Sept. 7.

Note: 1° LPC1769 from 2022S Semester, Waiting List. CANVAS. Scott.

2° LPC1768 pin-to-pin Compatable. (Mbed)

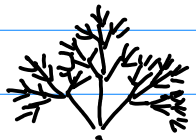
a. Step1. MCUXpresso IDE 1768 Binary Code.

Step2. "Firmware" Upload the binary file to the Flash. Need a prob

Step3. Interactive Debugging.

3° LPC11C24 Dig-Key in Stock. LPC1114 Site

GPP/SPI, FLASH (ON-Chip) 1/8 of the size Comparing to LPC1768/9



Homework (0 pt)

1. Form A Team By Wednesday. 4-Person

2. Select/Finalize your target platform. By the end of the week.

Example: Register File

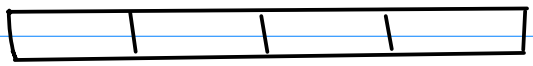
Special Purpose Registers

General Purpose Register

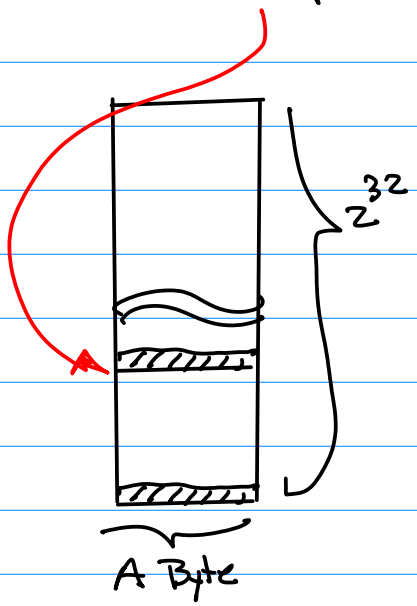
GPx CON

Root

Prefix 3 Letters for Port "x", x=0, 1, 2, 3



GPx CON 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)

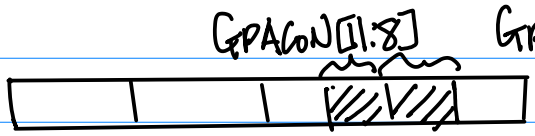
CmpE240

Sept. 7



For Example for Samsung ARM-11.

Consider:
Power up Address + Power Process.
Booting.




$GPx_{CON}$ its address is 32 Bits, it maps to the memory map.

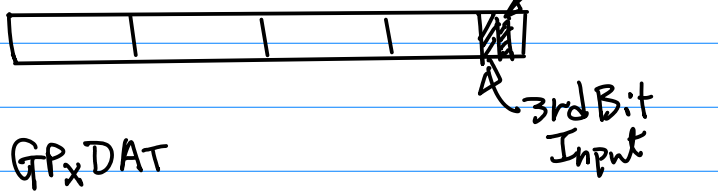
Design Requirements (Spec.)

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

To perform Init & Config.

GPACON[7:4] = 0001 = 0x1

GPACON[11:8] = 0000 = 0x0




$GPx_{DAT}$

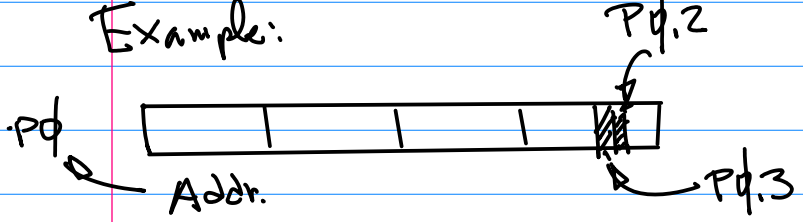
GPACON[3:0] = 0x10

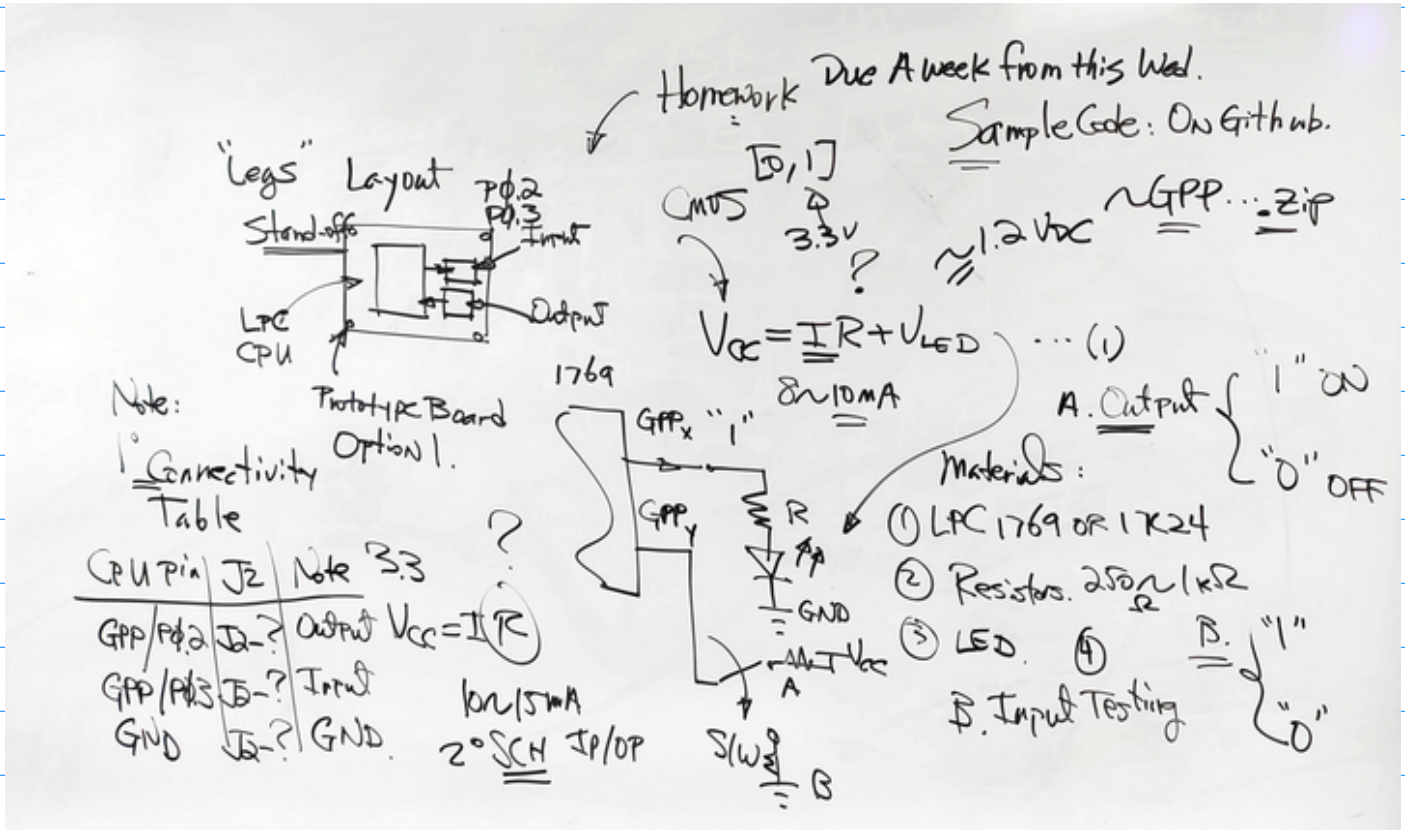
Sept. 12 (monday)

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

Note: Target platform.
LPC1768, LPC1124

Example:






Sept. 14 (Wed).

Note: 1° Check Homework Assignment on CANVAS.

Two Options { Prototype Board
{ e-Bay, Board B.

Topics today: IDE

1° GPP Software/Program

2° 2D Graphics Processing Engine Design.

Example: Set up the Expresso. 2

key points:

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

2° C/C++ Project settings. →
"Semihost"

3° Import LPC1769 patch.

1769 patch.zip

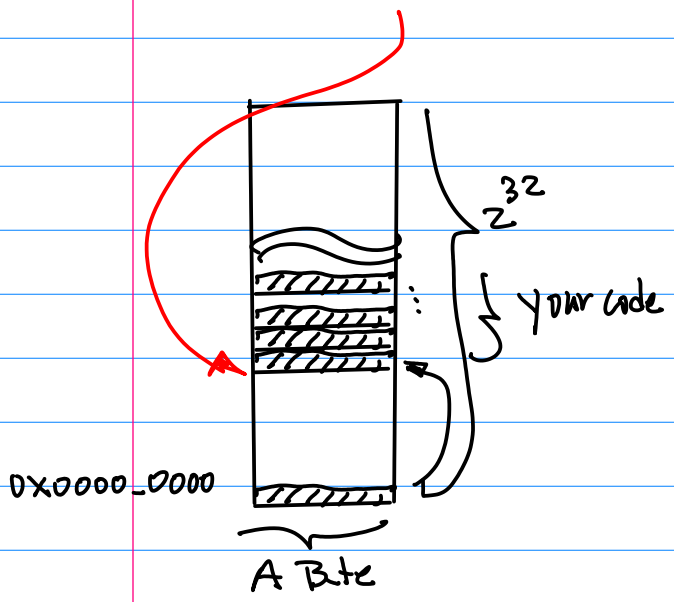
Note: 1° LPC1769 patch is already Config by NXP.

2° prob issue → fix:

Reconnect OR Reboot.

Import GPIO project to your MCU Xpresso.

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




Consider G.E. Design.

Math. Formulation, for Vector Graphics.

Example.

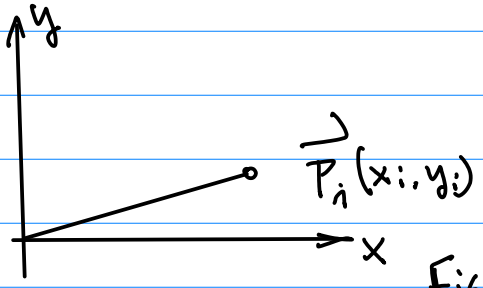



Fig. 1.

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

Also, a line $\rightarrow (x_i, y_i)$

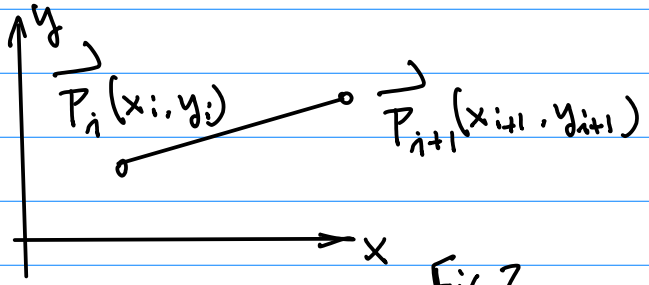



Fig. 2.

Note: To Uniquely Define A Line, we can add a directional vector, Denoted as

Definition 1:

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

$\vec{P}_{i+1}$: Ending pt. $\vec{P}_i$: Starting pt.

Definition 2: (Line Eqn. in Vector Form)

$$\vec{P}(x,y) = \vec{P}_i(x_i, y_i) + \lambda \vec{d}(x,y) \quad \ldots (2)$$

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

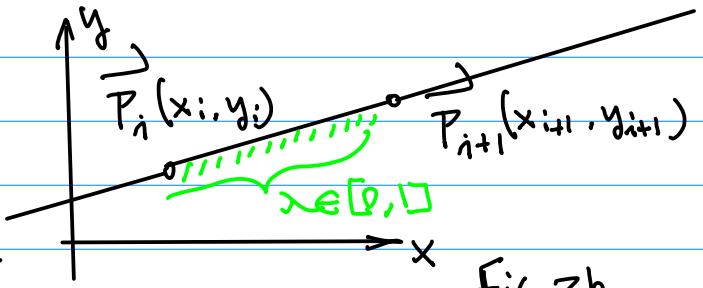



Fig. 2b.

