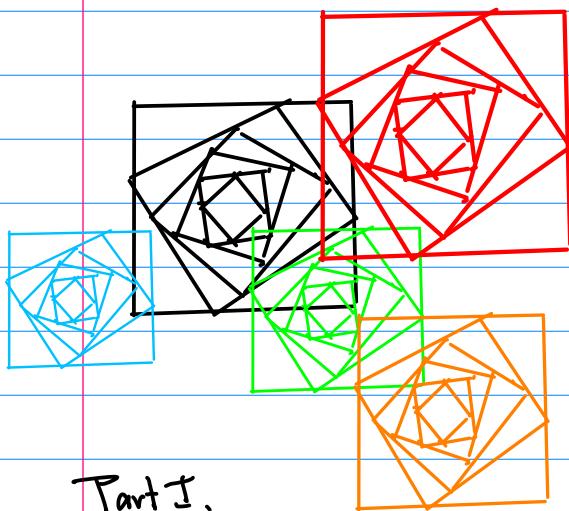
1st Project, 2D Graphics Engine

{ 2D Graphics Screen Saver
a collection of Rotating
Squares.

CompE240

Sept. 28, 22

Z0



Interface Between the LPC1769 and LCD is By SPI (Serial Peripheral Interface)

Fig. 1
(Level 10 or higher)



Part II. Trees. Optional for Artistic Presentation.

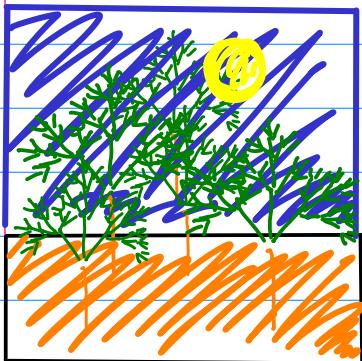


Fig. 2

"3+1" pins for SPI.

- 1. MOSI Output
- 2. MISO Input
- 3. SCK (Clock) Output
- 4. SS (CS) Output

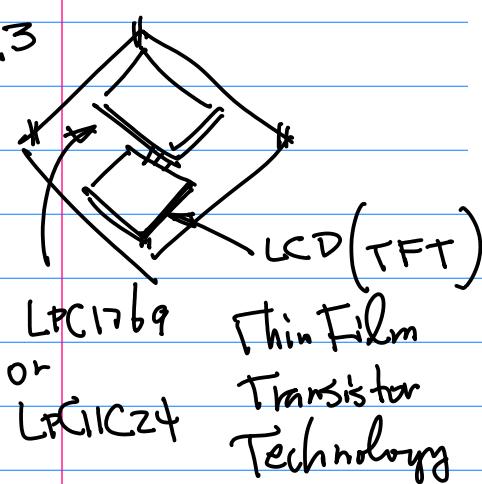
Now, consider LPC1769, CPU Datasheet
#p431, CR0

2021F-107b-sch-#LPC...

Consider the Graphics Engine Design from System View.

NXP Semiconductors

Fig. 3



(SSPOCR0)

CR0



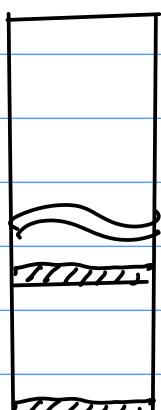
Addr. 0X4008-0000

UM10360
Chapter 18: LPC176x/5x SSP0/1

a. SSP b. CR0, CR1

Table 371: SSPn Control Register (SSP0CR0) address 0x4008 8000 (SSP1CR0 - 0x4003 0000) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	DSS		Data Size Select. This field controls the number of bits transferred in each frame. Values 0000-0010 are not supported and should not be used.	0000
		0011	4-bit transfer	



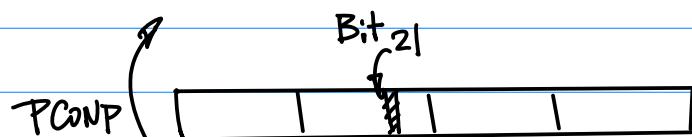
Find its
memory location at
which memory
Bank?

A Byte

Oct. 5 (Wed)

Example: SPR (for SSP/SPI)
CR₀.

20	-	Reserved.
21	PCSSP0	The SSP0 interface power/clock control bit.
22	PCTIM2	Timer 2 power/clock control bit.



Note: 2nd SPR: PCLKSEL

a. PCONP, Naming Convention "LPC_SC->PCONP"
Source Code from DrawLine ~ (github)

```

147  ** parameters:      None
148  ** Returned value:  None
149  ***
150  ****
151  void SSP0Init(void)
152  {
153      uint8_t i, Dummy=Dummy;
154
155      /* Enable AHB clock to the SSP0. */
156      LPC_SC->PCONP |= (0x1<<21);
157
158      /* Further divider is needed on SSP0 clock.
159      LPC_SC->PCLKSEL1 &= ~(0x3<<10);
160

```

Note: Bitwise op
to just (only)
Set Bit 21

PP.58, Table 41

b. CPU Datasheet, for PCONP.
C. Code(15b)

RTC interrupt is generated.

4.8.9 Power Control for Peripherals register (PCONP - 0x400F C0C4)

The PCONP register allows turning off selected peripheral functions for the purpose of saving power. This is accomplished by gating off the clock source to the specified peripheral blocks. A few peripheral functions cannot be turned off (i.e. the Watchdog timer, the Pin Connect block, and the System Control block).

CMPE240

Oct. 5th, 2

22

Table 41. Peripheral Clock Selection register 1 (PCLKSEL1 - address description)

Bit	Symbol	Description
1:0	PCLK_QEI	Peripheral clock selection for the Quadrature Encoder Interface.
3:2	PCLK_GPIOINT	Peripheral clock selection for GPIO interrupts.
5:4	PCLK_PCB	Peripheral clock selection for the Pin Connect
7:6	PCLK_I2C1	Peripheral clock selection for I2C1.
9:8	-	Reserved.
11:10	PCLK_SSP0	Peripheral clock selection for SSP0.
13:12	PCLK_TIMER2	Peripheral clock selection for TIMER2.

Note: PINSEL

```

161 /* P0.15~0.18 as SSP0 */
162 LPC_PINCON->PINSEL0 &= ~(0x3UL<<30);
163 LPC_PINCON->PINSEL0 |= (0x2UL<<30);
164 LPC_PINCON->PINSEL1 &= ~((0x3<<0)|(0x3<<2)|(0x3<<4));
165 LPC_PINCON->PINSEL1 |= ((0x2<<0)|(0x2<<2)|(0x2<<4));
166

```

PP117.

8.5.1 Pin Function Select register 0 (PINSEL0 - 0x4002 C000)

The PINSEL0 register controls the functions of the lower half of Port 0. The direction control bit in FIO0DIR register is effective only when the GPIO function is selected for pin. For other functions, the direction is controlled automatically.

Table 80. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit description

PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	R v
1:0	P0.0	GPIO Port 0.0	RD1	TXD3	SDA1	0
3:2	P0.1	GPIO Port 0.1	TD1	RXD3	SCL1	0
5:4	P0.2	GPIO Port 0.2	TXD0	AD0.7	Reserved	0
7:6	P0.3	GPIO Port 0.3	RXD0	AD0.6	Reserved	0
9:8	P0.4 ^[1]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2.0	0
11:10	P0.5 ^[1]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2.1	0
13:12	P0.6	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2.0	0
15:14	P0.7	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2.1	0
17:16	P0.8	GPIO Port 0.8	I2STX_WS	MISO1	MAT2.2	0
19:18	P0.9	GPIO Port 0.9	I2STX_SDA	MOSI1	MAT2.3	0
21:20	P0.10	GPIO Port 0.10	TXD2	SDA2	MAT3.0	0
22:21	P0.11	GPIO Port 0.11	RVD2	SDA2	MAT3.1	0

(1)

{

(2)

{

a pin. For other functions the direction is controlled automatically.

Table 81. Pin function select register 1 (PINSEL1 - address 0x4002 C004) bit description

PINSEL1	Pin name	Function when 00	Function when 01	Function when 10	Function when 11
1:0	P0.16	GPIO Port 0.16	RXD1	SSEL0	SSEL
3:2	P0.17	GPIO Port 0.17	CTS1	MISO0	MISO
5:4	P0.18	GPIO Port 0.18	DCD1	MOSIO	MOSI
7:6	P0.19 [1]	GPIO Port 0.19	DSR1	Reserved	SDA1
9:8	P0.20 [1]	GPIO Port 0.20	DTR1	Reserved	SCL1

Note: FIODIR (Direction), e.g. Input/Output is defined)

```
167 #if !USE_CS
168   LPC_PINCON->PINSEL1 &= ~(0x3<<0);
169   LPC_GPIO0->FIODIR |= (0x1<<16); /* P0.16 defined as GPIO and Outputs */
170 #endif
```

Note: a. Code → Tech. Spec. (Interpretation)

```
/* Set DSS data to 8-bit, Frame format SPI, CPOL = 0, CPHA = 0, and SCR is 15 */
LPC_SSP0->CR0 = 0x0797;
```

0000:0111:0000:0111
 SCR SPI 8 bit

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

Consider Display Device Interface.

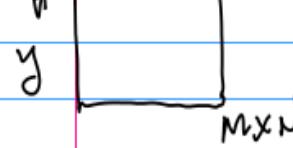
Background: Low Level Design of GUI
 } Hardware Driver Design.
 } Physical Display VS. Virtual Display.

Example: Physical Display VS. Virtual Display.

pp 19. from ref.
 (github, Notes ~ 105 ~)

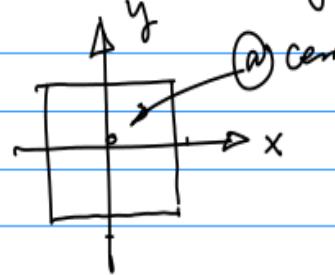
Physical Display VS. Virtual Display

X (P/I)



SPI LCD
Physical

No. of
M: Row
N: Col.



Mapping Function

Fig. 1.

(x_v, y_v) for Virtual
 (x_p, y_p) for Physical ~

Oct.10, 22

$$x_p = x_v + \frac{m}{2} \dots (7)$$

\downarrow in direction
6 offset

$$y_p = -y_v + \frac{N}{2} \dots (8)$$

(half of the
No. of Rows)

$$\left[\begin{array}{l} x_p = x_v + \frac{m}{2} \dots (7) \\ y_p = -y_v + \frac{N}{2} \dots (8) \end{array} \right]$$

$$\left[\begin{array}{l} x_p = x_v + \frac{m}{2} \dots (7) \\ y_p = -y_v + \frac{N}{2} \dots (8) \end{array} \right]$$

Oct.10 (Monday) Due Oct. 17th (Monday)

Homework: In-Class Demo. 1 pt.

Requirements: 1. Prototype Board with LCD Display. 2. Execution of 2D Graphics Processing (2D Screen Savers And VR Trees).

Example: Suppose we have a design

which gives $\vec{P}_i(x_i, y_i) = (-5, 10)$
in the virtual coordinate system.

Given: 1. Physical Display with

Resolution $M \times N$ (640×480),

Find the New point $\vec{P}_i(x'_i, y'_i)$

After the transformation from the
Virtual Display to the physical Display.

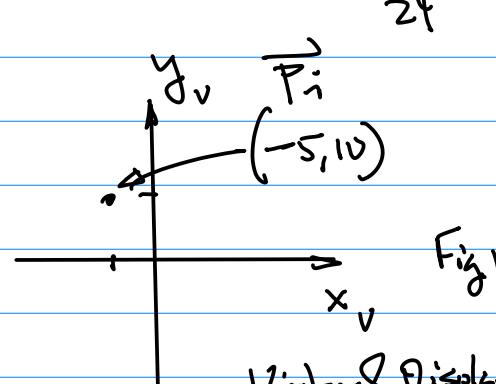


Fig.1.

Virtual Display.

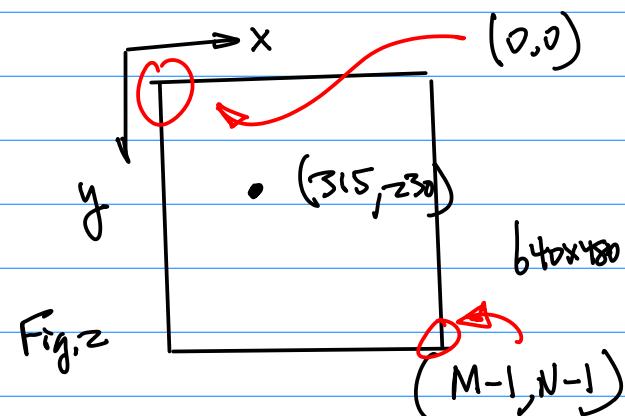


Fig.2

\Rightarrow : From Eqn(7) we have

$$x_p = x_v + \frac{m}{2} \Big|_{M=640}$$

$$= x_v + \frac{640}{2} = x_v + 320$$

where we have $x_v = -5$, so

$$x_p = x_v + 320 = -5 + 320$$

$$= 315 ;$$

For y_p from Eqn(8).

$$y_p = -y_v + \frac{N}{2} \Big|_{N=480}$$

$$= -y_v + \frac{480}{2} = -y_v + 240$$

$y_v = 10$, hence,

$$y_p = -10 + 240 = 230 ;$$

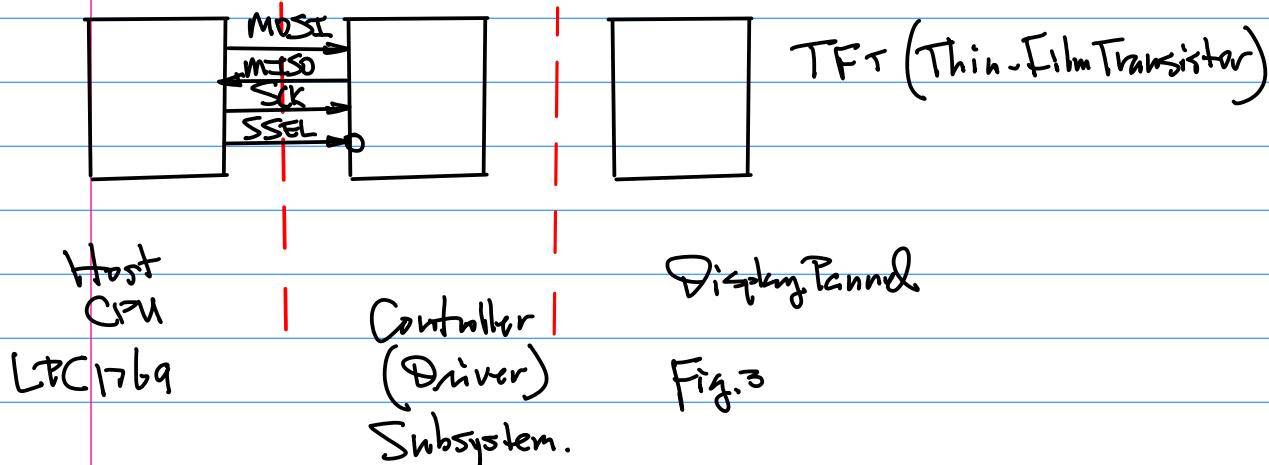
Now, Consider the Design of
Display Driver (Hardware)

CmpE240

LCD Display
module

Oct.10, 22

25



2nd LPC1769 For Example
(Not Most Cost Effective Solution)

The Hardware Features of T.F.T. Display:

- Sync. Signals
 - Frame Sync. Example: 30 F.P.S. $\text{Sync. F} \rightarrow f_F = 30 \text{ Hz} \dots (1)$
 - Horizontal Sync. for $M \times N$ Resolution, it Repeats N Times Per frame.
 - Data Sync. (Pixel) for $M \times N$ Resolution, Per Each Line, it Repeats M



Time to Scan Each Row is determined by Horizontal Sync.

2. Data Bus (Bi-Directional): 24 bits or higher, 32 Bits Common

3. Control Signals.

- Backlight
- Enable

Note: Design Hardware Counter, Counts by N. $\text{PCLK} \rightarrow \text{Sync. F}$

Consider the Relationships Between Sync_F & Sync_H , Denoted as f_H

$$f_H = N \cdot f_F \quad \dots (za)$$

$$\text{Sync}_H = N \cdot \text{Sync. F} \quad \dots (zb)$$

CMPE240
Oct. 10, 22

2b

Consider Sync. D V.S. Sync. H.

$$f_D = M \cdot f_N \quad \dots (3a)$$

$$\text{Sync}_D = M \cdot \text{Sync}_H \quad \dots (3b) \quad M: \text{No. of Cols.}$$

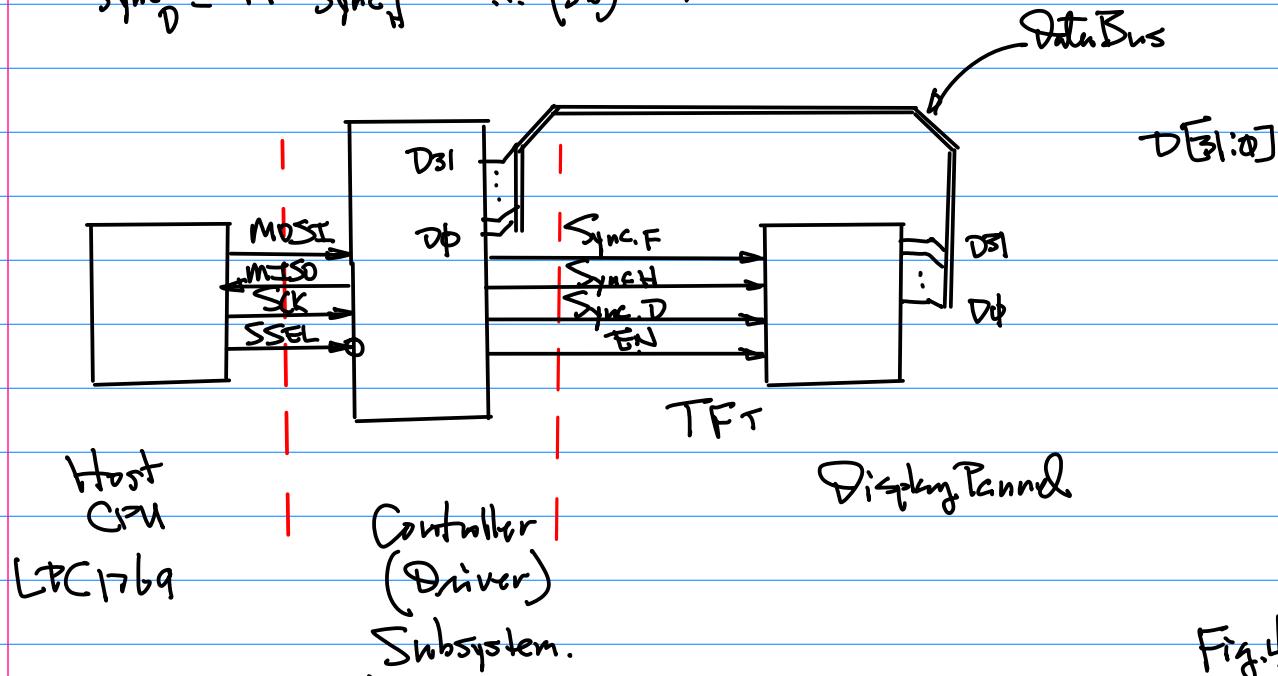
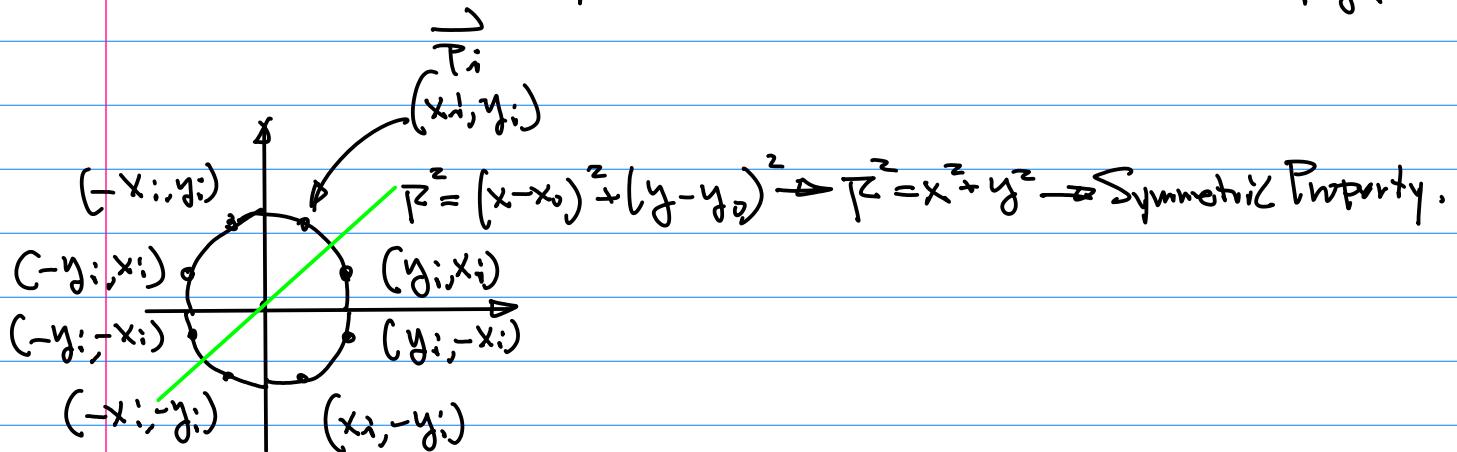
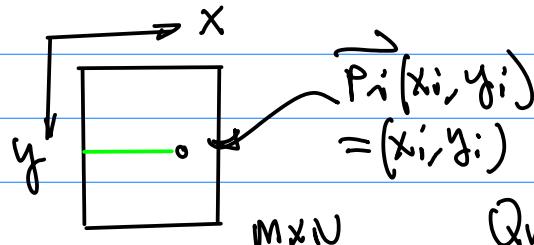


Fig.4



Oct. 12 (Wed)

Example:



Calculate the Location Information
Based on Timing (Sync. F, Sync. H, Sync. D)

Question: How much time does it take
for the Scanning to Reach line y_{i-1}

Assume FrameRate 30FPS.

$$T_H = \frac{1}{Sync_H} \quad \dots \quad (1)$$

And $T = (y_{i-1}) T_H \quad | \quad T_H = \frac{1}{Sync_H}$
 $= (y_{i-1}) / Sync_H. \quad \dots \quad (2)$

From this line at col = 0 Location,
 we start counting the time needed
 to get ready to plot the pixel

$$T_D = \frac{1}{Sync_D} \quad \dots \quad (3)$$

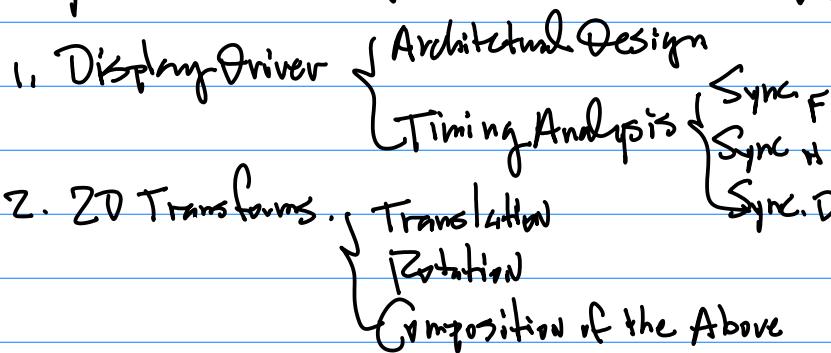
$$Sync_D = M Sync_H$$

$\therefore T' = (x_{i-1}) T_D = (x_{i-1}) / M Sync_H$
 $\dots \quad (4)$

Hence

$$\begin{aligned} T_{\Sigma} &= T + T' \\ &= (y_{i-1}) / Sync_H + (x_{i-1}) / (M Sync_H) \\ &\dots \quad (5) \end{aligned}$$

In Summary: 2D G.E. Design.



3. Vector Graphics Manipulation

$$\vec{P}(x, y) = \vec{P}_i(x_i, y_i) + \lambda (\vec{P}_{i+1}(x_{i+1}, y_{i+1}) - \vec{P}_i(x_i, y_i))$$

4. Primitive Graphics Basic Building Blocks.

Line Segment, Arcs (Circle), Elliptics etc.

No multiplication

$$y = ax + b$$

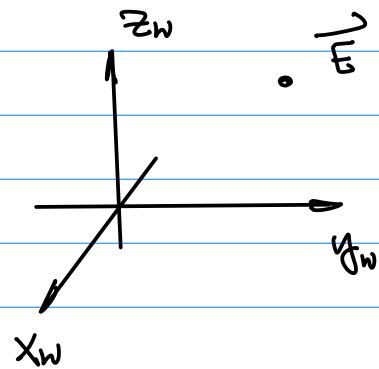
$$R^2 = (x-x_0)^2 + (y-y_0)^2$$

Consider 3D G.E. (Graphics Engine Design).

Ref. [github/malik/cmpe240](https://github.com/malik/cmpe240) . . .

... note ... May ...

Fig.1.



World Coordinate System.

RH: Right-Hand System Defined By Vector Cross Product
 $\vec{x}_{w1} \times \vec{y}_{w1}$

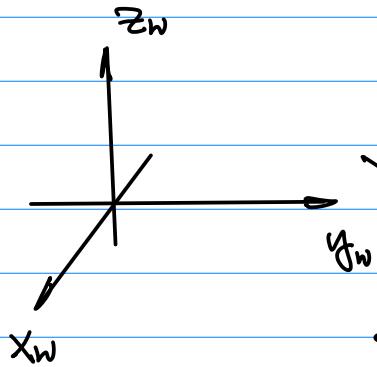


Fig. 2

place a virtual
Camera at $\vec{E}(x_e, y_e, z_e)$ Fig. 5

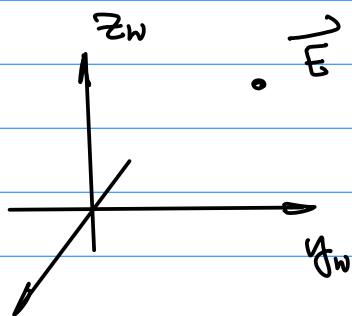
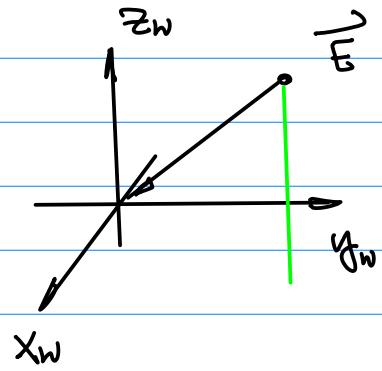
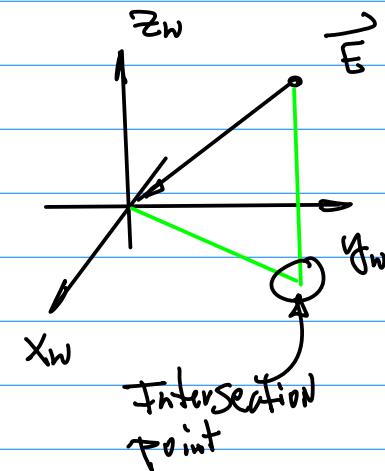


Fig. 3

Camera for
its Simplicity.
Its properties
include:

1° An Enclosure,
and a pin-Hole,
and projection plane.



2° Only light reaches
the projection plane is the
light passing through the
pin-Hole.

3° The Camera is always Looking Towards
the origin of the $x_w-y_w-z_w$ System.