Observation 1:

When $\lambda = 0$, Eqn (2) gives the Starting pt: $\vec{P}_i(x_i, y_i)$

$\lambda = 1$, .. .. 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:

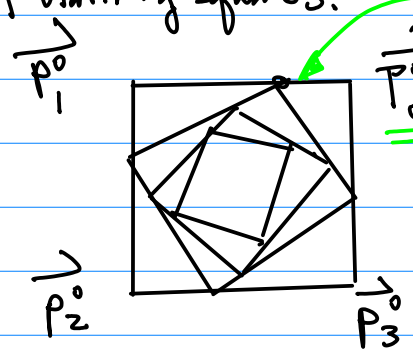
1° Check CANVAS for the Submission of the In-Class Exercise;

2° Graphics Lib has been ported to LPC11C24, Ref. Code & PPT will be provided.

Example: Given the equation Below,

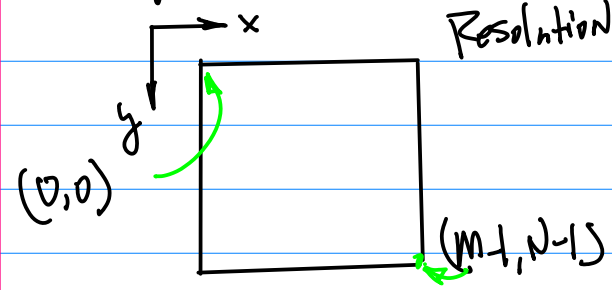
$$\vec{P}^{j+k}_{i}(x, y) = \vec{P}^{j}_{i}(x^{j}_{i}, y^{j}_{i}) + \lambda\left(\vec{P}^{j}_{i+1}(x^{j}_{i+1}, y^{j}_{i+1}) - \vec{P}^{j}_{i}(x^{j}_{i}, y^{j}_{i})\right) \quad \ldots (1)$$

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 $\{\vec{P}_i(x_i, y_i) \mid i = 0, 1, \ldots, 3\}$ for Resolution of A Display Device defined as $M \times N$

$\max(M, N) \leq 200$, Select the Size of the Cube $\simeq 50$




M x N
↑ ↑ No. of Rows
No. of Pixels per Row (Col).

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

$\vec{P}^0_0(x^0_0, y^0_0), \vec{P}^0_1(x^0_1, y^0_1), \ldots, \vec{P}^0_3(x^0_3, y^0_3)$ are defined as the same data pts on pp15.

Step 2. Get Line Equation, Based on Eqn (1). First pt on Level 1

$$\vec{P}^1_0(x^1_0, y^1_0) = \vec{P}^0_0(x^0_0, y^0_0) + \lambda\left(\vec{P}^0_1(x^0_1, y^0_1) - \vec{P}^0_0(x^0_0, y^0_0)\right)$$

$$= (60, 60) + \lambda\left((10, 60) - (60, 60)\right)$$

$$= (60, 60) + \lambda(-50, 0)$$

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

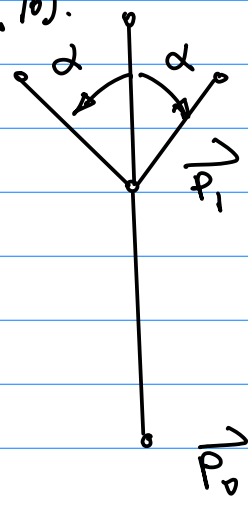
$$\begin{cases} x_i^{j+1} = x_i^j + \lambda(x_{i+1}^j - x_i^j) & \cdots (2a) \\ y_i^{j+1} = y_i^j + \lambda(y_{i+1}^j - y_i^j) & \cdots (2b) \end{cases}$$

```
x[i][j+1] = x[i][j] + lambda*(x[i+1][j] - x[i][j]);
y[i][j+1] = y[i][j] + lambda*(y[i+1][j] - y[i][j]);
```

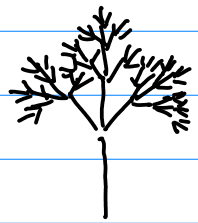
Consider Creating A Screen Saver
By Generating A tree.

Note: Level 0 : Tree Trunk;
Level 1 : { Branch (Main) : Mag. Reduction By 20% $\lambda = 0.8$ (Same Direction, Directional Vect. Same.)
Side Branch { L (CCW), Rotation by $\alpha$ ($\pi/6$). $\alpha > 0$
R (CW), $\alpha < 0$

$\alpha$ : Positive

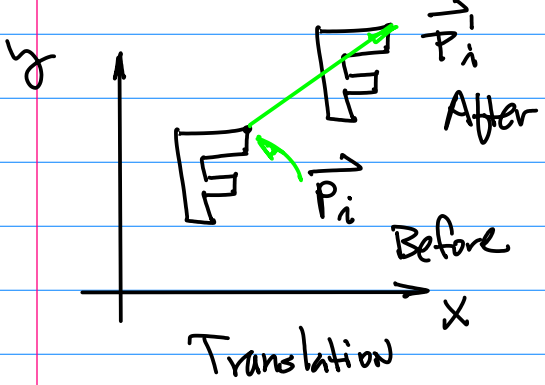


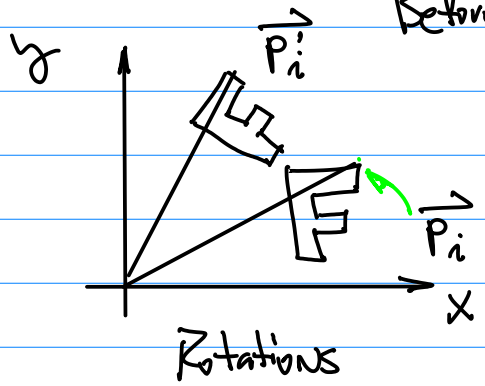

level 2, 3, ..., k.
Repeat Level 1 with Reference Vector ($P_i$) Updated Accordingly.



Background (2D Transformations)

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



Sept. 26.

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).

pt (After) $\vec{P_i'}$ pt (Before) $\vec{P_i}$

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

3D vector with One "Dummy" Dimension.

Rotation Matrix for Eqn (1b)

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \cdots (2)$$

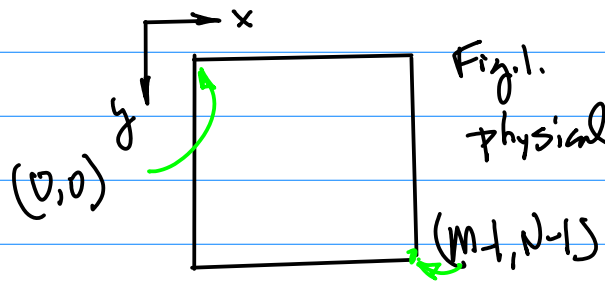
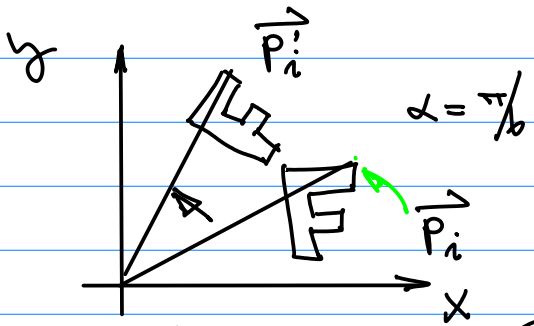



Fig.1. physical Display

$$\begin{cases} x_i' = x_i \cos\alpha - y_i \sin\alpha & \cdots (2\text{-}b) \\ y_i' = x_i \sin\alpha + y_i \cos\alpha & \cdots (2\text{-}c) \end{cases}$$

```
X_Prim[i] = X[i]*cos(alpha)
          - y[i]*sin(alpha)
```

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




Rotations

Fig.2 Virtual Display.

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

Example: for the Rotation illustrated in Fig.2.

pt (After) $\vec{P_i'}$ pt (Before) $\vec{P_i}$

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} \stackrel{?}{=} \Phi \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \cdots (1)$$

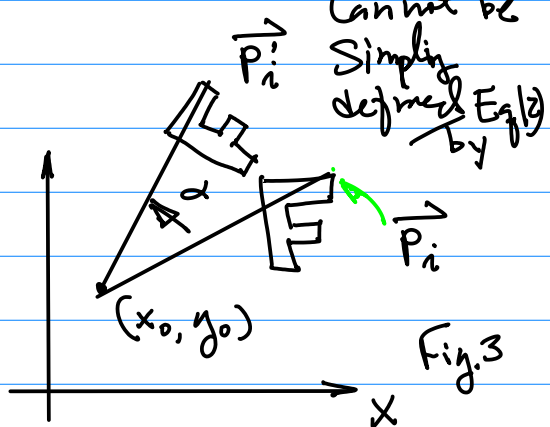



Fig.3

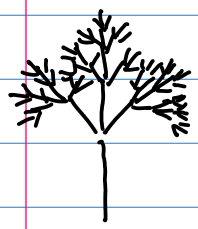


Fig. 4a

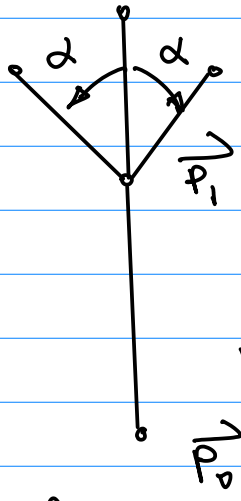



Fig. 4b.

These are different Rotations than that in Eqn (2).

Consider the Composition of 2D Transforms.

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

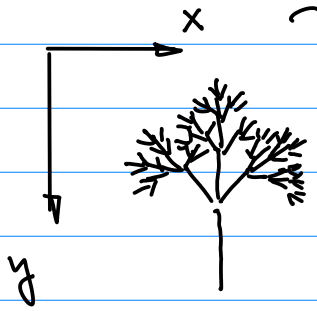



Fig. 5a. Physical Display

Mapping. $\phi_1$

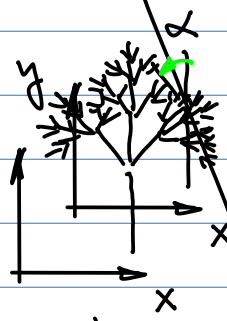



Fig. 5b. Virtual Display

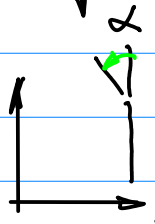



Fig. 5c. Rotation by α. CCW.

Mapping: Preprocessing — Translation. $\phi_2$ "M" matrix. $T_{3\times3}$

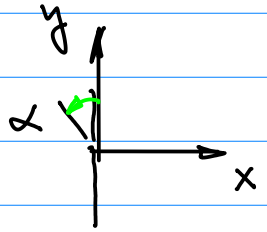
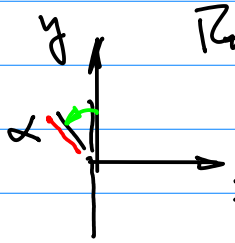



Fig. 5d. Preprocessing By Translation.

Rotation Eqn. (2).




Rotation: $\phi_3$ $R_{3\times3}$

Fig. 5e. Rotation: Done

Post Processing. To "move Back" to its original Position.

$\phi_4$ undo preprocessing $\phi_4 = \phi_2^{-1}$ or.