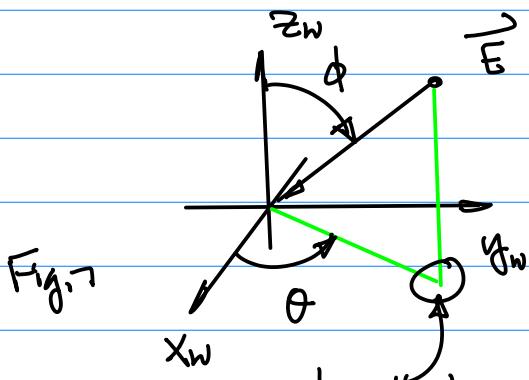


Fig. 7

Angle θ : ON x_w-y_w plane

C.C.W as a Positive Angle

Angle ϕ : Formed By \vec{E} and z_w Axis,

$\vec{E}(x_e, y_e, z_e)$

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

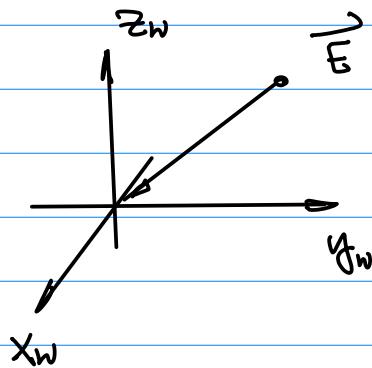
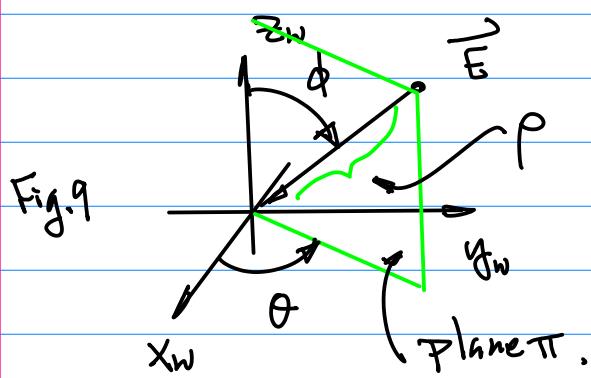
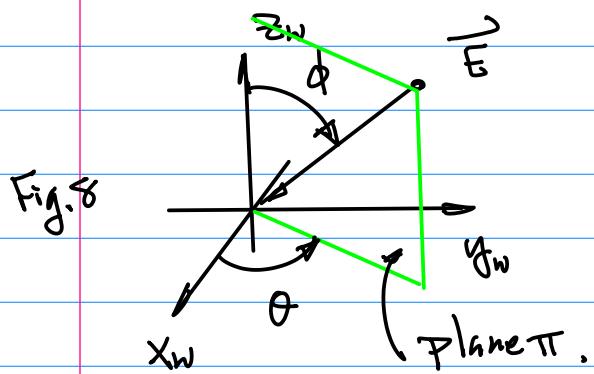


Fig. 4

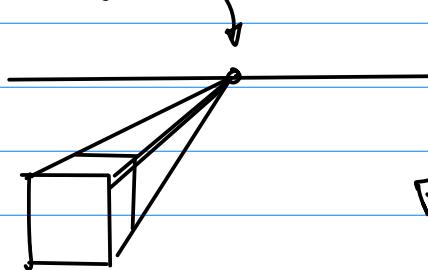
Oct. 17 (Monday).



Example: Perspective Projection.

Projection of 3D objects onto 2D plane.

b. Vanishing pt



a. horizon

Fig. 1

c. Object $\{P_i | i=0, 1, 2, \dots, N-1\}$, Connect Each pt P_i to the Vanishing pt.

d. Use Parallel plane to Cut the Vector Lines, to form a 3D object.

Let's consider the Viewer Coordinate

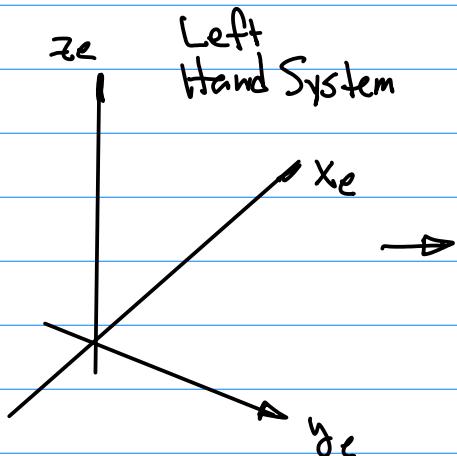
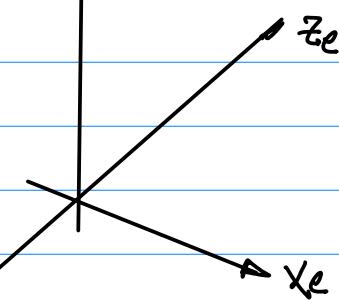
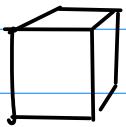
 $x_v - y_v - z_v$ Re-arrange it
Add Projection plane

Fig. 10

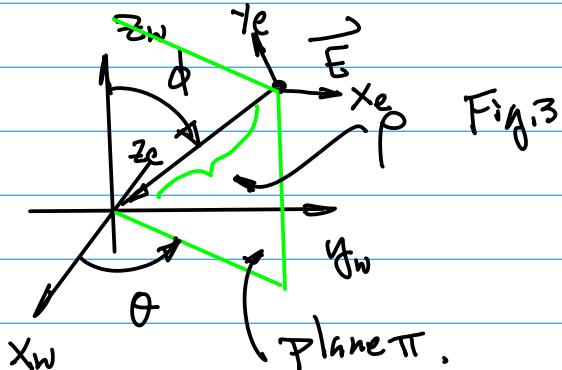
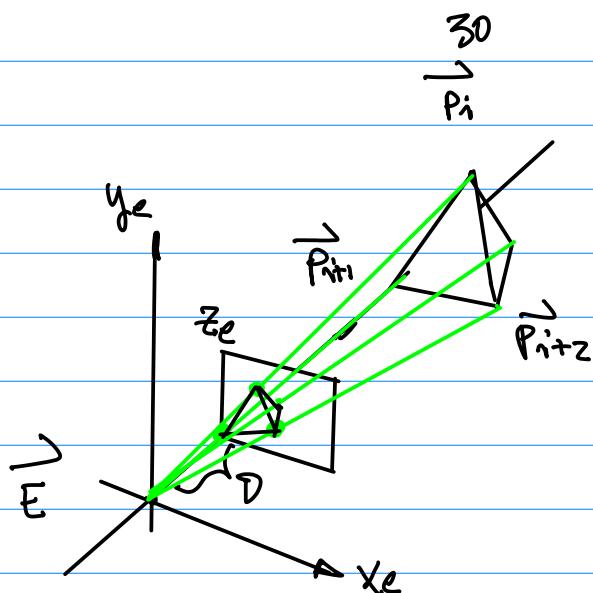
CMPE240

Oct. 17, 22



Perspective
Fig. 2. projection.

Note: The $x_v-y_v-z_v$ System, Virtual Camera.

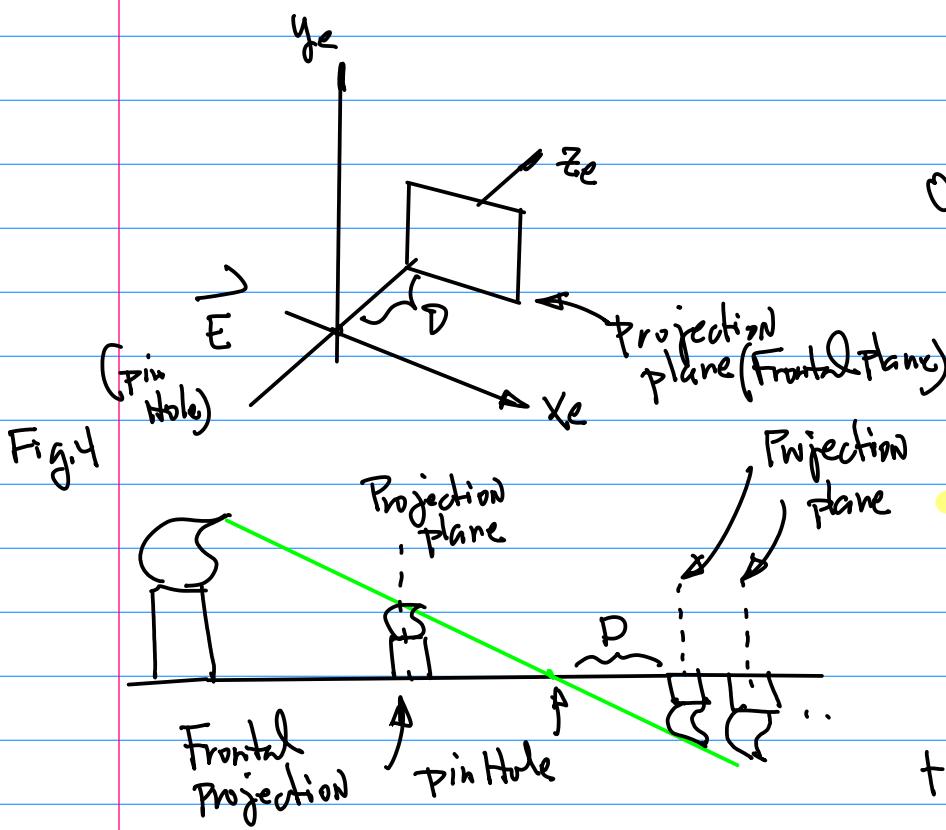


Transformation Pipeline
 $x_w-y_w-z_w \rightarrow x_v-y_v-z_v$
Perspective Projection

$$x_w-y_w-z_w \rightarrow x_v-y_v-z_v \rightarrow \text{P.P.}$$
$$\vec{P}_i(x_i, y_i, z_i) \rightarrow \vec{P}'_i(x'_i, y'_i, z'_i) \rightarrow \vec{P}''_i(x'', y'', z'')$$

In Viewer

Drop D in Z
so, 2D with



Oct. 19 (W) "Depth".

Note:

1^o Midterm Scheduled on Nov. 7 (Monday).

Homework: Due One week from Today. Oct. 30. and a half 11:59 PM.

1^o Build (Implement C-Code¹⁰ the Transformation Pipeline in the target platform.

2^o. Build (Plot) $x_w-y_w-z_w$

Step 1.

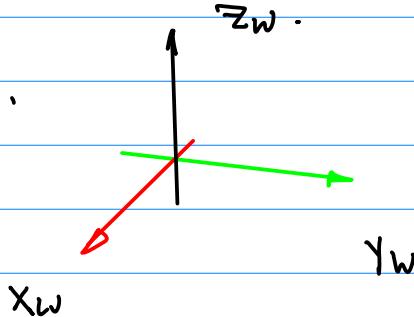
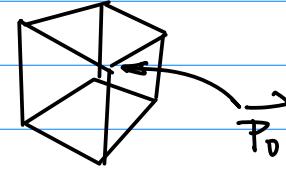


Fig 1.

Pick An Arbitrary point \vec{P}_0
 $\vec{T}_0(x_0, y_0, z_0) = (100, 100, 110)$

 \vec{P}_1 

$$\vec{P}_1(x_1, y_1, z_1) = (0, 0, 110), \text{ etc.}$$

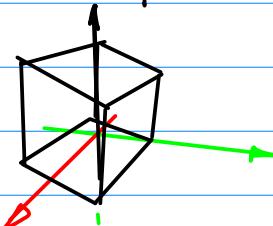
- ① Make $\Theta \in [20, 50]$;
- ② Length of Each axis = $50 \sim 100$
- ③ $E(x_e, y_e, z_e) = (200, 200, 200)$

Vectors for X_w, Y_w, Z_w .for example Starting pt $(0, 0, 0)$ Ending pt $(100, 0, 0)$ for X_w -axis

Note: Make sure the implementation of Virtual to physical display is ready. (Eqn (7) & (8) in the previous Lecture Note).

Step 2. Design A Data Set for a Cube.

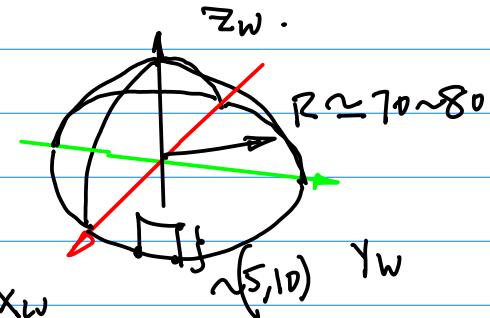
$$\left\{ \vec{P}_i(x_i, y_i, z_i) \mid i=0, 1, 2, \dots, N-1 \right\} \quad \cdots (1)$$

a. With Size as $100 \times 100 \times 100$ as illustrated Belowb. And elevated By 10 from X_w-Y_w plane

Displaying the Cube.

Step 3.

a. $R \approx 80$

b. Size of the upper part = 80.
(from R).Note: Do Each "Ring", with different R , then add Z value.

$$R_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \quad \left| \begin{array}{l} x_0 = y_0 = 0 \\ R_i \text{ for } i=1, 2, \dots, 8 \end{array} \right.$$

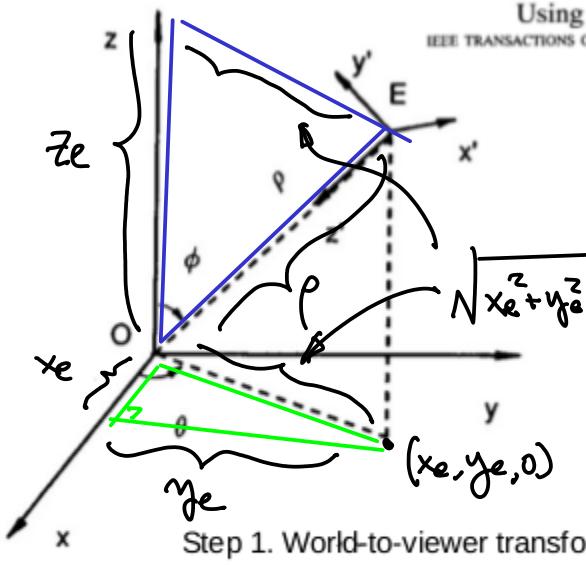
Computer

Oct. 19, 22

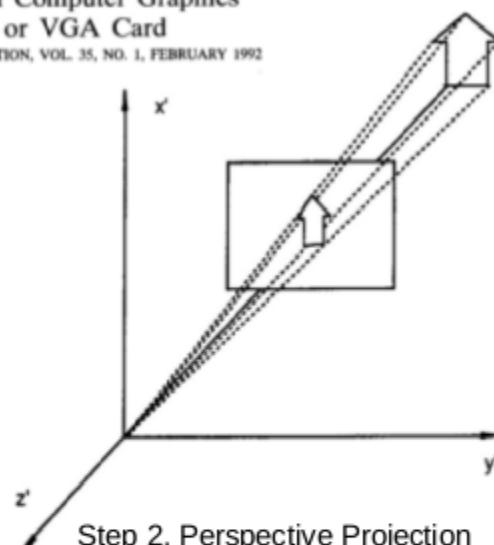
32

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

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

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

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

Harry Li, Ph.D.