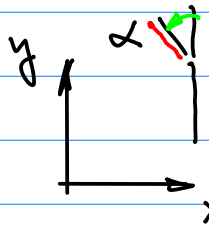


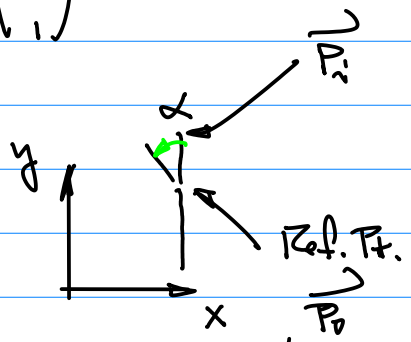

Fig. 5f.

$$\phi_4 = \phi_2\Big|_{\phi_2 = T_{3\times3}} = T_{3\times3}^{-1} \quad \ldots (3)$$

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

Start at given pt. to Be Rotated

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




Next. Preprocessing $\phi_2$, $T_{3\times3}$

To make $\vec{P}_0(x_0, y_0)$ to overlap with the origin $(0,0)$.

$$T_{3\times3} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \quad \ldots (5)$$

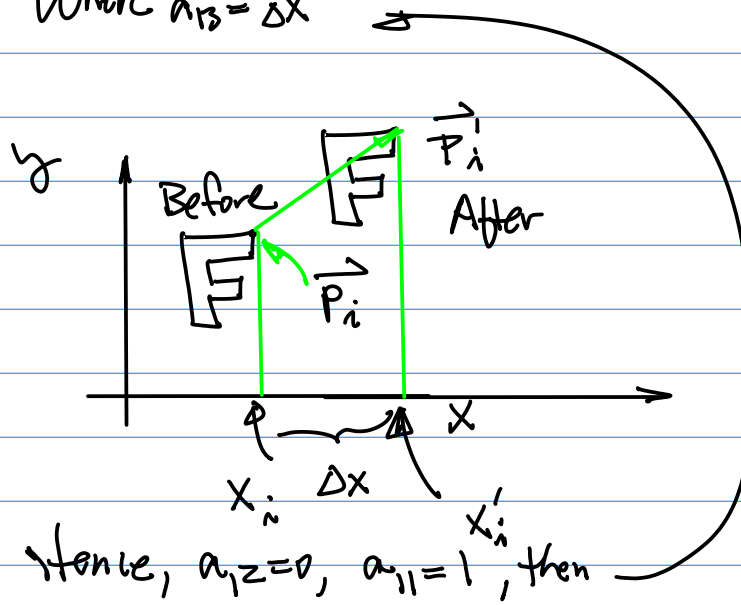
$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = T_{3\times3} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \ldots (6)$$

$P_i'$ After — Before

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

$$x_i' = a_{11}x_i + a_{12}y_i + a_{13}\cdot 1 \quad \ldots (6b)$$

Where $a_{13} = \Delta X$




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

So

$$x_i' = x_i + 0 + \Delta X = x_i + \Delta X \quad \ldots (6c)$$

$$y_i' = y_i + \Delta Y \quad \ldots (6d)$$

Therefore

$$T_{3\times3} = \begin{pmatrix} 1 & 0 & \Delta X \\ 0 & 1 & \Delta Y \\ 0 & 0 & 1 \end{pmatrix} \quad \ldots (7).$$

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

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

in Our Fig 5. Series.

Now, Rotation, $R_{3\times3}$.

Then, Post Processing. $\phi_4$

$$M_{\phi_3} = T_{3\times3}^{-1} = \begin{pmatrix} 1 & 0 & -\Delta X \\ 0 & 1 & -\Delta Y \\ 0 & 0 & 1 \end{pmatrix} \quad \ldots (8)$$

CmpE240
Sept.26, 22

Put All these together, we have

$$\begin{pmatrix} x_i'' \\ y_i'' \\ 1 \end{pmatrix} = T_{3\times3}^{-1} R_{3\times3} T_{3\times3} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \dots (9)$$

Rotated Post-Processed point.

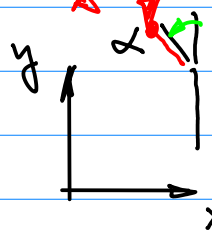



Fig.7

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

$XL[i] = \cos\alpha * X[i] - \sin\alpha * y[i] + \Delta x \cos\alpha - \Delta y \sin\alpha - \Delta x$

Similarly for $y_L$,

delta_x

$YL[i] = \sin\alpha * X[i] + \cos\alpha * y[i] + \Delta x \sin\alpha + \Delta y \cos\alpha - \Delta y$

Note: Work on Virtual to physical mapping

delta_y

Sept.28 Wed.

Note: Conclusion on 2D Transforms is given in the Above Eqn.(9), Coding/Implementation in C/C++.

Composition of

$$\begin{pmatrix} x_i'' \\ y_i'' \\ 1 \end{pmatrix} = \overbrace{T_{3\times3}^{-1} R_{3\times3} T_{3\times3}}^{T_\Sigma} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

$T_\Sigma$ is 3x3 matrix

Last Row of $T_{\Sigma(3\times3)}$ is (0,0,1) No Need for Coding.

Only the top 2 Rows from the matrix matter.

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

$\vec{P_i}$ ← Load

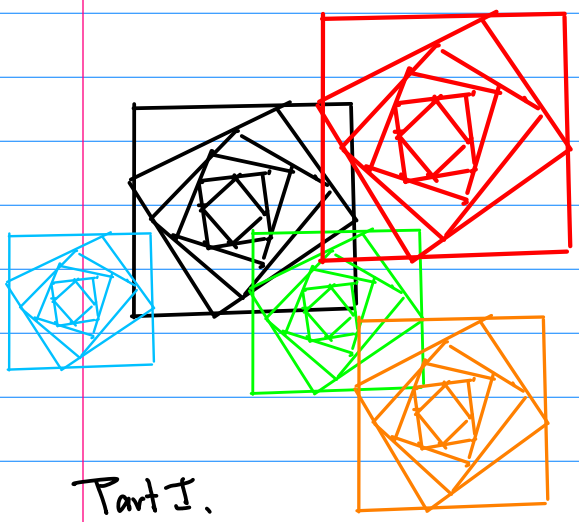
$i$ ← Enumeration of the Points.

$\vec{P_{i,l}}$ 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.



Part I.

Fig. 1

(Level 10 or higher)

Interface Between the LPC1769 and LCD is By S.P.I (Serial Peripheral Interface)


LPC1769 CPU — LCD

Part II. Trees. Optional for Artistic Presentation.

Fig. 2

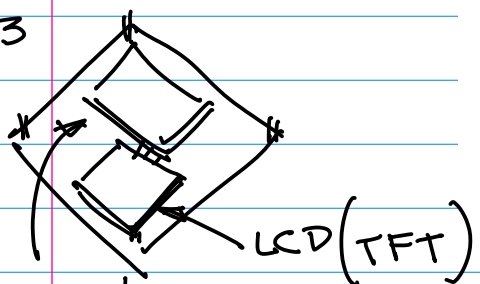
"3+1" pins for S.P.I.

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

Now, consider LPC1769, CPU Datasheet pp451. CRΦ 2021F-107b-sch-#LPC...

Consider the Graphics Engine Design from System View.

Fig. 3




LPC1769 or LPC11C24

Thin Film Transistor Technology

NXP Semiconductors — UM10360

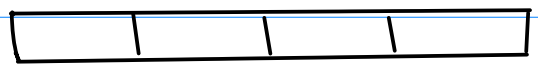
a. SSP → Chapter 18: LPC176x/5x SSP0/1

b. CRΦ, CR1 → Table 371: SSPn Control Register 0 (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 | |

(SSP0CR0)

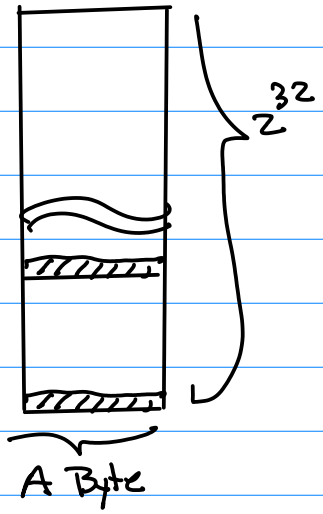
CRΦ



Addr. 0x4008_0000

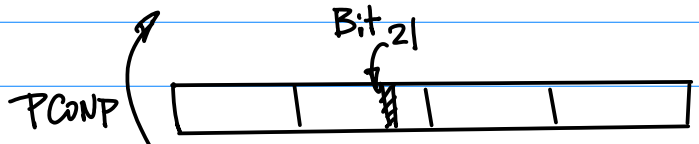
CMPE240

21




Find its memory Location at Which memory Bank?

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




Oct.5 (Wed)

Example: SPR (for SSP/SPI) CR0.

Note: 2nd STR: 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

TP.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, 12



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

| Bit | Symbol | Description |
|---|---|---|
| 1:0 | PCLK_QEI | Peripheral clock selection for the Quadrature E 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 f pin. For other functions, the direction is controlled automatically.

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

| 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 |

(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 | MOSI0 | 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)

```
172   /* Set DSS data to 8-bit, Frame format SPI, CPOL = 0, CPHA = 0, and SCR is 15 */
173   LPC_SSP0->CR0 = 0x0707;
```

0000 : 0111 : 0000 : 0111

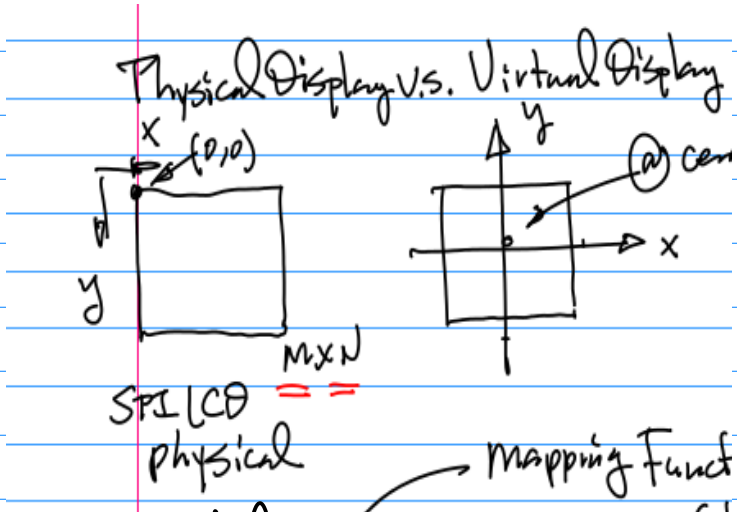
SCR | SPI | 8 bit

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

Consider Display Device Interface.

Background: Low Level Design of GUI
↓ Hardware Driver Design.
Physical Display v.s. Virtual Display.

Example: Physical Display v.s. Virtual Display.
pp 19. from Ref.
(github. Notes ~105~)



CMPE240

Oct.10, 22

$$x_p = x_v + M/2 \quad \cdots(7)$$

{ a Direction
{ b Offset

$$y_p \stackrel{?}{=} -y_v + \frac{N}{2} \quad \cdots(8)$$

(half of the No. of Rows)

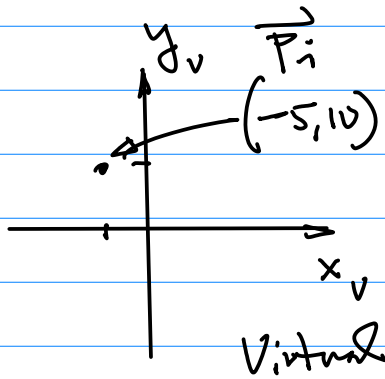



Fig 1. Virtual Display.

$$\begin{cases} x_p = x_v + M/2 & \cdots(7) \\ y_p = -y_v + N/2 & \cdots(8) \end{cases}$$

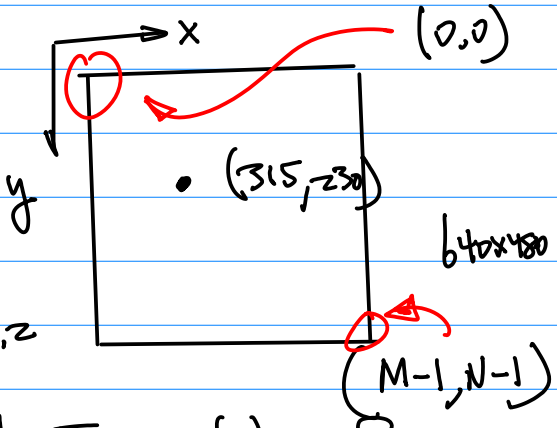



Fig.2

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/OR 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 MxN (640x480),

Find the New point $\vec{P_i}(x_i', y_i')$ After the transformation from the Virtual Display to the physical Display.