Points in $X_w-Y_w-Z_w$: $\vec{P}_i(x_i, y_i, z_i)$

" in $X_v-Y_v-Z_v$: $\vec{P}'_i(x'_i, y'_i, z'_i)$

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

Points After Perspective Projection

$$\vec{P}''_i(x''_i, y''_i)$$

Oct. 24 (Mon).

Continued from the Transformation Pipeline.

Example:

After

$$\begin{pmatrix} x'_i \\ y'_i \\ z'_i \end{pmatrix} = \begin{bmatrix} -\sin \theta & \cos \theta & 0 & 0 \\ -\cos \phi \cos \theta & -\cos \phi \sin \theta & \sin \phi & 0 \\ -\sin \phi \cos \theta & -\sin \phi \cos \theta & -\cos \phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

Before

$$x'_i = -\sin \theta \cdot x_i + \cos \theta \cdot y_i \quad (x_{\text{prim}}[i] = -\sin(\theta)x[i] + \cos(\theta) * y[i];)$$

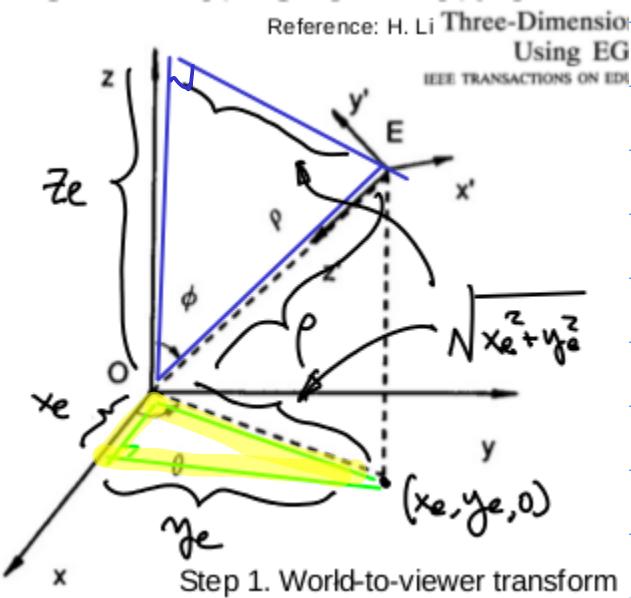
$$y'_i = -\sin \phi \cos \theta \cdot x_i - \cos \phi \sin \theta \cdot y_i + \sin \phi \cdot z_i \quad \dots (z)$$

$$z'_i = -\sin \phi \cos \theta x_i - \sin \phi \sin \theta y_i - \cos \phi z_i + \rho \dots (3)$$

C-Code can be derived accordingly. Similar to Eqn (1)

$$\text{Assume } E(x_e, y_e, z_e) = (200, 200, 200)$$

$$\text{Find } \cos \theta = \frac{x_e}{\sqrt{x_e^2 + y_e^2}} = \frac{200}{\sqrt{200^2 + 200^2}} = \frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{2}$$



$$\text{Find } \sin \theta = \frac{y_e}{\sqrt{x_e^2 + y_e^2}} = \frac{200}{\sqrt{200^2 + 200^2}} = \frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{2}$$

Now, for $\cos \phi$ (Blue Triangle)

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

$$= \frac{200}{\sqrt{200^2 + 200^2 + 200^2}} = \frac{1}{\sqrt{3}} = \frac{\sqrt{3}}{3}$$

$$\sin \phi = \frac{\sqrt{x_e^2 + y_e^2}}{\rho} =$$

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

$$= \frac{200\sqrt{2}}{200\sqrt{3}} = \frac{\sqrt{2} \cdot \sqrt{3}}{3}$$

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

a. $\begin{bmatrix} - \\ - \\ - \\ - \end{bmatrix}$

b.. $\begin{bmatrix} S & C \\ C & S \\ C & C \end{bmatrix}$

c. for ϕ $\begin{bmatrix} C & C & S \\ S & S & C & \rho \end{bmatrix}$

Example: Perspective Projection

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

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

Mapping Before (In x_v - y_v - z_v)
Virtual Camera Coordinate, and After
(In 2D Display Space),

Note: $\frac{x'_i}{x''_i} = \frac{y'_i}{y''_i} = \frac{z'_i}{z''_i}$

Due to the
Similar Triangle (see Fig.1)
In Green.

$\frac{D}{z_e}$:

IGA or VGA Card
EDUCATION, VOL. 35, NO. 1, FEBRUARY 1992

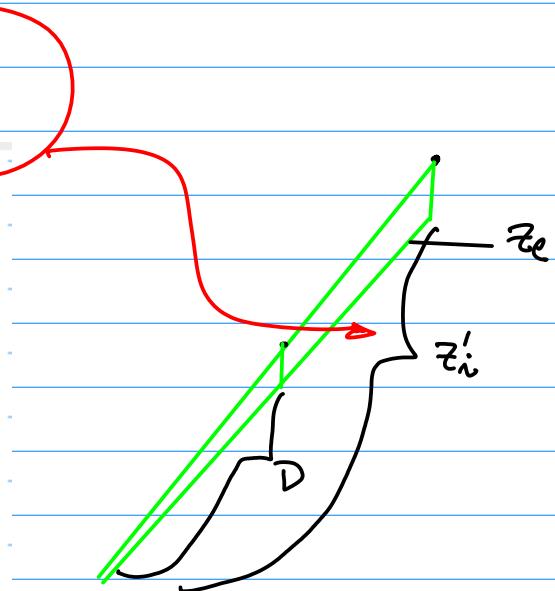
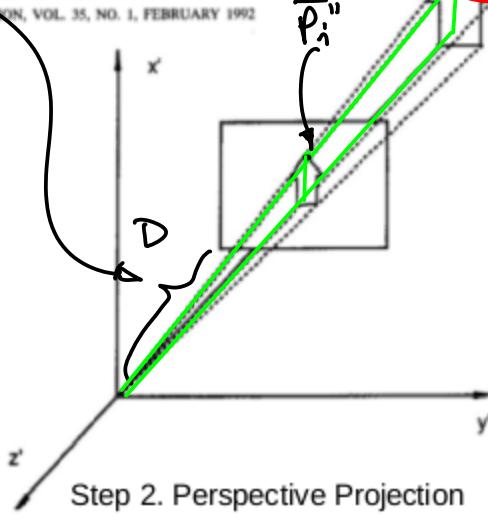
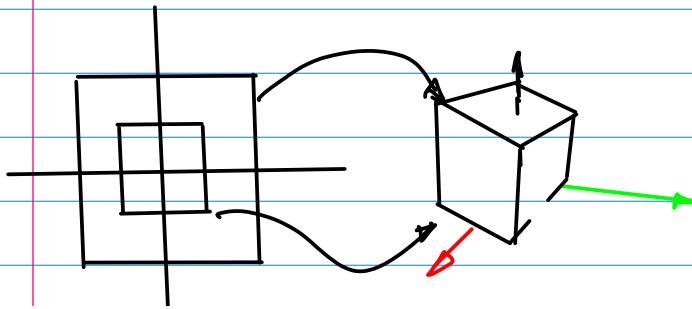


Fig.1.



2018F-118-13diffuseInterpolation20181127...

Ref: Sample Code.

$$\vec{E}(x_e, y_e, z_e)$$

```
34 float Xe = 200.0f, Ye = 200.0f, Ze = 250.0f; //virtual camera location
35 float Rho = sqrt(pow(Xe, 2) + pow(Ye, 2) + pow(Ze, 2));
36 float D_focal = 100.0f;
```

$$D$$

Define $X_w-Y_w-Z_w$

```
70     //define the x-y-z world coordinate
71     world.X[0] = 0.0;    world.Y[0] = 0.0;    world.Z[0] = 0.0;    // origin
72     world.X[1] = 50.0;   world.Y[1] = 0.0;    world.Z[1] = 0.0;    // x-axis
73     world.X[2] = 0.0;    world.Y[2] = 50.0;   world.Z[2] = 0.0;    // y-axis
74     world.X[3] = 0.0;    world.Y[3] = 0.0;    world.Z[3] = 50.0;   // z-axis
75

127     //sin and cosine computation for world-to-viewer
128     float sPhieta = Ye / sqrt(pow(Xe, 2) + pow(Ye, 2));
129     float cPhieta = Xe / sqrt(pow(Xe, 2) + pow(Ye, 2));
130     float sPhi = sqrt(pow(Xe, 2) + pow(Ye, 2)) / Rho;
131     float cPhi = Ze / Rho;
```

Oct.26(W).

Note: 1) Midterm Scheduled

on Nov.7; Need to Bring the Prototype System, and to Run/Program the code

2) Project 1D pts. 3D : Part I

$X_w-Y_w-Z_w$ World Coordinate

System; Part II — 3D Objects

A Cube & A Sphere.

Create A Dataset for the Sphere.

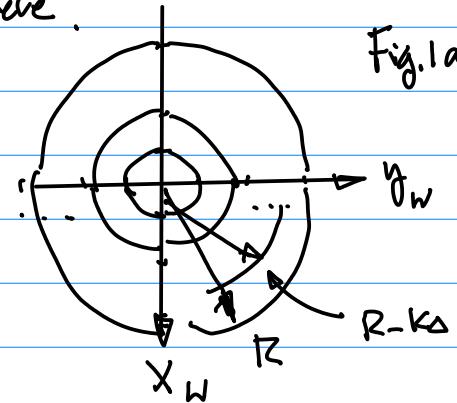
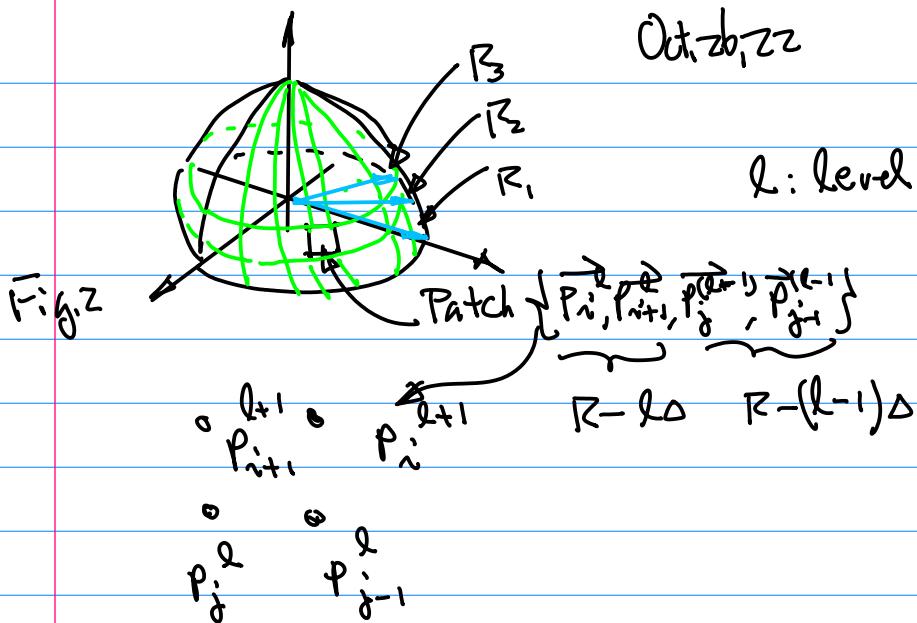


Fig.1a.

Fig.1.

Oct. 26, 22

36



l : Level of the Circle.
 $l = 1, 2, \dots, 8$

Note: Project! (Not Homework), 10 pt.

Due Nov. 2nd (W), Extended

2 Days.

Compute Transformation ($X_W \rightarrow W \rightarrow Z_W \rightarrow X_V - Y_V - Z_V$)

```

243     /*-----world to viewer-----*/
244     viewer.X[i] = -sPheta * world.X[i] + cPheta * world.Y[i];
245     viewer.Y[i] = -cPheta * cPhi * world.X[i]
246     - cPhi * sPheta * world.Y[i]
247     + sPhi * world.Z[i];
248     viewer.Z[i] = -sPhi * cPheta * world.X[i]
249     - sPhi * cPheta * world.Y[i]
250     -cPheta * world.Z[i] + Rho;
    
```



$$x'_i = -\sin\theta \cdot x_i + \cos\theta \cdot y_i \quad (x_{\text{prim}}[i] = -\sin(\theta)x[i] + \cos(\theta)y[i]) \quad \dots (1)$$

$$y'_i = -\sin\theta \cos\phi \cdot x_i - \cos\theta \sin\phi \cdot y_i + \sin\phi \cdot z_i \quad \dots (2)$$

$$z'_i = -\sin\theta \cos\phi \cdot x_i - \sin\theta \cos\phi \cdot y_i - \cos\theta \cdot z_i + \rho \quad \dots (3)$$

Perspective Projection

```

252 //-----perspective projection-----
253 perspective.X[i] = D_focal * viewer.X[i] / viewer.Z[i] ;
254 perspective.Y[i] = D_focal * viewer.Y[i] / viewer.Z[i] ;
255 if (perspective.X[i] > xMax) xMax = perspective.X[i];
256 if (perspective.X[i] < xMin) xMin = perspective.X[i];
257 if (perspective.Y[i] > yMax) yMax = perspective.Y[i];
258 if (perspective.Y[i] < yMin) yMin = perspective.Y[i];
259 /*

```

Optional: $x_w - y_w - z_w$ to $x_v - y_v - z_v$ Transform

Background: Two Basic Types of Operations.