Sol: From Eqn(7) we have

$$x_p = x_v + M/2 \Big|_{M=640} = x_v + 640/2 = x_v + 320$$

Where we have $x_v = -5$, so

$$x_p = x_v + 320 = -5 + 320 = 315 ;$$

For $y_p$ from Eqn(8).

$$y_p = -y_v + N/2 \Big|_{N=480} = -y_v + 480/2 = -y_v + 240$$

$y_v = 10$, hence,

$$y_p = -10 + 240 = 230 . //$$

Now, Consider the Design of Display Driver (Hardware)

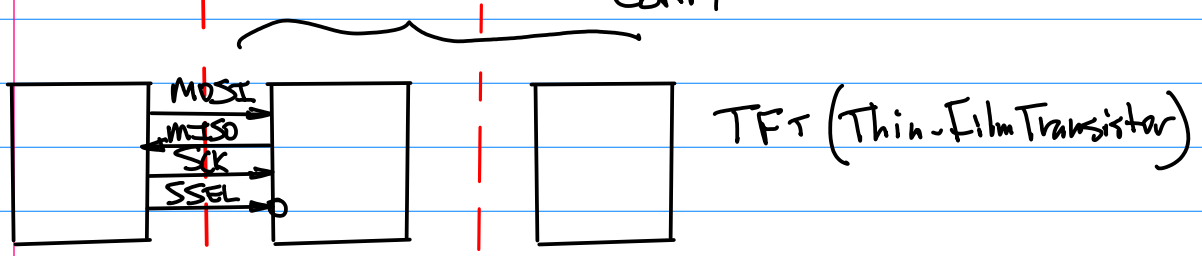
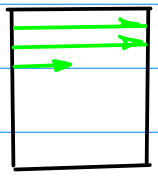



Fig.3

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

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

1. Sync. Signals
   - ① Frame Sync. Example: 30 F.P.S. $\text{Sync.}_F \rightarrow f_F = 30\,\text{Hz}$ ... (1)
   - ② Horizontal Sync. for M x N Resolution, it Repeats N Times Per frame.
   - ③ Data Sync. (Pixel) for M x N Resolution, Per Each Line, it Repeats M



Time to Scan Each Row is determined by Horizontal Sync.

M x N

2. Data Bus (Bi-Directional): 24 bits or Higher, 32 Bits Common
3. Control Signals. { Backlight, Enable }

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

Consider the Relationships Between $\text{Sync}_F$ & $\text{Sync}_H$, Denoted as $f_H$

$$f_H = N \cdot f_F \quad \cdots (2a)$$

$$\text{Sync}_H = N \cdot \text{Sync.}_F \quad \cdots (2b)$$

Consider $Sync._D$ V.S. $Sync._H$.

$f_D = m \cdot f_H$ ... (3a)

$Sync_D = M \cdot Sync_H$ ... (3b) M: No. of Cols.

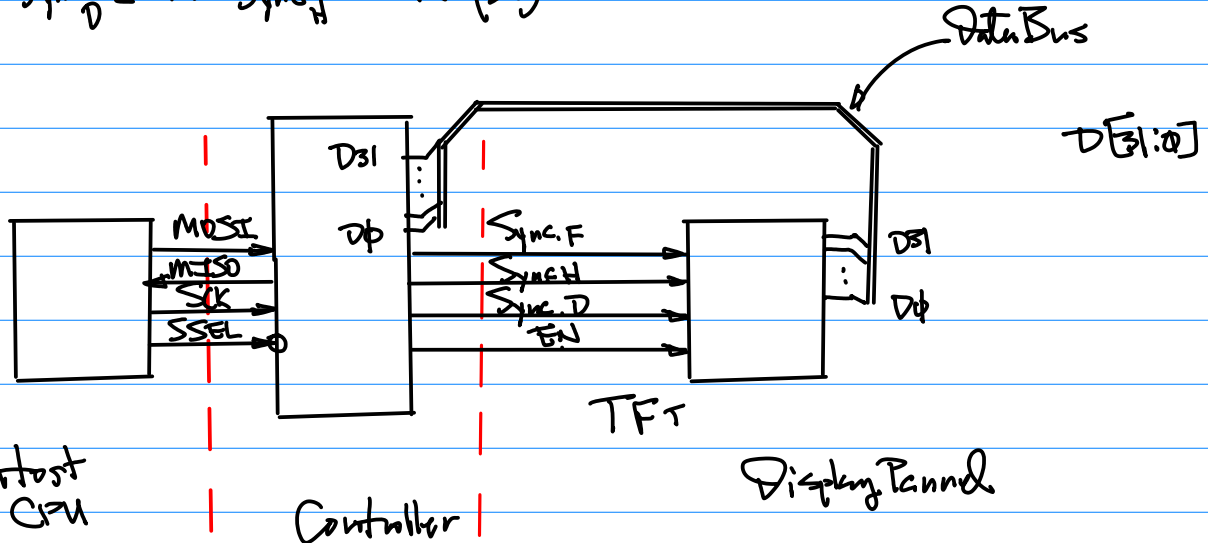
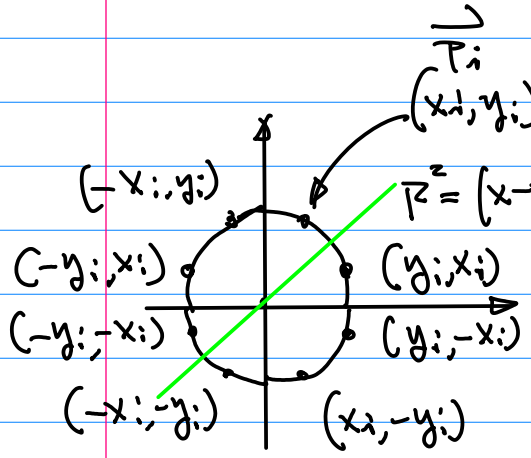


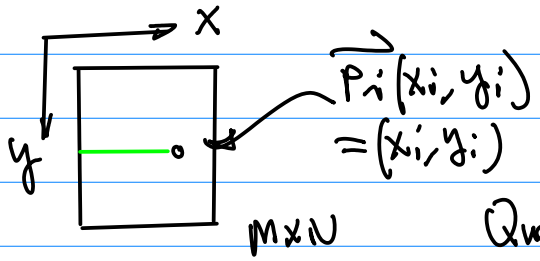

Fig.4




$r^2 = (x-x_0)^2 + (y-y_0)^2 \rightarrow r^2 = x^2 + y^2 \rightarrow$ Symmetric Property.

Oct. 12 (Wed)

Example:




Calculate the location information Based on Timing ($Sync._F$, $Sync._H$, $Sync._D$)

Assume Frame Rate 30FPS.

Question: How much time does it take for the Scanning to Reach line $y_i - 1$

$T_H = 1/Sync_H$ ... (1)

And $T = (y_i - 1) T_H \Big|_{T_H = 1/Sync_H}$

$= (y_i - 1)/Sync_H$ ... (2)

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

$T_D = 1/Sync_D$ ... (3)

$Sync_D = M\, Sync_H$

So

$T' = (x_i - 1) T_D = (x_i - 1)/M Sync_H$

... (4)

Hence

$T_\Sigma = T + T'$

$= (y_i - 1)/Sync_H + (x_i - 1)/(M Sync_H)$

... (5)

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), Ellipses etc.

No multiplication

$y = ax + b$

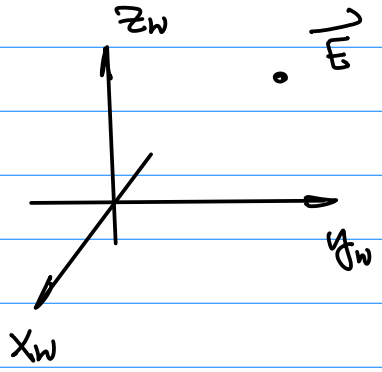
No. Sqrt Computation

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

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

Ref. github/hnalli/cmpe240 ...

...note.. May... Fig.1.




World Coordinate System.

RH: Right-Hand System Defined By Vector Cross Product

$\vec{X_W i} \times \vec{Y_W j}$

In Summary: 2D G.E. Design.

1. Display Driver
   - Architectural Design
   - Timing Analysis
     - $Sync_F$
     - $Sync_H$
     - $Sync_D$
2. 2D Transforms.
   - Translation
   - Rotation
   - Composition of the Above

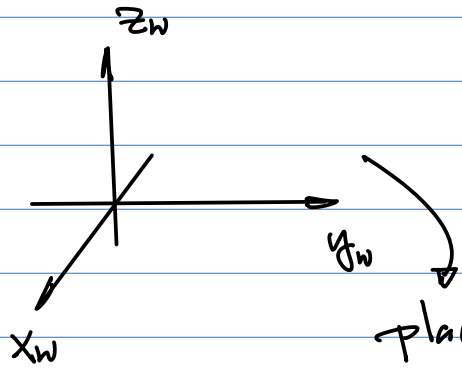



Fig.2

place A virtual Camera @ $\vec{E}(x_e, y_e, z_e)$

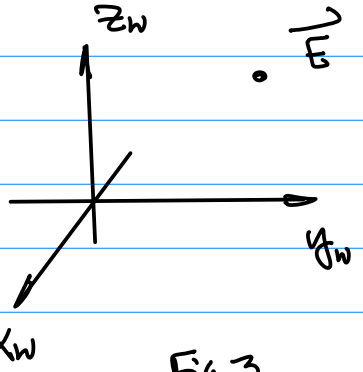



Fig.3

"pin-Hole" 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.

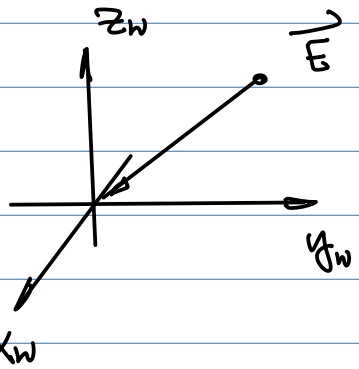



Fig.4

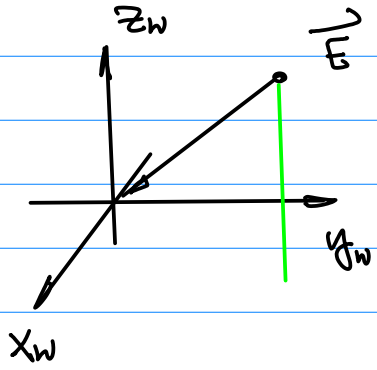



Fig.5

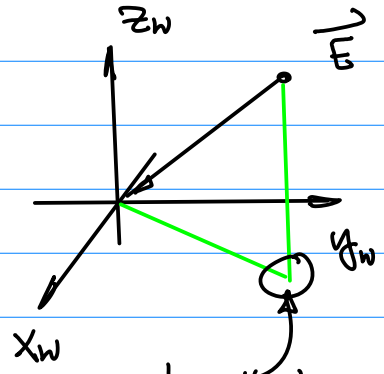



Fig.6

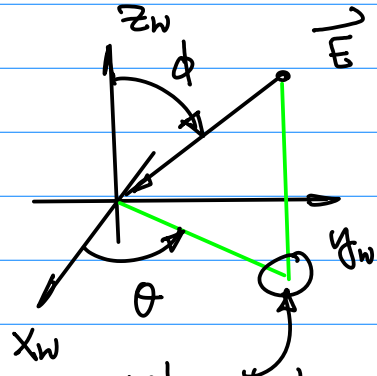



Fig.7

Angle $\theta$: ON $X_w$-$Y_w$ plane C.C.W as a Positive Angle

Angle $\phi$: Formed By $\vec{E}$ and $z_w$ Axis.

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

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

Oct. 17 (Monday).

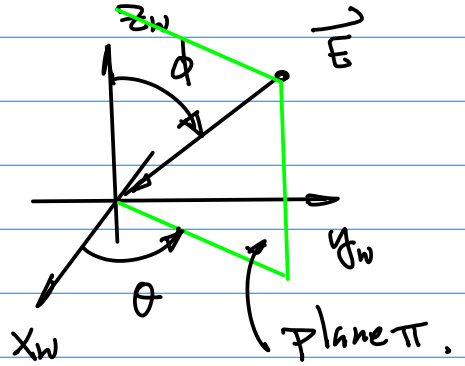



Fig. 8

Example: Perspective Projection.

Projection of 3D objects onto 2D plane.