↓ Translation
↓ Rotation(s)

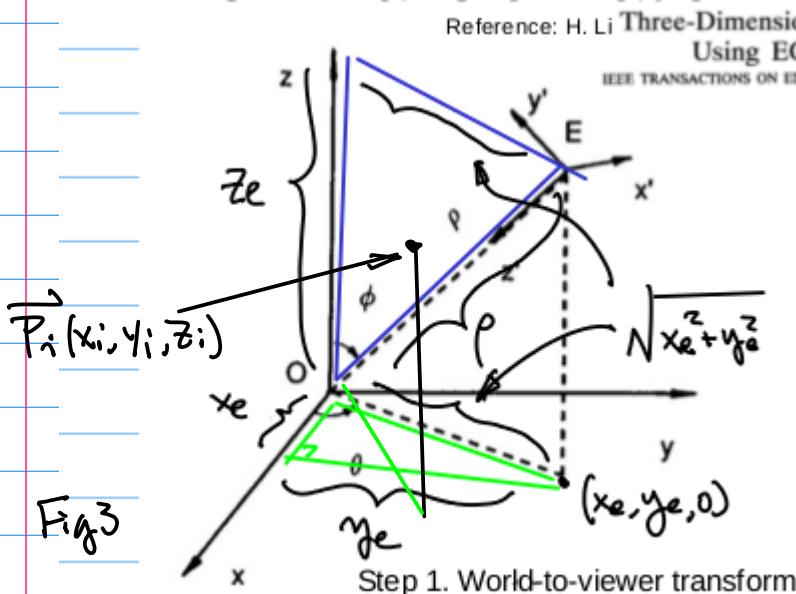


Fig 3

Translation:

$$T_{\text{translate}} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \dots (1)$$

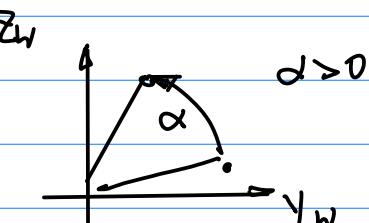
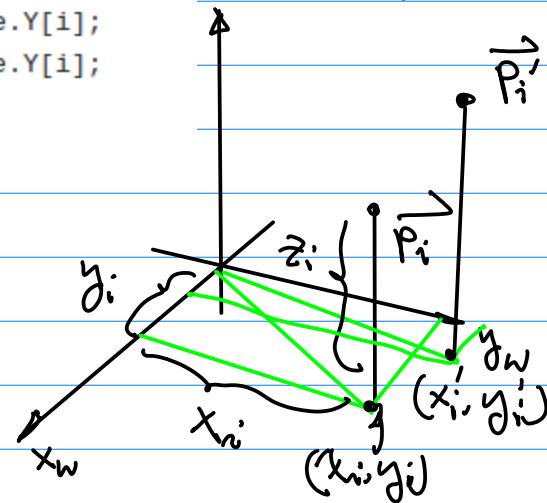


Fig 4



$$\begin{cases} x'_i = x_i + \Delta x \\ y'_i = y_i + \Delta y \\ z'_i = z_i + \Delta z \end{cases}$$

Rotation w.r.t x_w -axis

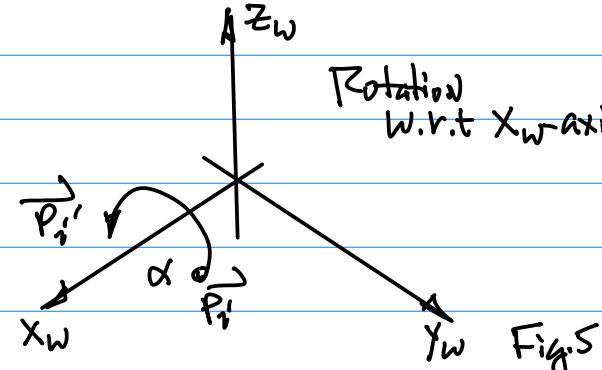
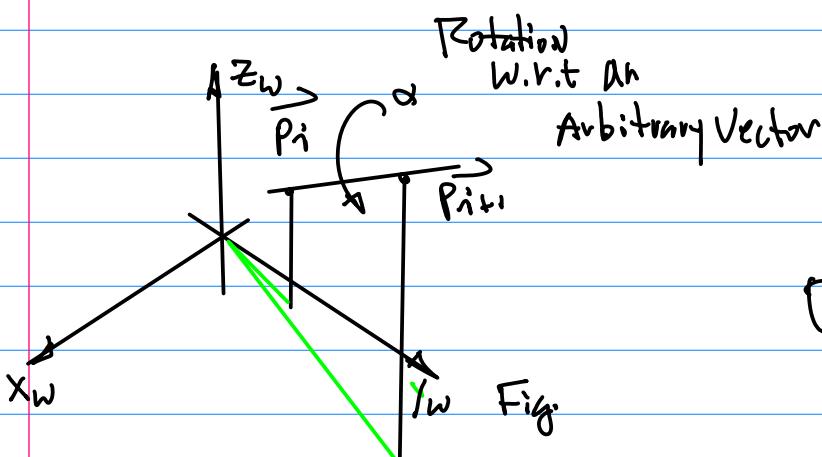
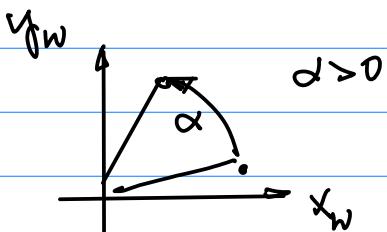
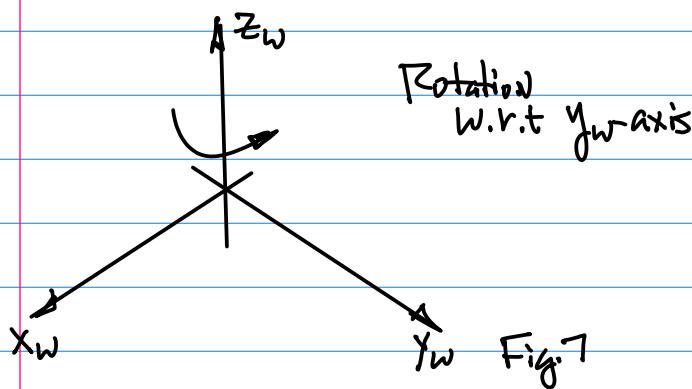
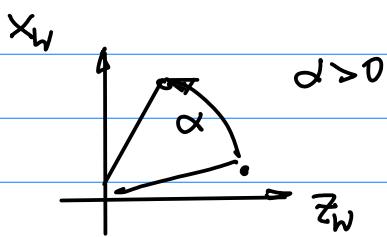
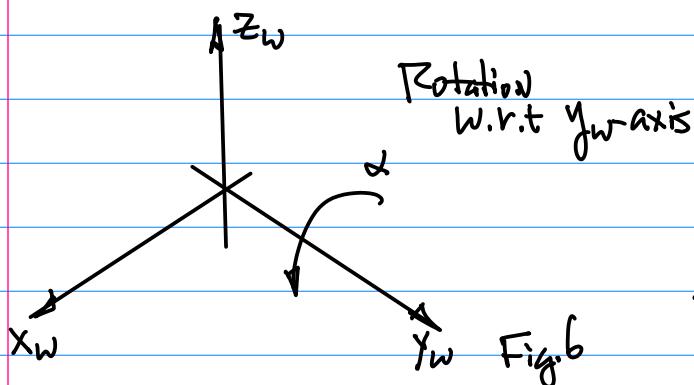


Fig 5



Oct. 31 (Monday) Today's Topics:

1. Coding Session.
2. 3D Transformations.

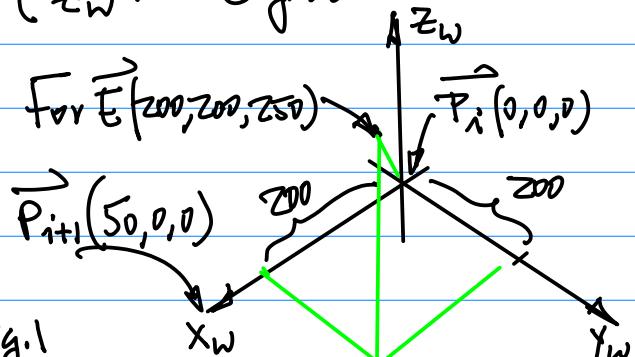
Example: Coding Session.

Step 1. Creation of DataSet

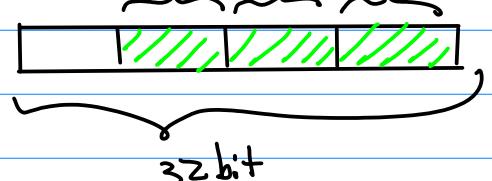
- a. for X_w, Y_w, Z_w Axis;
- b. $E(x_e, y_e, z_e)$,
- c. Sphere Creation

Defining Vertices to make Line Segments.

$\left\{ \begin{array}{l} X_w \text{ Line Segment} \\ Y_w \text{ Line Segment} \\ Z_w \text{ Line Segment} \end{array} \right.$



Note: Primitive Colors R, g, b are implemented as 32bit Data 8 bit Color



OpenCV: BGR; Deep Learning: RGB
Computer Vision;

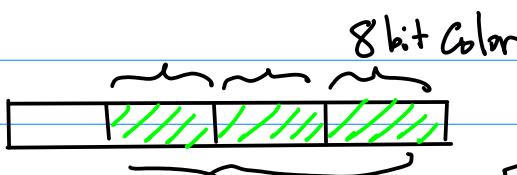


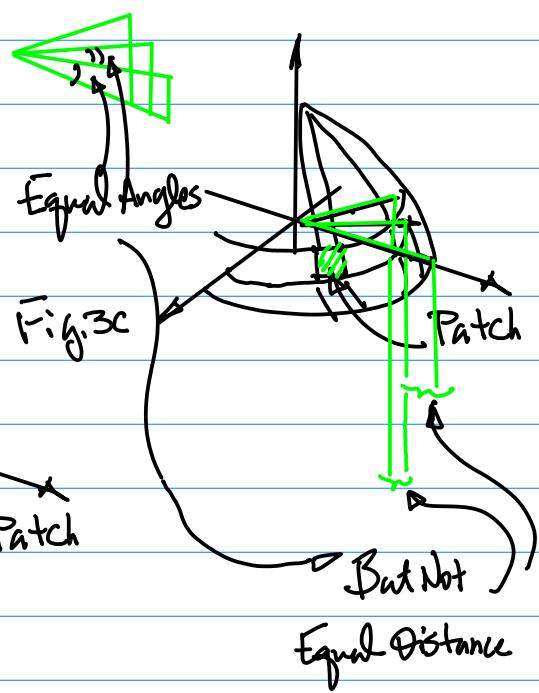
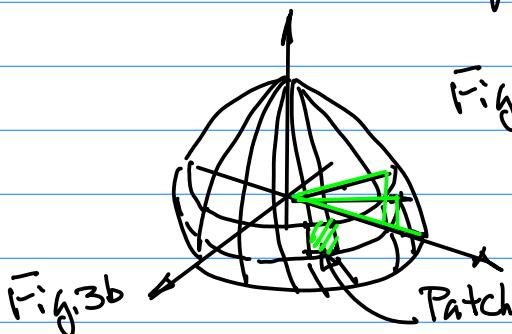
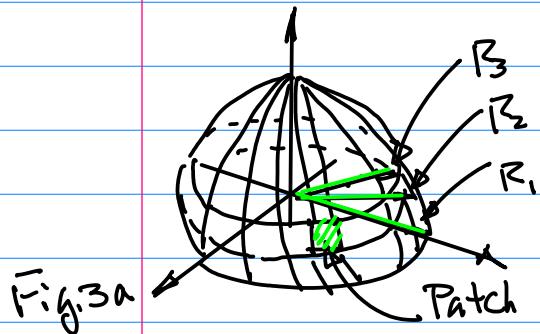
Fig.2

Note: ARM CPU is 32 or 64 Bits. 24 bits Occupies 4 Bytes

Older Machine 15 bits or 16 bits Implementation.

Step 2. $x_w - y_w - z_w \rightarrow x_v - y_v - z_v$
 Perspective Projection
 θ, ϕ
 $\cos\theta, \sin\theta ; j$
 $\cos\phi, \sin\phi ; \rho$

Step 3. Virtual Display Coordinate System to the physical Display System (Reflects Your Actual LCD)



Computer

Oct. 31, 22

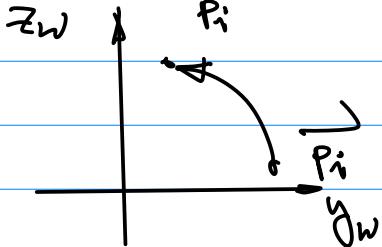
40

Mathematical Formulation for
Rotation of $P_i(x_i, y_i, z_i)$ w.r.t.
 x_w -axis. Fig. 5, pp. 37.

Before \vec{P}_i , After \vec{P}'_i

$$x_i = x'_i$$

↓ Rotation on y_w - z_w
plane,



Hence,

$$R_{xw} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{pmatrix} x'_i \\ y'_i \\ z'_i \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \dots (1)$$

Note: In 2D Rotation

$$\text{Inpt. } \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \xrightarrow{\text{Indp.}} \text{Func.}$$

Nuv. 2nd (Wrd).

Review for the Midterm Exam On Nov. 7th
(Monday)

1. One Hour Exam. 3 Question.

(Question): Architectural Aspects of the

ARMGPU,

Memory maps

Special Purpose Registers, CPU Datasheet

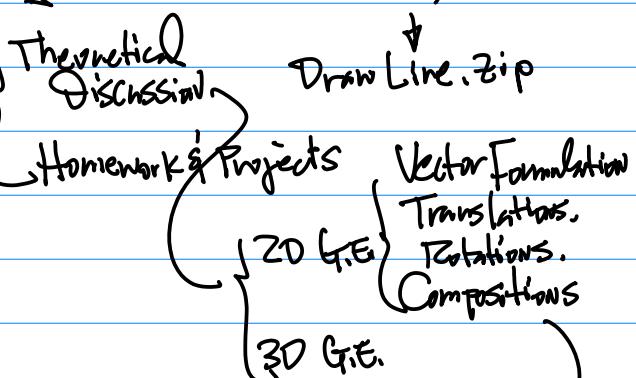
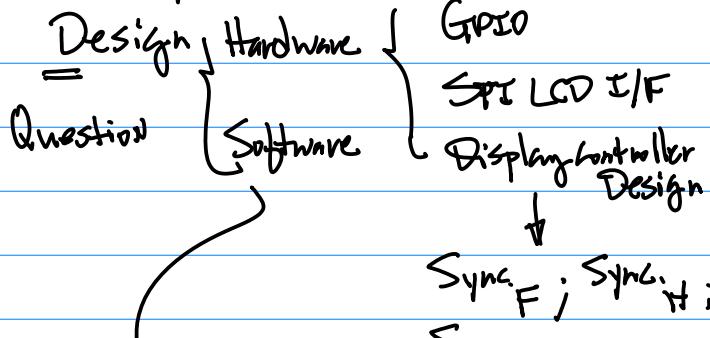
Design Specs, Define Init & Config Pattern;

GPP, SPI

Coding

Note: 1. Need CPU Datasheet.

2. Bring your Prototype System, together
Your Laptop Computer.



Virtual Display v.s.
Physical Display.

- 1. Transformation Pipeline
World-Z-Viewer;
Perspective Projection
- 2.

1. Transformation Pipeline
World-Z-Viewer;
Perspective Projection
- Project: Implementation.
Requires the execution
of your code.

After projecting the Rotation onto
Xw-Yw plane

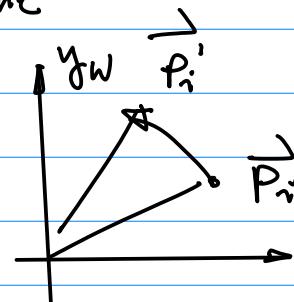


Fig 1b.

3. Exam Paper will be posted
on CANVAS.

1 hr + 15 min

4. Paper Answer Sheets.

a) First-LastName-SID(4 Digits)-CmpE240

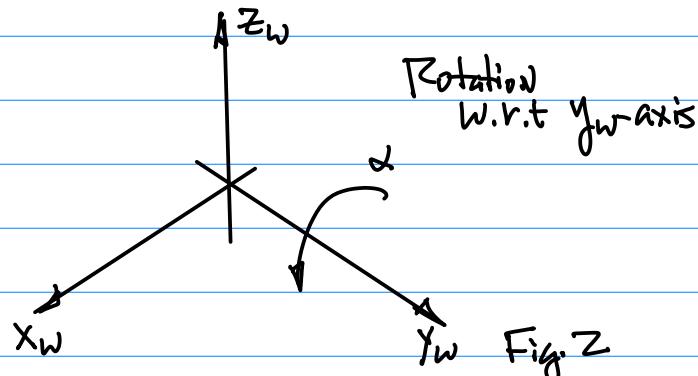
b) Photo for Each Page (jpg, png)

c) On-Line tool to convert them to
pdf.

d) Arrange the pdfs in a Sequential
order, merge them into one pdf

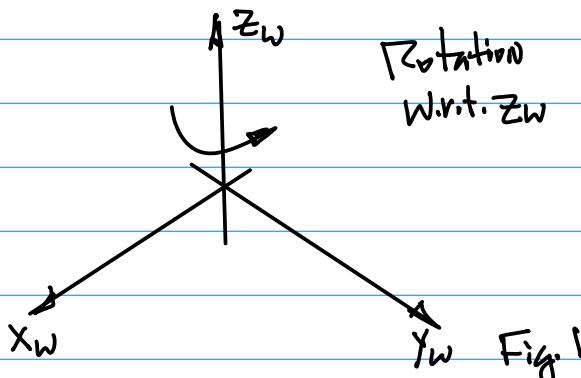
e) pdf for Both Answer Sheets
One & photos

$$R_{zw} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \dots (1)$$



Note: Projection onto Zw-Xw plane.
Zw — Irrep, Xw — Func.
Yi is constant (Before & After
it stays the same).

Example: Continuation of 3D Rotations.

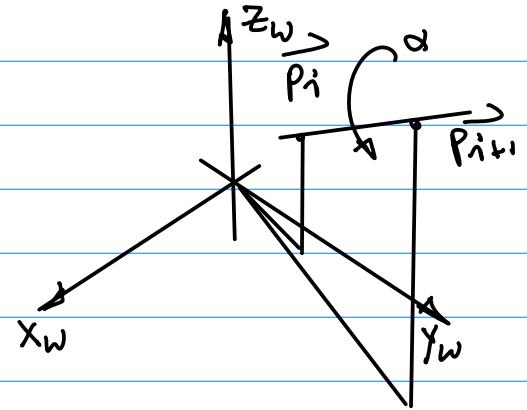
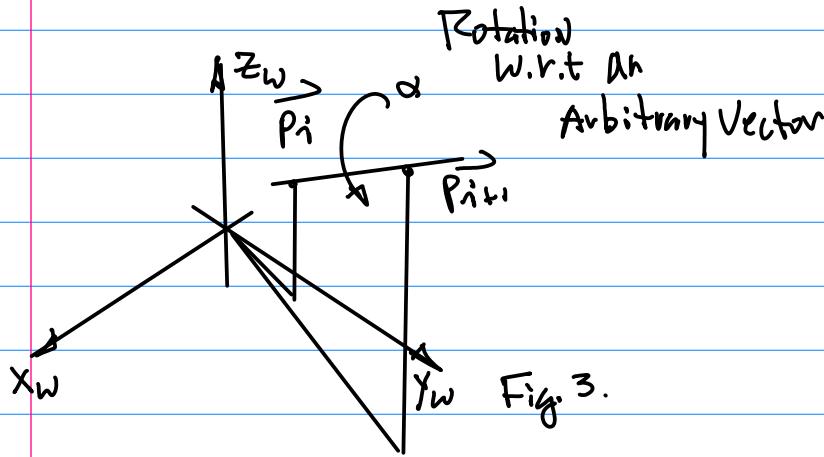


$$R_{yw} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \dots (2)$$

Nov. 2nd, 22

$$\vec{P}_i = (x_i, y_i, z_i), \vec{P}_{i+1} = (x_{i+1}, y_{i+1}, z_{i+1}) \dots (3)$$

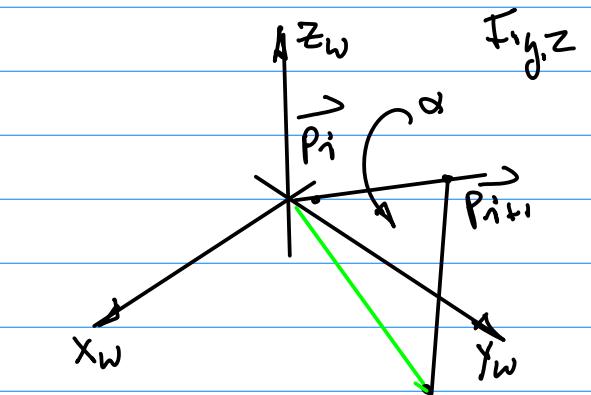
$$\|\vec{P}_{i+1} - \vec{P}_i\| = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \dots (4)$$



Nov. 9 (Wed).

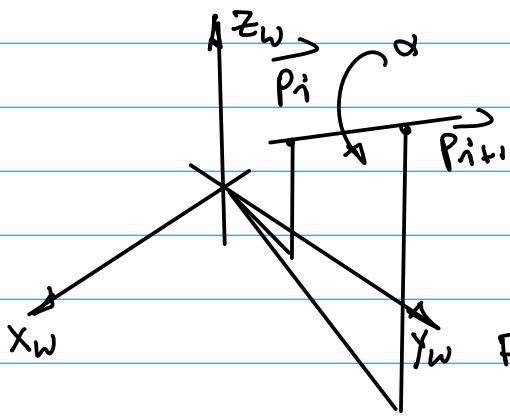
Example: Continuation of the
Rotation w.r.t An Arbitrarily
Axis

7 Steps { The Pre-processing:
3 steps
Rotation wrt Z-axis
Post-Processing: 3 steps
to undo the
Pre-processing.



$$T = \begin{bmatrix} 1 & 0 & 0 & \Delta X \\ 0 & 1 & 0 & \Delta Y \\ 0 & 0 & 1 & \Delta Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \dots (1)$$

Where $\Delta X = -x_i$ from $\vec{P}_i(x_i, y_i, z_i)$
 $\Delta Y = -y_i$, $\Delta Z = -z_i$



Step 1.
Translation to move
the Arbitrary Vector
to the origin,
e.g. $\vec{T}_i(0, 0, 0)$

Step 2. Rotation Wrt ZW-axis,

till the vector is on
ZW-XW.

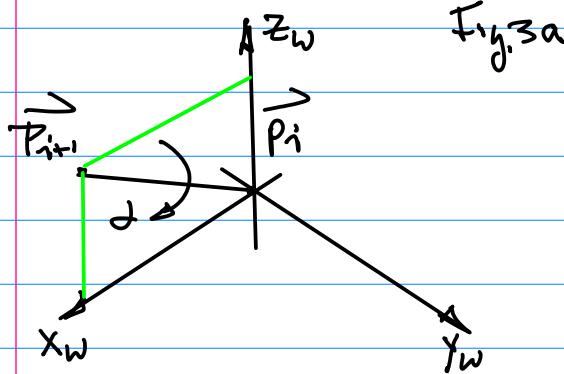
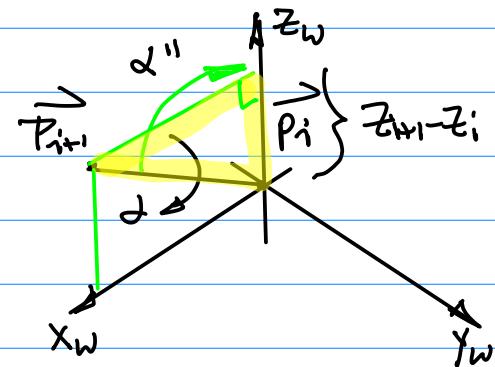


Fig. 3a

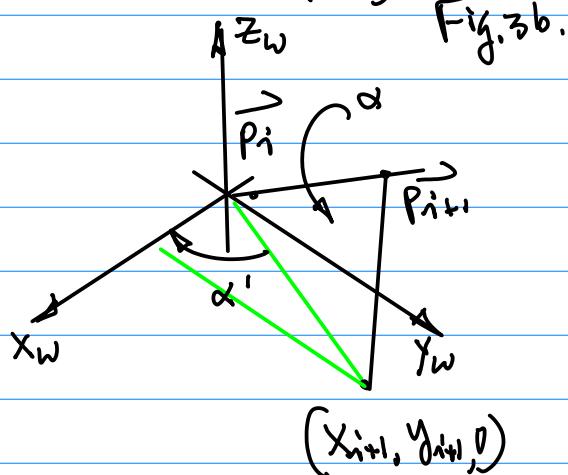


$$R_{Zw} = \begin{pmatrix} \cos\alpha' & -\sin\alpha' & 0 & 0 \\ \sin\alpha' & \cos\alpha' & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \dots (z)$$

$$R_{Yw} = \begin{pmatrix} \cos\alpha'' & 0 & \sin\alpha'' & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha'' & 0 & \cos\alpha'' & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \dots (3)$$

$$\cos\alpha' = \frac{x_{i+1}}{\sqrt{x_{i+1}^2 + y_{i+1}^2}}, \text{ Find } \sin\alpha' \\ \dots (z-b)$$

$$\vec{P}_{i+1}(x_{i+1}, y_{i+1}, z_{i+1}), \vec{P}_i(x_i, y_i, z_i)$$



$(x_{i+1}, y_{i+1}, 0)$

Step 3. Rotates the vector on
Zw-Xw plane wrt Yw-axis
(Note \vec{P}_i is fixed on the
origin).

$$\cos\alpha'' = \frac{z_{i+1} - z_i}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \dots (3-b)$$

Verify this!

) see pp. 42

$$\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}$$

Hence,

$$\cos\alpha'' = \frac{z_{i+1} - z_i}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}} \dots (3-c)$$

Note:

For Release, we have

$$T_{\Sigma_{Pre}} = R_{Yw} R_{Zw} T \quad (4 \times 4 \text{ Matrix})$$

For Debugging, use Separate Step-by-Step Matrix.

Step 4

Now, the Actual Rotation of

$$R_2 = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \dots (4)$$

$$T_7 = T_1^{-1}$$

$$T_7 = T_1^{-1} = \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix} \dots (7)$$

Post Processing:

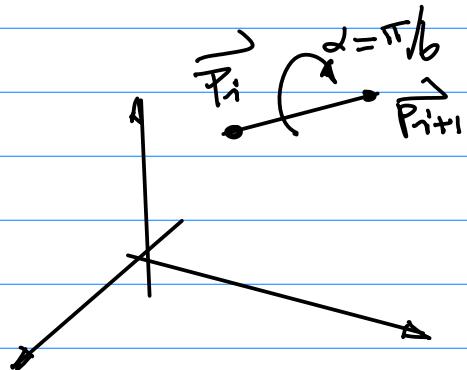
Step 5. (Undo Step 3)

$$R_5 = R_3^{-1}$$

Note: No Need to Find the inverse matrix, just change the sign of the rotation matrix.

$$R_5 = R_{yw}^{-1} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \dots (5).$$

$$T_{\Sigma, \text{Post}} = T_7 R_6 R_5 \dots (8)$$



$$\vec{P}_i(100, 100, 300), \vec{P}_{i+1}(200, 200, 310)$$

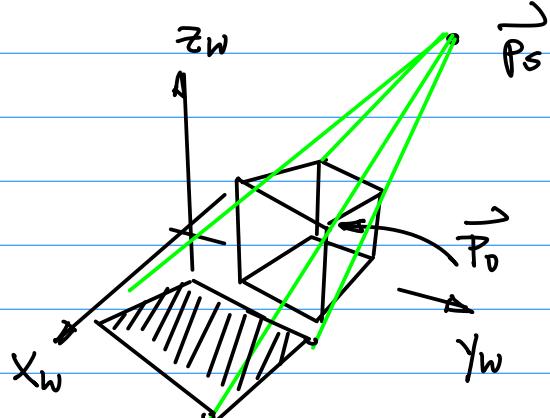
Step 6. Similarly. (undo Step 2)

$$R_6 = R_z^{-1}, \text{ e.g.}$$

$$R_6 = R_{zw}^{-1}$$

$$R_{zw}^{-1} = \begin{pmatrix} \cos\theta & 0 & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \dots (b)$$

Step 7. Undo Step 1.