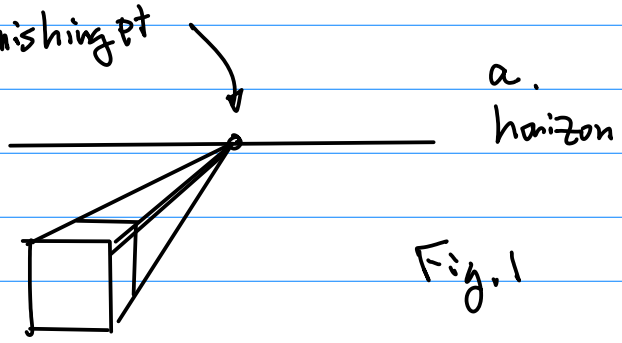



Fig. 1

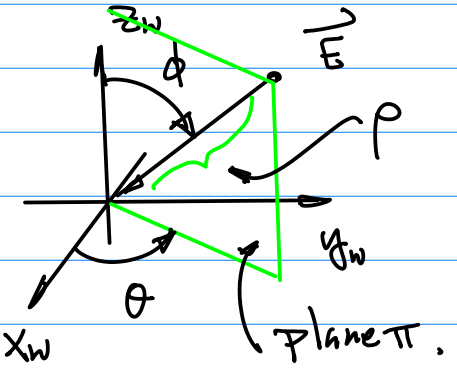


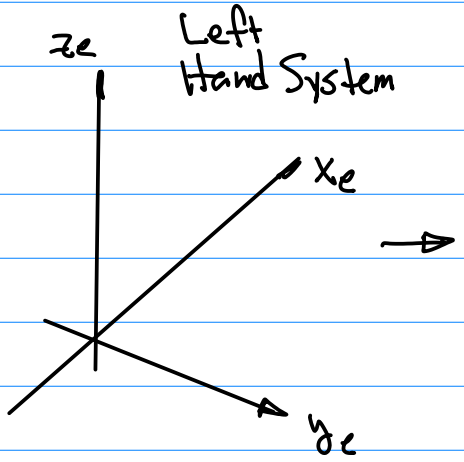

Fig. 9

c. Object $\{\vec{P_i} \mid i = 0, 1, 2, \ldots, N-1\}$, Connect Each pt $\vec{P_i}$ 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

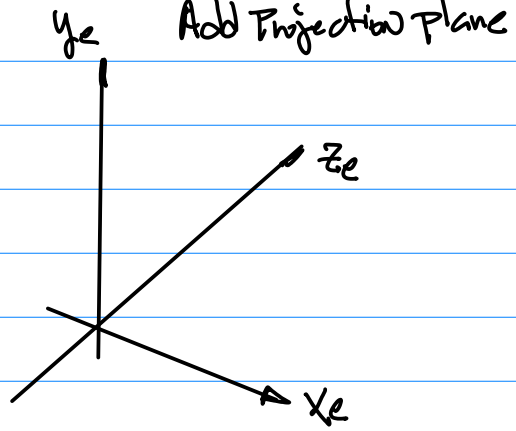
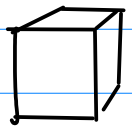



Fig. 10

CmpE240

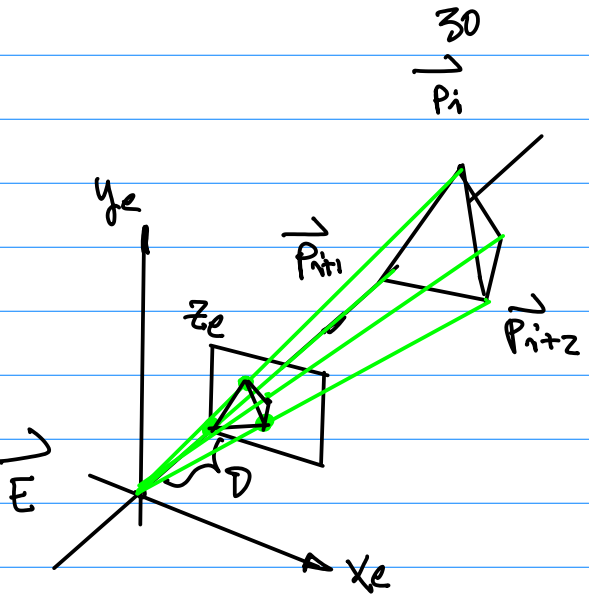
Oct.17,22



Perspective projection.

Fig.2.

Note: The $X_v$-$Y_v$-$Z_v$ System, Virtual Camera.



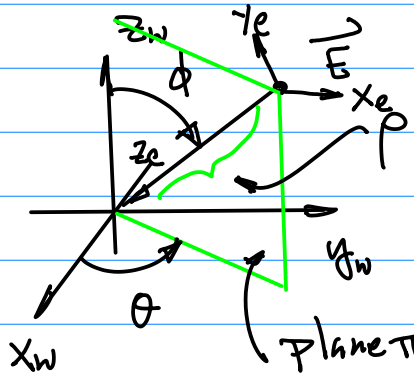




Fig.3

Transformation pipeline: $X_w$-$Y_w$-$Z_w$ to $X_v$-$Y_v$-$Z_v$ World 2 Viewer

Perspective Projection

$X_w$-$Y_w$-$Z_w$ → $X_v$-$Y_v$-$Z_v$ → P.P.

$\vec{P_i}(x_i, y_i, z_i)$ → $\vec{P_i'}(x_i', y_i', z_i')$ → $\vec{P_i''}(x_i'', y_i'', D)$ In Viewer

Drop D in Z so 3D with "Depth".

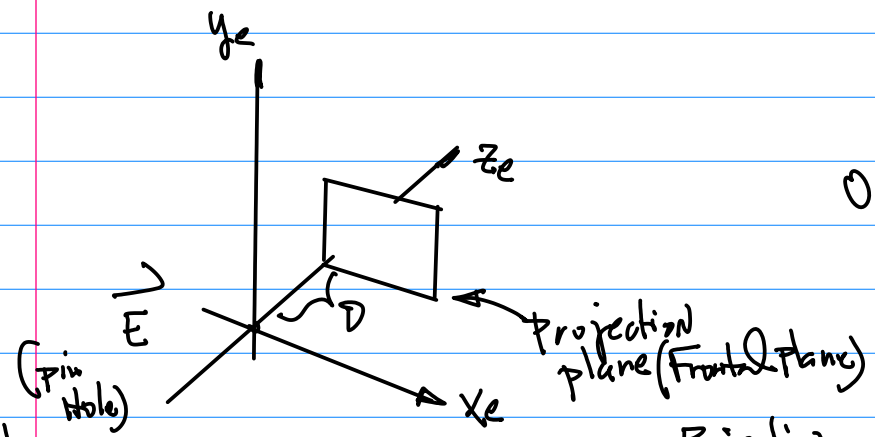
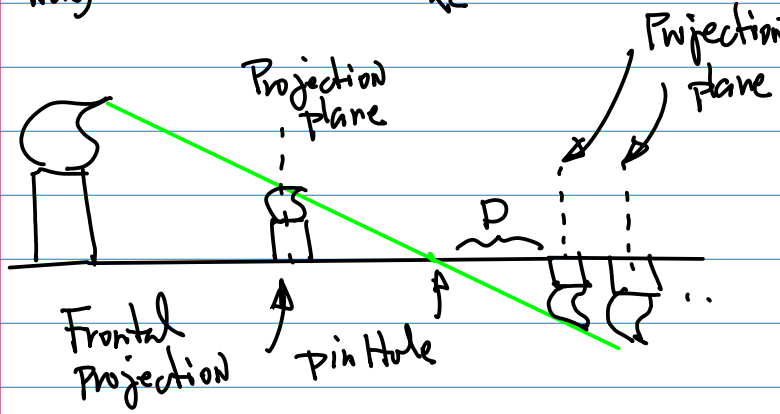



Fig.4




Oct.19(W)

Note:

1° Midterm Scheduled on Nov.7 (Monday).

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

1° Build (Implement C-Code on the Transformation Pipeline in the target platform.

2°. Build (Plot) $X_w$-$Y_w$-$Z_w$

Step 1.

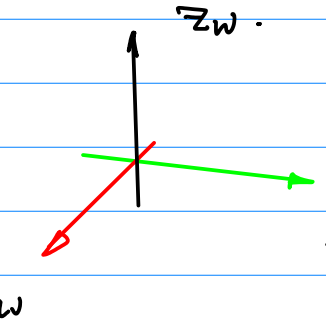



Fig. 1.

① Make $\theta \in [20, 50]$;

② Length of Each axis = 50 ~ 100

③ $\vec{E}(x_e, y_e, z_e) = (200, 200, 200)$

Vectors for $X_w, Y_w, Z_w$.

for example Starting pt (0,0,0)
Ending pt (100,0,0)
for $X_w$-axis

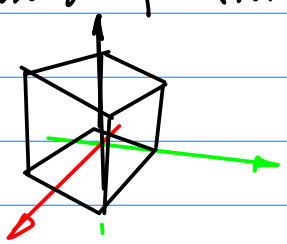
Note: Make sure the implementation of Virtual to physical Display is ready. (Eqn (7) & (8) in the previous Lecture Note).

Step 2. Design A Data set for a Cube.

$$\{\vec{P_i}(x_i, y_i, z_i) \mid i = 0, 1, 2, \cdots, N-1\} \quad \cdots (1)$$

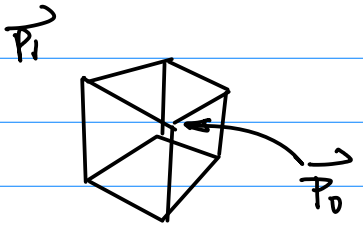
a. With Size as 100 x 100 x 100 as illustrated Below

b. And elevated By 10 from $X_w$-$Y_w$ plane



Pick An Arbitrary point $\vec{P_0}$

$\vec{P_0}(x_0, y_0, z_0) = (100, 100, 110)$

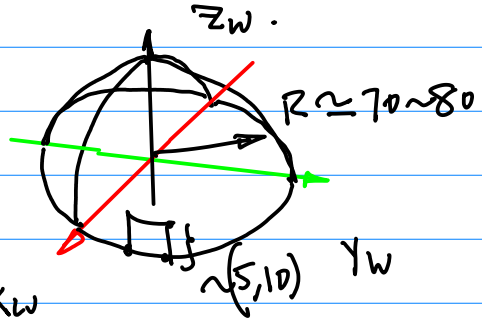



$\vec{P_1}(x_1, y_1, z_1) = (0, 0, 110)$, etc.

Displaying the Cube.

Step 3.

a. $R \simeq 80$




b. Size of the upper part = 80. (from R).

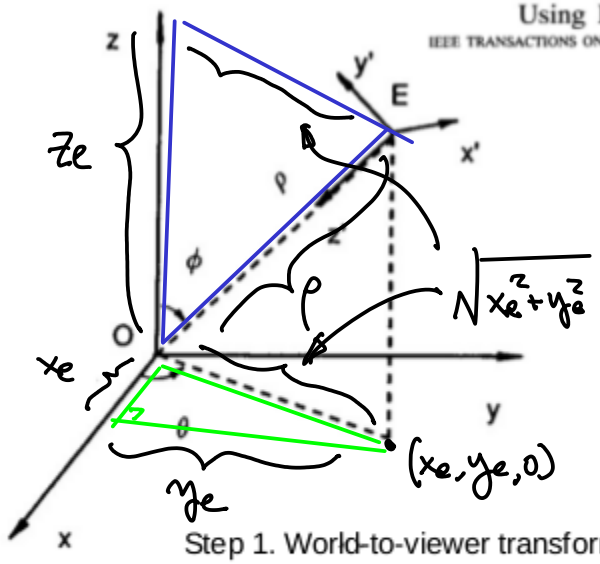
Note: Do Each "Ring", with Different R, then add Z value.

$$R^2 = \sqrt{(x - x_0)^2 + (y - y_0)^2} \quad \Big| \; x_0 = y_0 = 0$$

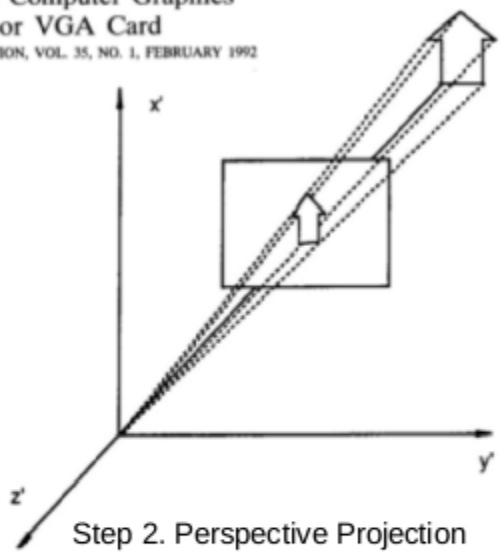
$R_i$ for $i = 1, 2, \cdots, 8$.

2018F-114-lec-3D-vectors-v3-2018-10-26.pdf

Reference: H. Li Three-Dimensional Computer Graphics Using EGA or VGA Card
IEEE TRANSACTIONS ON EDUCATION, VOL. 35, NO. 1, FEBRUARY 1992



Step 1. World-to-viewer transform



Step 2. Perspective Projection

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

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

Harry Li, Ph.D.

Points in $X_w$-$Y_w$-$Z_w$ : $\vec{P_i}(x_i, y_i, z_i)$
" in $X_v$-$y_v$-$z_v$ : $\vec{P_i'}(x_i', y_i', z_i')$
Points After Perspective Projection
$\vec{P_i''}(x_i'', y_i'')$

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

Oct. 24 (Mon).
Continued from The Transformation Pipeline.
Example:

After

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

Before

$x_i' = -\sin\theta \cdot x_i + \cos\theta \cdot y_i$ (X_prim[i] = -sin(theta) * X[i] + cos(Theta) * y[i];) ... (1)

$y_i' = -\sin\phi\cos\theta\, x_i - \cos\phi\sin\theta\, y_i + \sin\phi\, z_i$ ... (2)

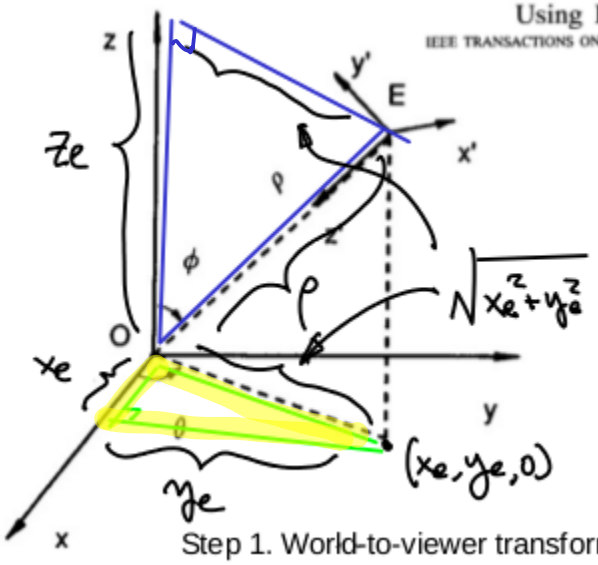
$$z_i' = -\sin\phi\cos\theta\, x_i - \sin\phi\cos\theta\, y_i - \cos\phi\, z_i + \rho \quad \dots (3)$$

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

Assume $E(x_e, y_e, z_e) = (200, 200, 200)$

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

Reference: H. Li Three-Dimensio
Using EG
IEEE TRANSACTIONS ON ED




Step 1. World-to-viewer transform

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

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

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

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

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

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

CmpE240

Oct. 24, 22



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

a. $\begin{bmatrix} - & & \\ - & - & \\ - & - & - \end{bmatrix}$ b. for $\theta$ $\begin{bmatrix} S & C & \\ C & S & \\ C & C & \end{bmatrix}$

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

Example: Perspective Projection

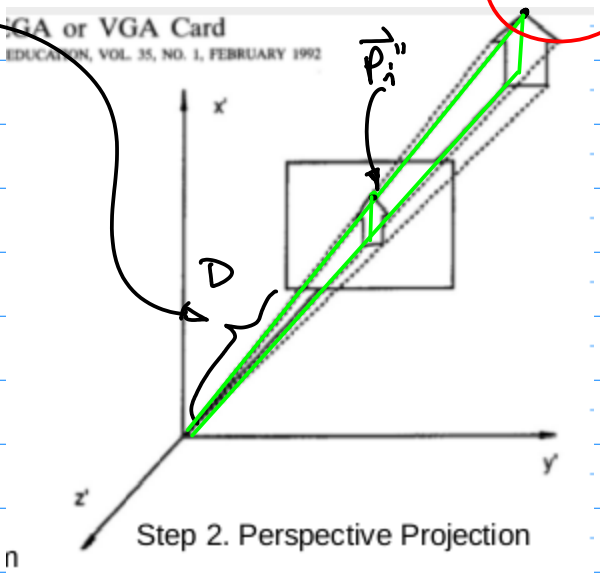
$$x_p = x_e\left(\frac{D}{z_e}\right)$$
$$y_p = y_e\left(\frac{D}{z_e}\right)$$

Mapping Before (In $X_v$-$Y_v$-$Z_v$) Virtual Camera Coordinate, and After (In 2D Display Space);

Note: $\frac{x_i'}{x_i''} = \frac{y_i'}{y_i''} = \frac{z_i'}{z_i''}$, $z_i''$ = D

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

$\frac{D}{z_e}$ :



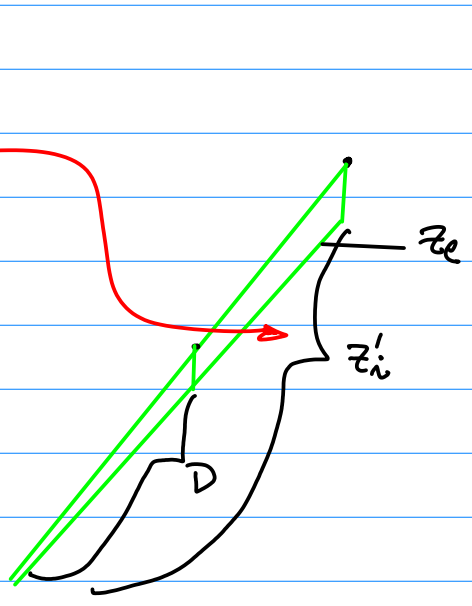
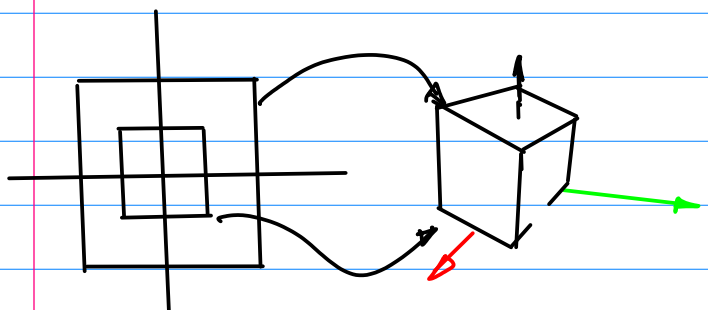




Fig.1.



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;
```

ρ

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;   // y-axis
```

```
127     //sin and cosine computation for world-to-viewer
128     float sPheta = Ye / sqrt(pow(Xe,2) + pow(Ye,2));
129     float cPheta = 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 10 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.

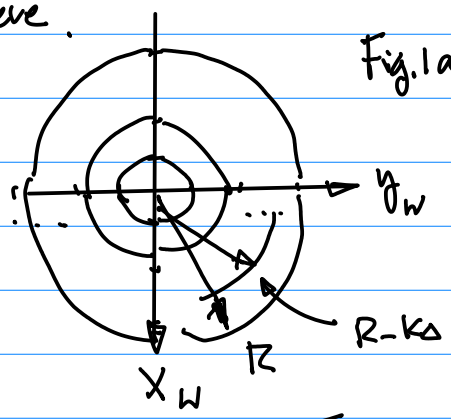



Fig. 1a.

Fig 1.

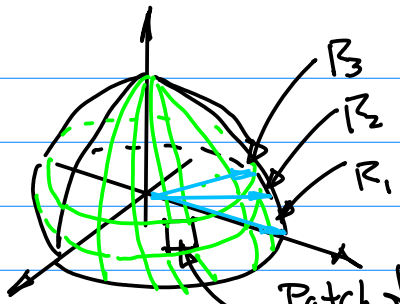



Fig.2

$\ell$: level of the cycle.
$\ell = 1, 2, \cdots, 8$

$\{\vec{P}_i^{\ell}, \vec{P}_{i+1}^{\ell}, \vec{P}_j^{\ell-1}, \vec{P}_{j-1}^{\ell-1}\}$

$R - \ell\Delta$ $\quad R - (\ell-1)\Delta$

$P_{i+1}^{\ell+1}$ $\quad P_i^{\ell+1}$

$P_j^{\ell}$ $\quad P_{j-1}^{\ell}$

Note: Project1 (Not Homework), 10 pt.
Due Nov. 2nd (W), Extended
2 Days.
Compute Transformation $(X_w\text{-}Y_w\text{-}Z_w \Rightarrow X_v\text{-}Y_v\text{-}Z_v)$

```
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;
```

$X_i' = -\sin\theta \cdot x_i + \cos\theta \cdot y_i$ (X_prim[i] = -sin(theta)*X[i] + cos(Theta)*Y[i];) ... (1)

$y_i' = -\sin\phi\cos\theta\, x_i - \cos\phi\sin\theta\, y_i + \sin\phi\, z_i$ ... (2)

$Z_i' = -\sin\phi\cos\theta\, x_i - \sin\phi\cos\theta\, y_i - \cos\phi\, z_i + \rho$ ... (3)

Perspective Projection

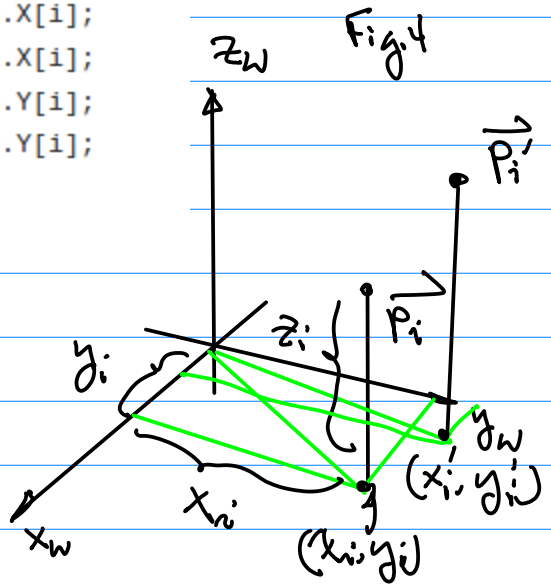
```
252         //*-------perspective projection--------------------*
253         perspective.X[i] = D_focal * viewer.X[i] / viewer.Z[i] ;
254         perspective.Y[i] = D_focal * viewer.Y[i] / viewer.Z[i] ;
255         if (perspective.X[i] > xMax) xMax = perspective.X[i];
256         if (perspective.X[i] < 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.4




Reference: H. Li Three-Dimensio
Using EG
IEEE TRANSACTIONS ON ED

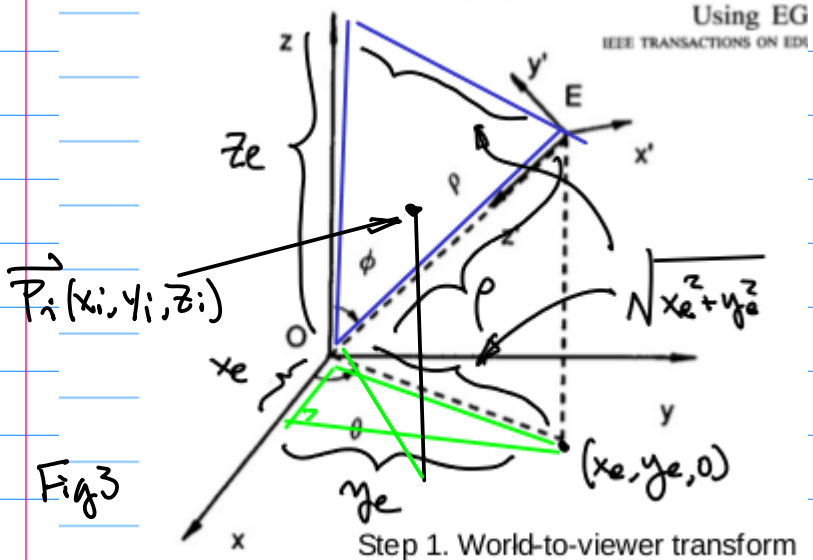



Fig3

Step 1. World-to-viewer transform

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

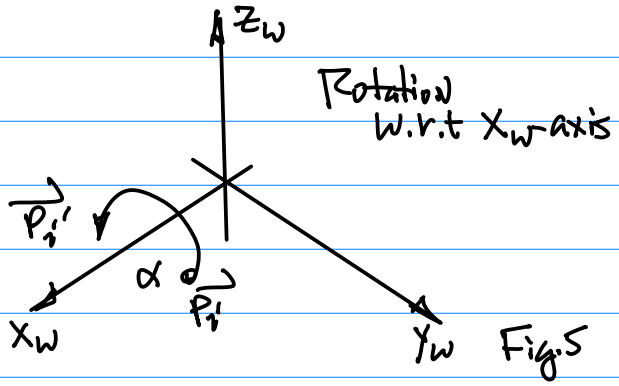
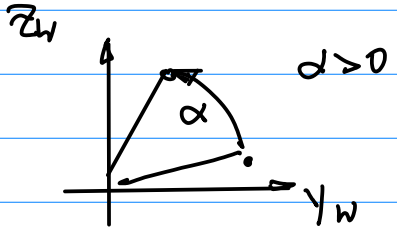
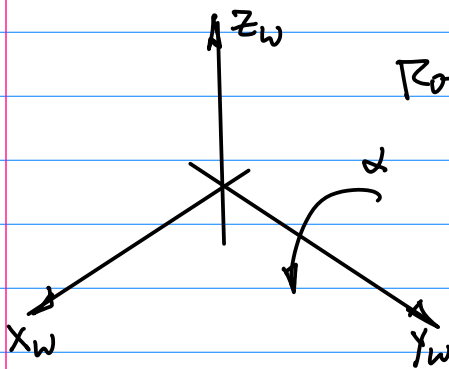