Nov. 14 (Monday)

Note: For the 3D project. Due Nov. 30th.

Features:

1° Rotation of the Cube by Small

amount, such as $5\pi/10$ Degree
w.r.t An Arbitrary Vector \vec{o} of
your choice with the consideration
of Shadow Computation, and
Diffuse Reflection;

2° Decoration of the Cube Surface

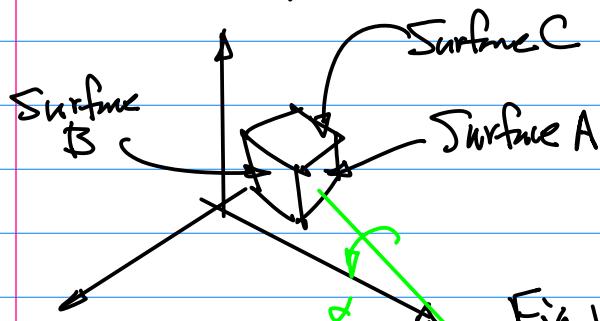


Fig.1.

With 2D Existing Projects
Rotating Squares, and Trees

Note: Keep the Sphere.

3° Shadow

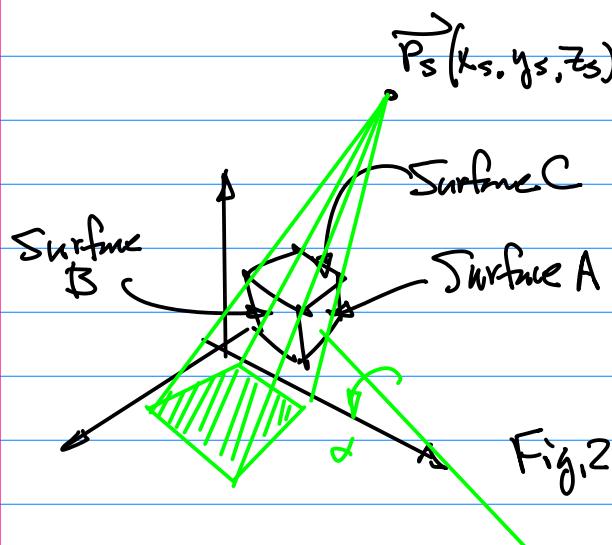


Fig.2

4. Diffuse Reflection.

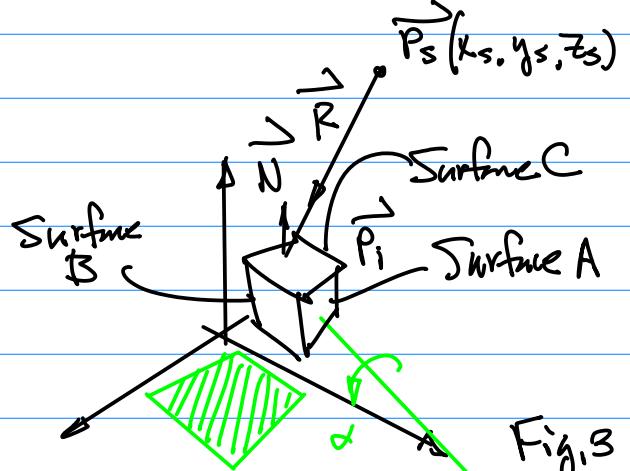


Fig.3

Step1. Compute Diffuse Reflection on
the 4 Vertices in $X_w/Y_w/Z_w$.

After P.P.

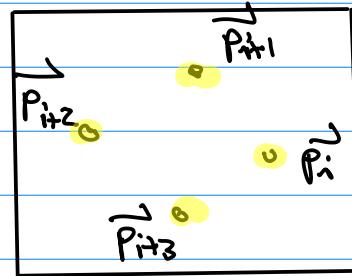


Fig.4

Step2. Compute the Color on the Boundary

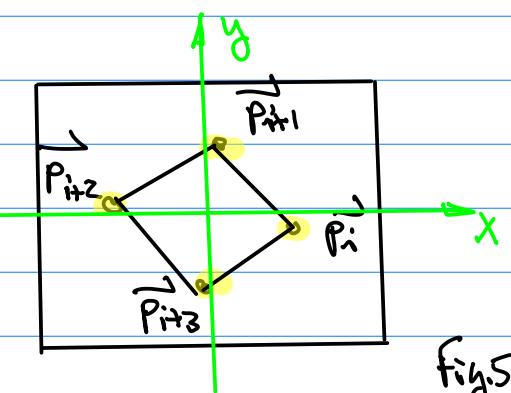
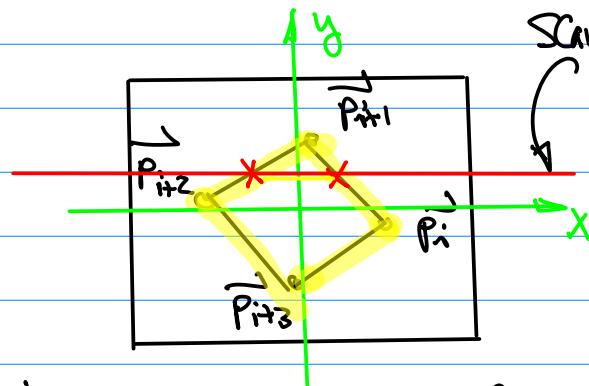


Fig.5

Step 3. Compute the Color Fig.b



Note: Bonus Option 1. Digital photo of your choice on $X_w - Y_w$ plane. 3% of Total Score of the Entire Semester.

Bonus Option 2: Diffuse Reflection on the Sphere.

Options are Not Mandatory.

Example: Decoration Algorithm.

Given 2D Patterns. $\{\vec{P}_i | i=1, 2, \dots, k\}$

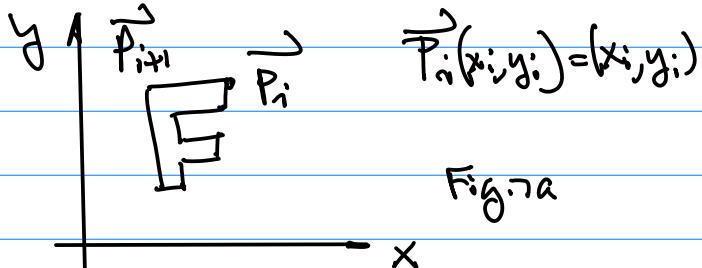


Fig.7a

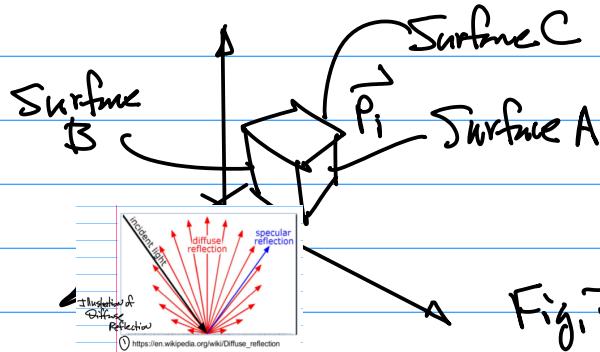


Fig.7b

First, to move 2D Pattern $\{\vec{P}_i(x_i; y_i) | i=1, \dots, k\}$ to $X_w - Y_w - Z_w$, by Adding $Z_i = 0$.

$$\vec{P}'_i(x'_i; y'_i; 0) = (x'_i, y'_i, 0)$$

Then, Draw it in $X_w - Y_w - Z_w$

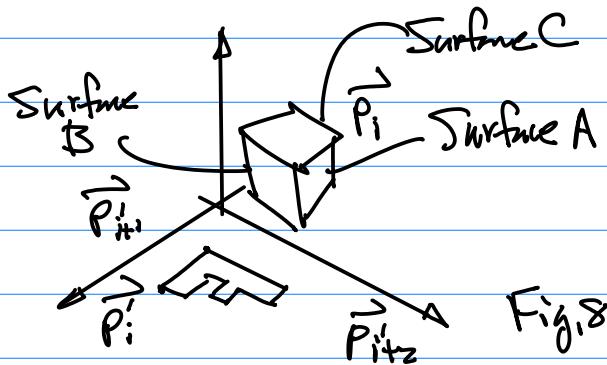


Fig.8

Step 2. Formulation to Decorate Surface C.

$$\left. \begin{array}{l} \text{After} \\ x'_i = x_i \\ y'_i = y_i \\ z'_i = C + H \end{array} \right\} \dots (1)$$

Side of the Color floatation

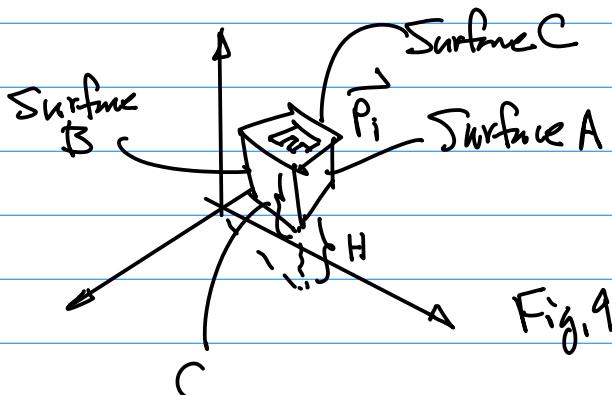


Fig.9

Example:

Decorate Surface B

After

$$\left. \begin{array}{l} y'_i = x_i \text{ Ind.} \\ z'_i = y_i \text{ Func} \\ x'_i = C \text{ (Side of the Cube)} \\ \dots (z) \end{array} \right\}$$

Before

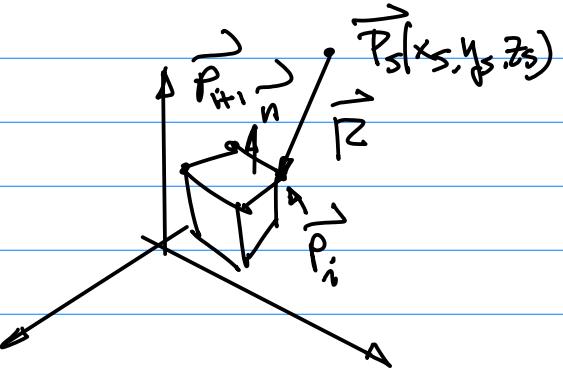
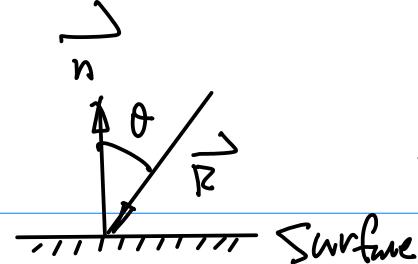


Fig.1.

For Surface A:

$$\left. \begin{array}{l} z'_i = x_i \text{ Ind.} \\ x'_i = y_i \text{ Func} \\ y'_i = C \\ \dots (3) \end{array} \right\}$$

Nov.21 (Monday) Diffuse Reflection

Ref:

2021F-101b-notes-cmpe240-2021-12-1.pdf

Example: First, Consider the Diffuse Reflection at 4 corners of a given Cube.

Given Conditions:

1° $\vec{P}_s(x_s, y_s, z_s)$ 2° A set of Points $\{\vec{P}_i | i=1, 2, \dots, k\}$ in $X_w-Y_w-Z_w$.3° Normal Vector $n(x_n, y_n, z_n)$ is defined

Note:

1° Definition: PP.st. Reflection
Uniformly in All different directions.

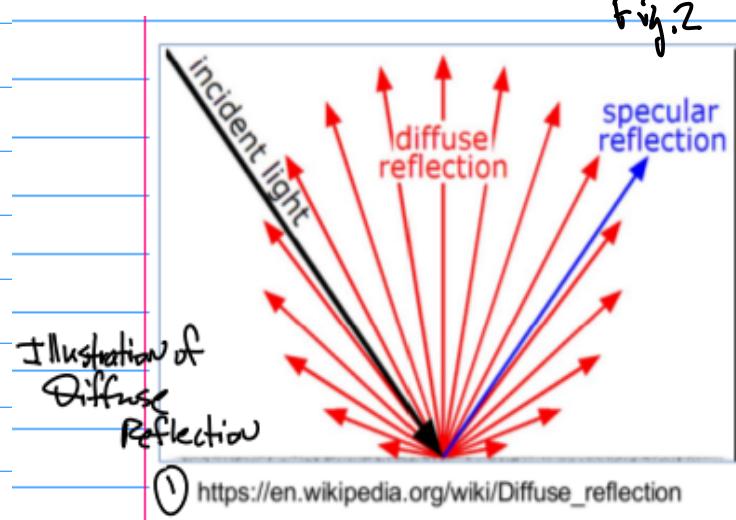


Fig.2

(1) https://en.wikipedia.org/wiki/Diffuse_reflection

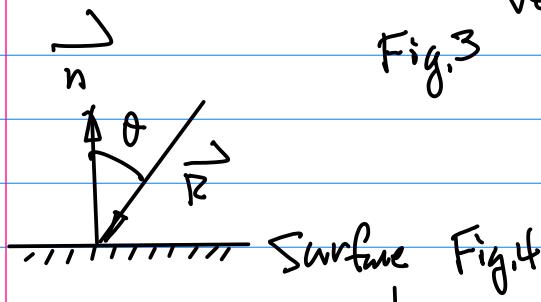
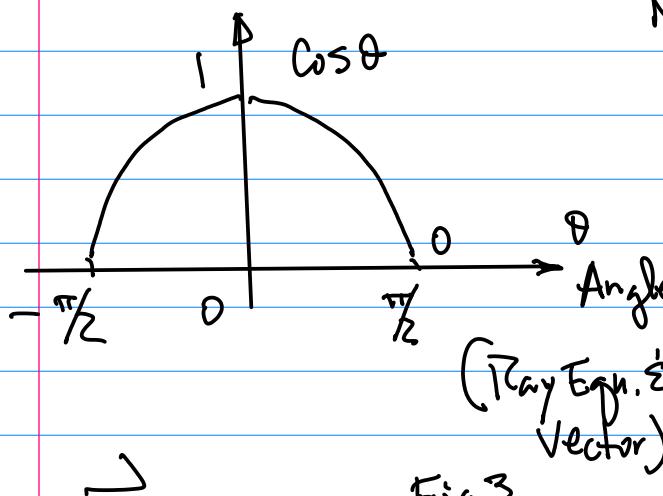
2° Intensity of the Diffuse Reflection, The Intensity of $I(x, y) = (r(x, y), g(x, y), b(x, y))$

r red g green b blue

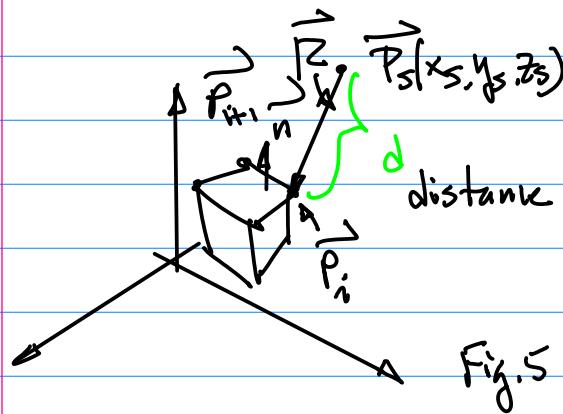
depends on the incoming angle of the Ray Equation.