Fig.5

Translation:

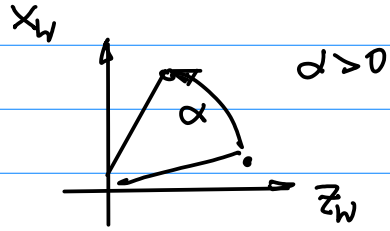
$$T_{4\times4} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \cdots(1)$$

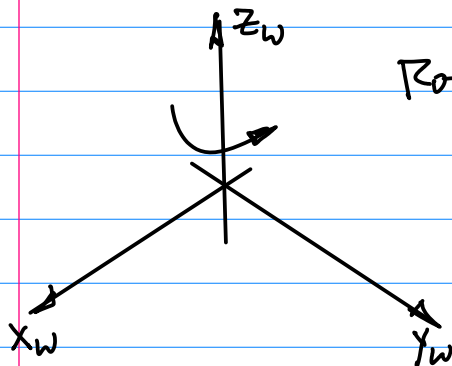





Rotation w.r.t $y_w$ axis

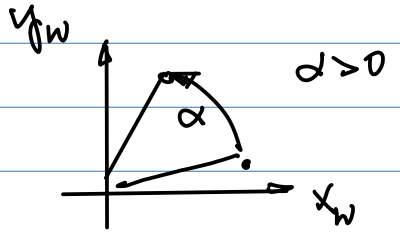
Fig.6

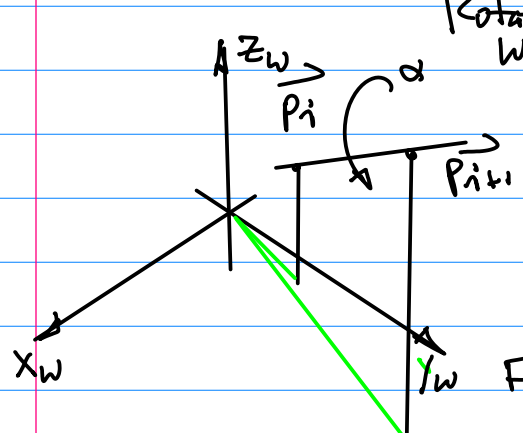



Rotation w.r.t $y_w$ axis

Fig.7




Rotation w.r.t an Arbitrary Vector

Fig.

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

c. Sphere Creation

Define Vertices to make Line Segments.

- $X_w$ line Segment
- $y_w$ line Segment
- $Z_w$ line Segment

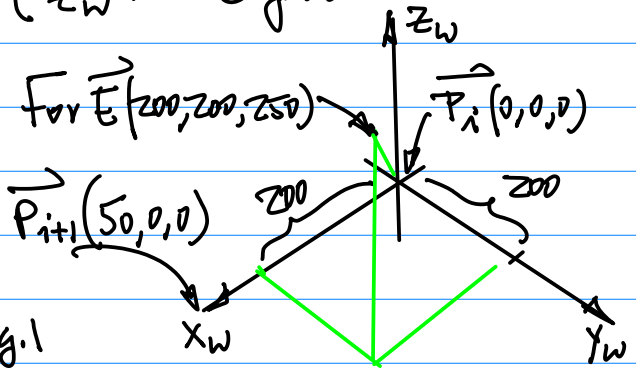
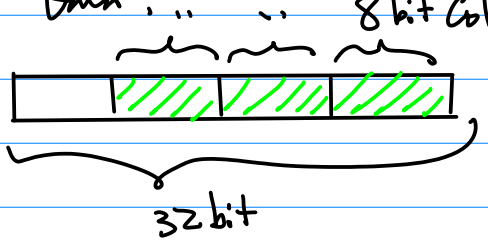



Fig.1

Note: Primitive Colors r, g, b are implemented as 32-bit Data




OpenCV: BGR; Deep Learning: RGB
Computer Vision;

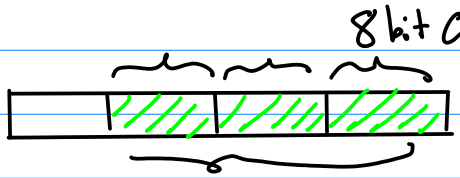



Fig.2

24 bits

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

Older Machine 15 bits or 16 bits Implementation.

Step2. $X_w$-$Y_w$-$Z_w$ to $X_v$-$y_v$-$z_v$
Perspective Projection

$\theta, \phi$
↓
$\cos\theta, \sin\theta$ ;
$\cos\phi, \sin\phi$ ; $\rho$

Step3. Virtual Display Coordinate System to the physical Display System (Reflects Your Actual LCD)

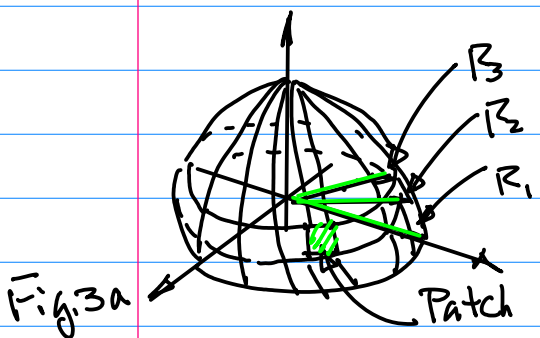



Fig.3a

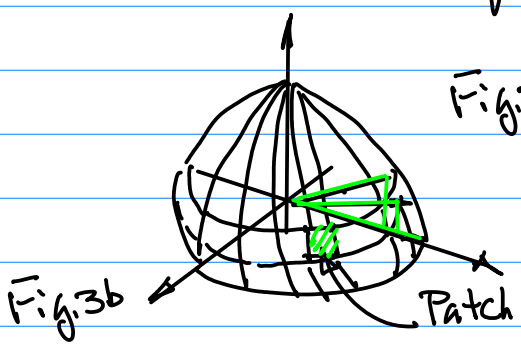



Fig.3b

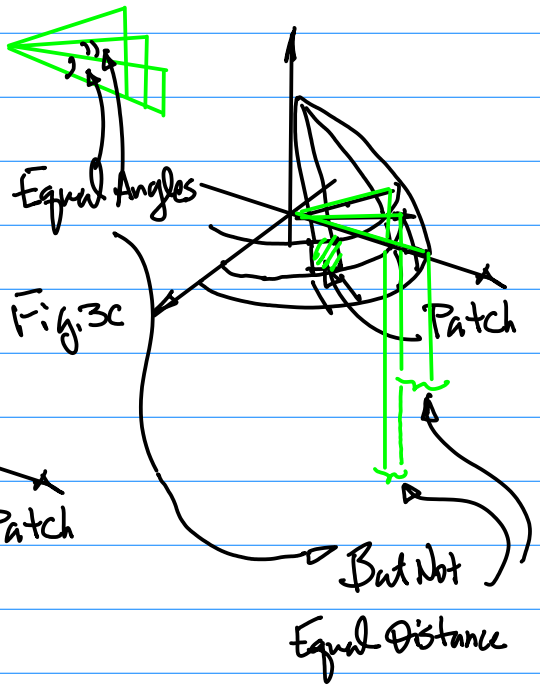


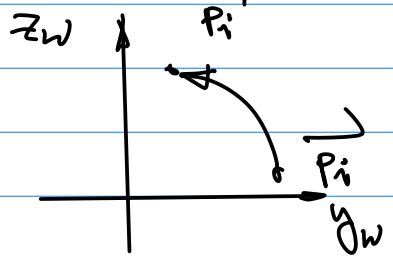

Fig.3c

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, $P_i$




Hence,

$$R_{x_w} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{pmatrix} x_i' \\ y_i' \\ z_i' \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \quad \cdots (1)$$

Note: In 2D Rotation

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

Indp. → $x_i'$, $y_i'$ ← Func.

Indp. → $x_i$, $y_i$ ← Func.

Nov. 2nd (Wed).

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

1° One Hour Exam. 3 Question.

Question: Architectural Aspects of the ARM GPU,
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.

Design = Question

Design
- Hardware
  - GPIO
  - SPI LCD I/F
  - Display Controller Design
    - $Sync._F$; $Sync._H$; $Sync._D$;
    - Draw Line. zip
- Software
  - Theoretical Discussion
  - Homework & Projects
    - 2D G.E.
      - Vector Formulation
      - Translations.
      - Rotations.
      - Compositions
    - 3D G.E.
      - 1. Transformation Pipeline
        World-2-Viewer;
        Perspective Projection
      - 2.

Virtual Display v.s. Physical Display.

CMPE240
Nov. 2nd, 22



1. Transformation Pipeline
   World-2-Viewer;
   Perspective Projection
2. Project: Implementation.
   Requires the execution of your code.

3. Exam. Paper will be posted on CANVAS.
   1 hr + 15 min
4. Paper Answer Sheets.

a) First_LastName_SID(4Digits)_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 Answersheets & photos

After projecting the Rotation onto $x_w$-$y_w$ plane

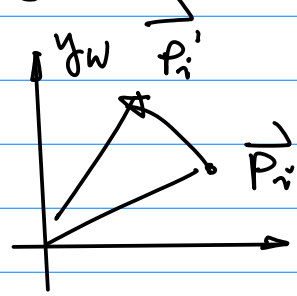


Fig 1b.

$$R_{z_w} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \ldots(1)$$

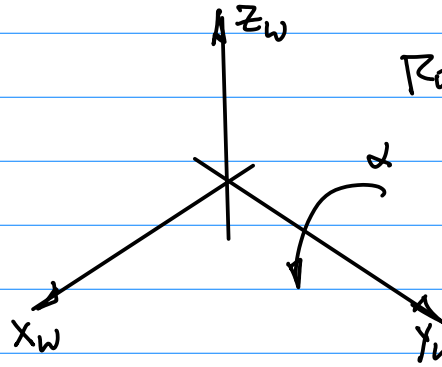


Fig. 2

Note: Projection onto $z_w$-$x_w$ plane.
$z_w$ — Indp, $x_w$ — Func.
$y_i$ is a constant (Before & After it stays the same).

Example: Continuation of 3D Rotations.