Nov. 21, 22

48



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



Computation, Formulation.

To find Squared distance:

From the Ray Equation.

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

PP 57.

$$\text{then, } \|\vec{r}\| = \sqrt{[x_i + \lambda(x_s - x_i)]^2 + [y_i + \lambda(y_s - y_i)]^2 + [z_i + \lambda(z_s - z_i)]^2} \quad \dots (2)$$

$$\text{Hence, } \frac{1}{\|\vec{r}\|^2} \quad \dots (3)$$

Now, Angle θ , then $\cos \theta$

$$\vec{r} \cdot \vec{n} = \|\vec{r}\| \|\vec{n}\| \cos \theta$$

$$\cos \theta = \frac{\vec{r} \cdot \vec{n}}{\|\vec{r}\| \|\vec{n}\|} \quad \dots (4)$$

$$= \frac{(x_r, y_r, z_r) \cdot (n_x, n_y, n_z)}{\sqrt{x_r^2 + y_r^2 + z_r^2} \cdot \sqrt{n_x^2 + n_y^2 + n_z^2}}$$

$$= \frac{x_r n_x + y_r n_y + z_r n_z}{\sqrt{x_r^2 + y_r^2 + z_r^2} \cdot \sqrt{n_x^2 + n_y^2 + n_z^2}}$$

... (4b)

Together

$$\text{Intensity} \sim \frac{1}{\|\vec{r}\|^2} \cos \theta$$

$$= \frac{1}{\|\vec{r}\|^2} \frac{\vec{r} \cdot \vec{n}}{\|\vec{r}\| \|\vec{n}\|} \quad \dots (5)$$

$$\text{Reflectivity } \vec{K}_d = (K_{dr}, K_{dg}, K_{db})$$

↑ ↑ ↑
Red Green Blue

$$\vec{I}_{dr}^{(x, y)} = K_{dr} \frac{1}{\|\vec{r}\|^2} \frac{\vec{r} \cdot \vec{n}}{\|\vec{r}\| \|\vec{n}\|} \quad \dots (6)$$

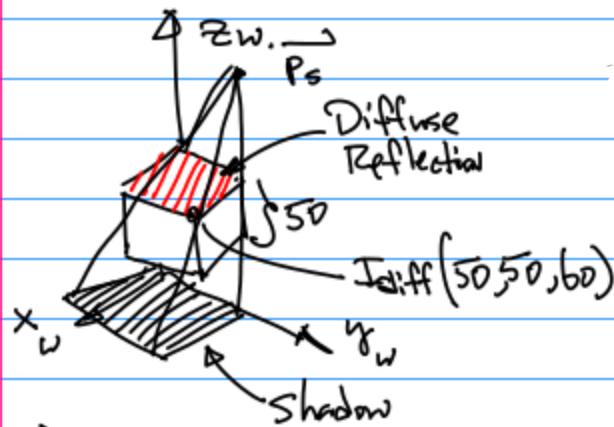
Nov.21, 22

49

Note: $K_{dr} \in [0, 1]$

No Reflection Full Reflection

Example: PP57.



$\vec{P}_s(40, 50, 120)$. Cube is floating by $z=10$. Hence:

Find $\vec{I}_{diff}(50, 50, 60) = ?$

Sol: From Eqn (b), we have

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

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

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

Since $K_{dr} = 0.8$, $K_{dg} = 0$, $K_{db} = 0$

First, find Squared distance. PP.57

$$\begin{aligned} \|\vec{r}\|^2 &= (x_s - x_i)^2 + (y_s - y_i)^2 + (z_s - z_i)^2 \\ &= (40-50)^2 + (50-50)^2 + (120-60)^2 \\ &= 10^2 + 10^2 + 60^2 \end{aligned}$$

$$\text{then, find } \cos \theta = \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|}$$

$$\text{where } \vec{n} = (0, 0, 1)$$

directional vector for \vec{r} :

$$(\vec{P}_s - \vec{P}_i)$$

$$= (x_s - x_i, y_s - y_i, z_s - z_i)$$

$$\begin{aligned} \vec{n} \cdot \vec{r} &= (0, 0, 1) \cdot (40, 50, 120) = 0 \times 40 + 0 \times 50 \\ &\quad + 1 \times 120 = 120 \end{aligned}$$

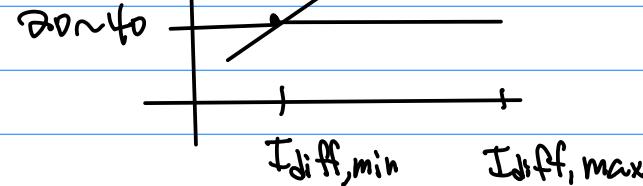
$$\frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} = \frac{120}{1 \cdot \sqrt{10^2 + 10^2 + 60^2}} = \frac{120}{\sqrt{40^2 + 60^2 + 120^2}}$$

Together

$$\vec{I}_{diff,r}(x, y, z) = 0.8 \cdot \frac{1}{10^2 + 10^2 + 60^2} \cdot \frac{1}{\sqrt{40^2 + 60^2 + 120^2}}$$

$$\vec{P}_i(50, 50, 60) \approx 255$$

I_{diff}



Nov. 28.

Note: Ref:

CMPE240-Adv-Microprocessors / 2018F / 2021F-101b-notes-cmpe240-2021-12-1.pdf

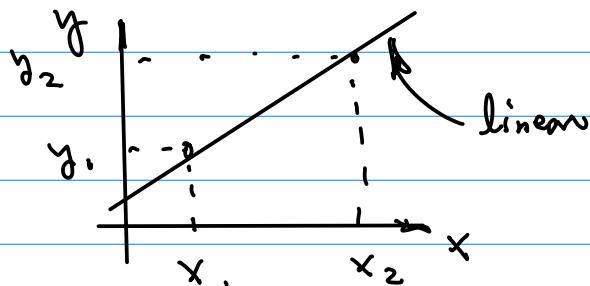
PP.58

Example: Computation of Diffuse Reflection Along the Boundary Lines (Points).

PP59.

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

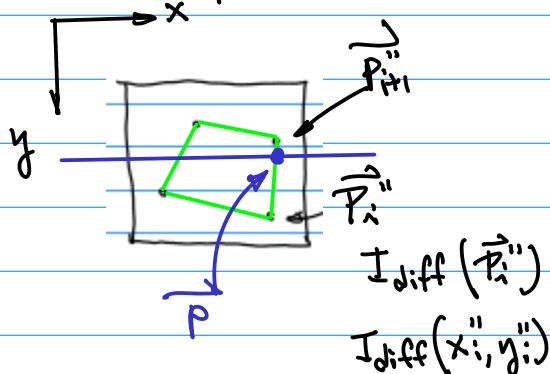
... (1b)



Eqn (1b) in the following form:

$$y = ax + b \quad \dots (1c)$$

To Compute diffuse reflection on Boundary Lines.



1° Locations $\vec{P}_i''(x_i'', y_i'')$ are

Known (After Perspective Projection)

2° $I_{\text{diff}}(\vec{P}_i'')$ are known, $i=1, 2, 3, 4$.

From Eqn (1b) & (1c):

$$y = ax + b$$

Intensity

Indep. Variables $\{x_i''\}$
 $\{y_i''\}$

$$I_{\text{diff}, x}(\vec{P}) = ax + b$$

Now, for y

$$I_{\text{diff}, y}(\vec{P}) = ay + b$$

Then, Combine them to form the solution

By Taking the Average

$$I_{\text{diff}}(\vec{P}) = \frac{1}{2} (I_{\text{diff}, x}(\vec{P}) + I_{\text{diff}, y}(\vec{P})) \quad \dots (2)$$

where

$$a = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \frac{I_{\text{diff}}(\vec{P}_i) - I_{\text{diff}}(\vec{P}_{i+1})}{x_i - x_{i+1}} \quad \dots (3)$$

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

$$= - \frac{I_{\text{diff}}(\vec{P}_i) - I_{\text{diff}}(\vec{P}_{i+1})}{x_i - x_{i+1}} x_{i+1} + I_{\text{diff}}(\vec{P}_i) \quad \dots (4)$$

Now, for y

$$a' = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \frac{I_{\text{diff}}(\vec{P}_i) - I_{\text{diff}}(\vec{P}_{i+1})}{y_i - y_{i+1}} \quad \dots (5)$$

$$b' = -\frac{y_1 - y_2}{x_1 - x_2} x_2 + y_2$$

$$= -\frac{I_{\text{diff}}(\vec{P}_i) - I_{\text{diff}}(\vec{P}_{i+1})}{y_i - y_{i+1}} y_{i+1} + I_{\text{diff}}(\vec{P}_{i+1})$$

... (6)

Nov.30 (Wed)

Note: DDA (Digital Differential Algorithm). Motivation:

$$(1) \quad x_{k+1} = x_k + 1 \quad \dots (1a)$$

$$y_{k+1} = ax_{k+1} + b \quad \text{Need "mult".} \quad \dots (1b)$$

Hence, $I_{\text{diff},x}(\vec{P}) = ax + b$ (from $\vec{P}(x, y)$)
 Blue pt. ... (7)

$I_{\text{diff},y}(\vec{P}) = a'y + b'$ (from $\vec{P}(x, y)$,
 Blue pt.) ... (8)

Reduce it to Addition.

② "GAPS" problem.

2021F-101b-notes-cmpe240-2021-12-1.pdf

Finally, Combine Them, From Eqn(7) & (8)

$$I_{\text{diff}}(\vec{P}(x, y)) = \frac{1}{2} (I_{\text{diff},x}(\vec{P}) + I_{\text{diff},y}(\vec{P}))$$

Reference for the Numerical Example is on page 61

Consider Finite Resolution of A Display Device
 In Hardware, we Need to find Exact pt on the
 "Green Line".

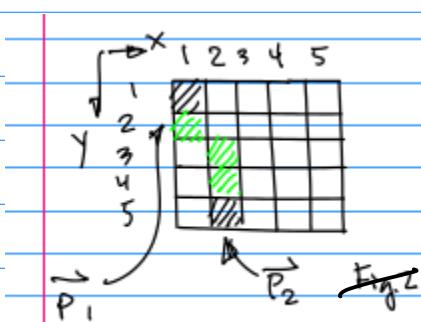
Example: PP5a. $\vec{P}_1(1, 1), \vec{P}_2(2, 5)$ 

Fig.1

Example: On Page 63.

$$\frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \frac{5-1}{2-1} = 4$$

from Eqn(1a)

$$x_{k+1} = x_k + 1 \quad | \quad x_k = 1 \quad = 1 + 1 = 2$$

Hence, Eqn(1b)

$$y_{k+1} = ax_{k+1} + b$$

$$= -4 \cdot 2 + b = 8 + 13 = 5$$

where

$$b = -\frac{y_1 - y_2}{x_1 - x_2} x_2 + y_2, \text{ e.g.}$$

$$= -\frac{1-5}{1-2} \cdot 2 + 5 = 4 \cdot 2 + 5$$

$$= 13$$

Start from the Formulation

$$y = ax + b, \quad a = \frac{y_1 - y_2}{x_1 - x_2} = \frac{1-5}{1-2} = 4 \rightarrow \text{"GAP!"}$$

$$a = \frac{y_1 - y_2}{x_1 - x_2} = \frac{1-5}{1-2} = 4$$

Nov. 30, 22

52

Design Guideline:

Condition 1. Check $|a| \leq 1$, if it is satisfied, then

$$\begin{cases} X_{k+1} = X_k + 1 & \dots |z a) \\ y_{k+1} = ax_{k+1} + b & \dots |z b) \end{cases}$$

from $y_{k+1} = ax_{k+1} + b$

$$\begin{aligned} &= a(x_k + 1) + b \\ &= ax_k + a + b \\ &= \underbrace{ax_k + b}_{= y_k} + a \\ &= \underline{\underline{y_k + a}} \end{aligned}$$

Note: at the end of the computation, when ready to plot this point, use "Rounding" (to the integer value).

Condition 2: When $|a| > 1 \rightarrow$ "GAP"

OR "GRABS"

to fix this problem, make y as an independent variable,
but x as a function.

$$y = ax + b$$

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

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

$$x = a'y + b' \dots |3)$$

where $a' = 1/a$, $b' = -b/a$

Example: Fig. 1. Pg 51.

$$\therefore a = \frac{y_1 - y_2}{x_1 - x_2} = \frac{1}{4}, |a| > 1$$

therefore, use Eqn (4)

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

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

from Eqn (4),

$$x_2 = x_1 + a' \quad \left| a' = \frac{1}{4} \right. = 1 + \frac{1}{4} = \frac{5}{4} \approx 1 \frac{1}{4}$$

To plot, Rounding, then $(x_2, y_2) = (1, 2)$

No GAP.

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

$$x_3 = x_2 + \frac{1}{a} = \frac{5}{4} + \frac{1}{4} = \frac{6}{4} \approx 1 \frac{1}{2}$$

$$(x_3, y_3) = (2, 3)$$

$$\begin{cases} y_{k+1} = y_k + 1 \\ x_{k+1} = x_k + a' \end{cases} \dots |4)$$