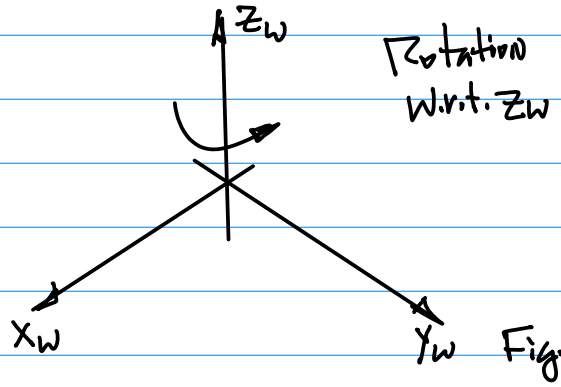


Fig. 1

$$R_{y_w} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \ldots(2)$$

CmpE240

Nov. 2nd, 22

$\vec{P_i} = (x_i, y_i, z_i)$, $\vec{P_{i+1}} = (x_{i+1}, y_{i+1}, z_{i+1})$ ... (3)

$\|\vec{P_{i+1}} - \vec{P_i}\| = \sqrt{(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2 + (z_{i+1}-z_i)^2}$ ... (4)

Rotation w.r.t an Arbitrary Vector

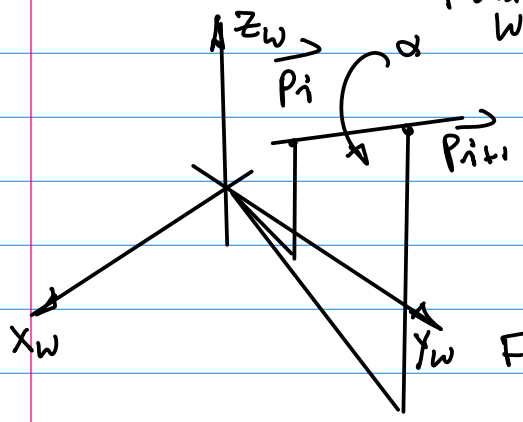



Fig. 3.

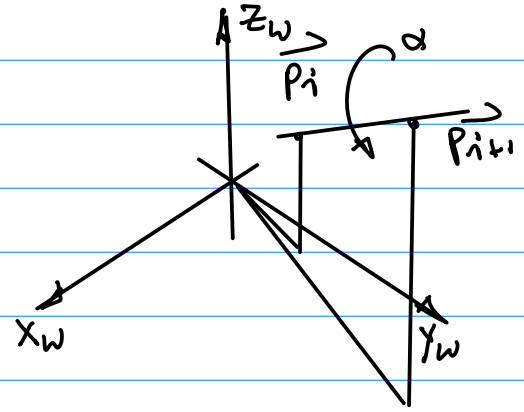
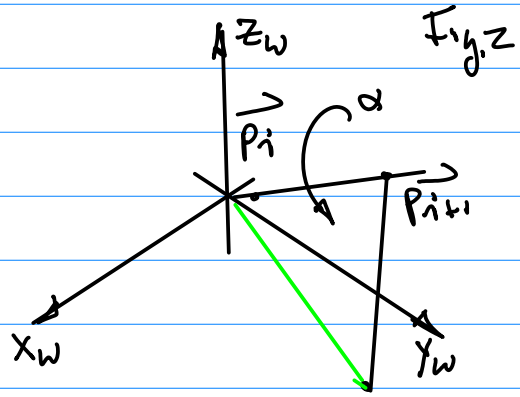



Fig. 2

Nov. 9 (Wed).

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

7 Steps
- Pre-processing: 3 Steps
- Rotation wrt $Z$-axis
- Post-processing: 3 Steps to undo the pre-processing.




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

Where $\Delta X = -x_i$ from $\vec{P_i}(x_i, y_i, z_i)$

$\Delta Y = -y_i$, $\Delta Z = -z_i$

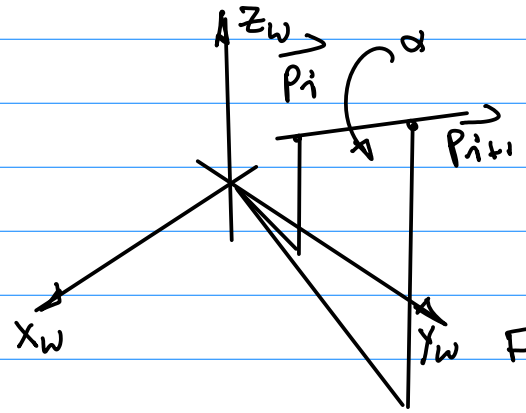



Fig. 1.

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

Step 2. Rotation wrt $Z_W$-axis, till the vector is ON $Z_W$-$X_W$.

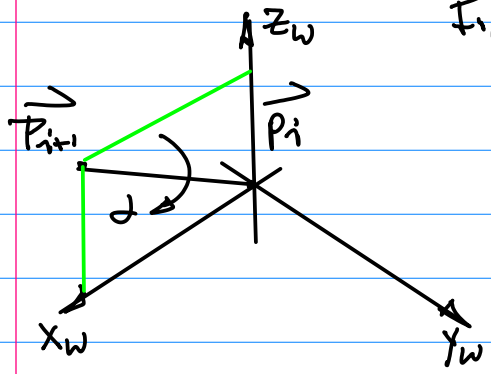



Fig.3a

$$R_{Z_w} = \begin{pmatrix} \cos\alpha' & -\sin\alpha' & 0 & 0 \\ \sin\alpha' & \cos\alpha' & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \cdots (2)$$

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

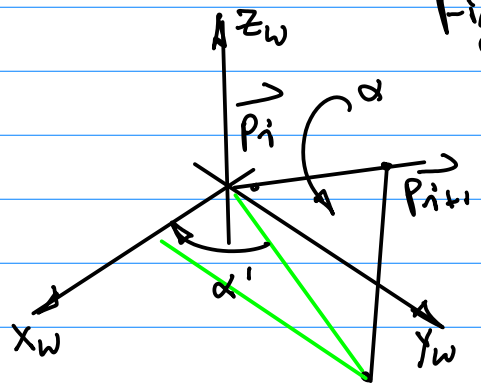
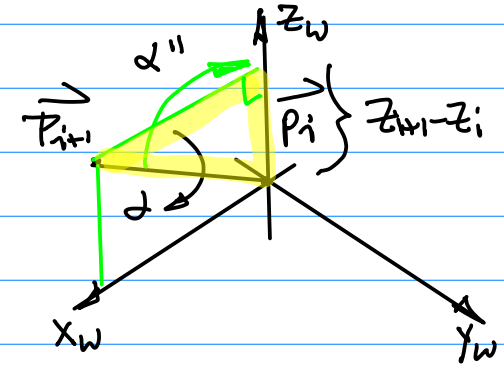



Fig.3b.

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

Step 3. Rotate the vector on $Z_w$-$X_w$ plane wrt $Y_w$-axis (Note $\vec{P}_i$ is hinged on the origin).




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

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

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

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,

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

Note:
For Release, we have

$T_{\Sigma_{Pre}} = R_{Y_w} R_{Z_w} T$ (4x4 matrix)

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

# CmpE240

Nov. 9, 22

Step 4

Now, the Actual Rotation $\alpha$

$$R_2 = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \cdots (4)$$

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

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

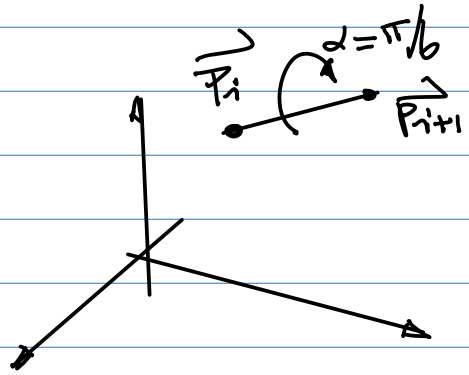
Post Processing:

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.

$$T_{\Sigma,Post} = T_7 R_6 R_5 \quad \cdots (8)$$



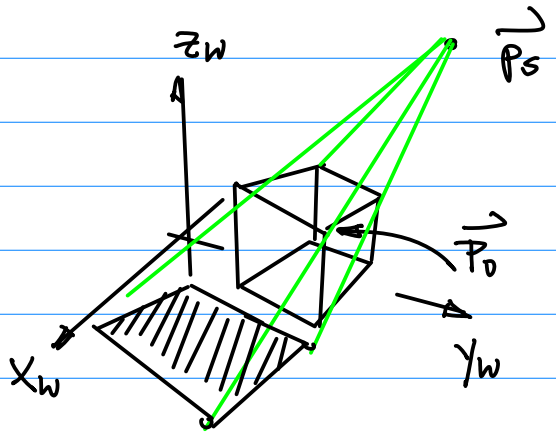

$$R_5 = R_{yw}^{-1} = \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \cdots (5).$$

$\vec{P_i}(100, 100, 300)$, $\vec{P_{i+1}}(200, 200, 310)$

Step 6. Similarly. (Undo Step 2)

$R_6 = R_2^{-1}$, e.g.

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




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

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 or 10 Degree
W.r.t An Arbitrary Vector of
your Choice with the consideration
of Shadow Computation, and
Diffuse Reflection;

2° Decoration of the Cube Surface

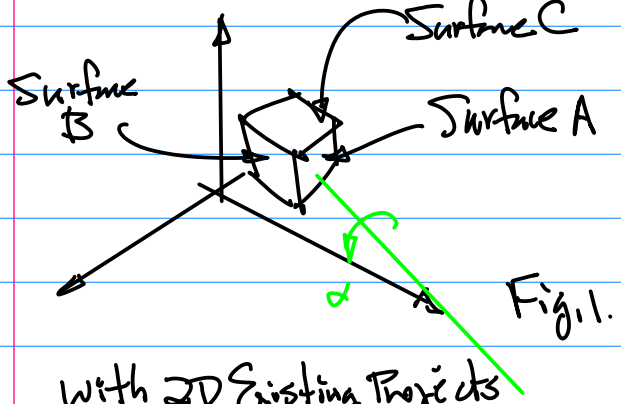


Fig.1.

With 2D Existing Projects
Rotating Squares, and Trees
Note: Keep the Sphere.

3° Shadow

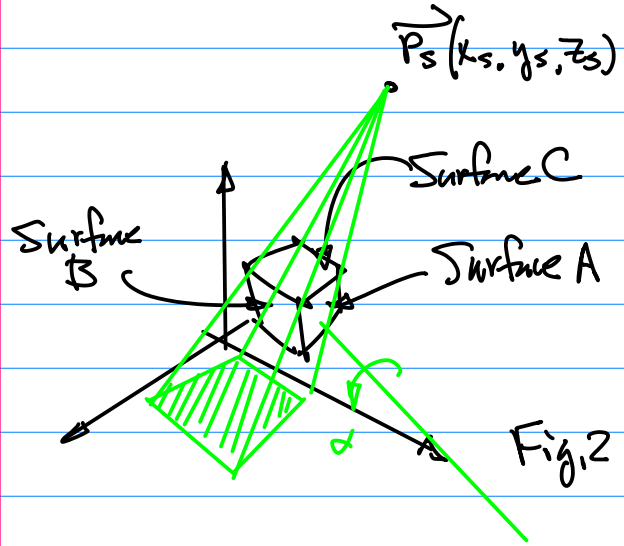


Fig.2

4. Diffuse Reflection.

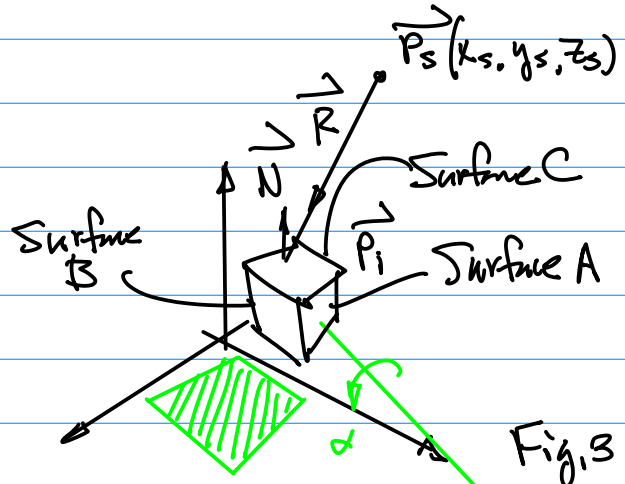


Fig.3

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

After P.P.

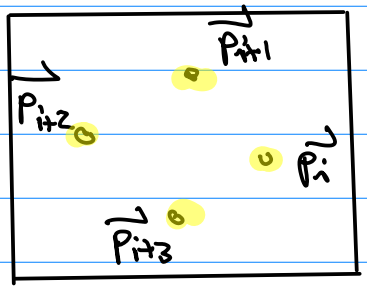


Fig.4

Step2. Compute the Color on the Boundary

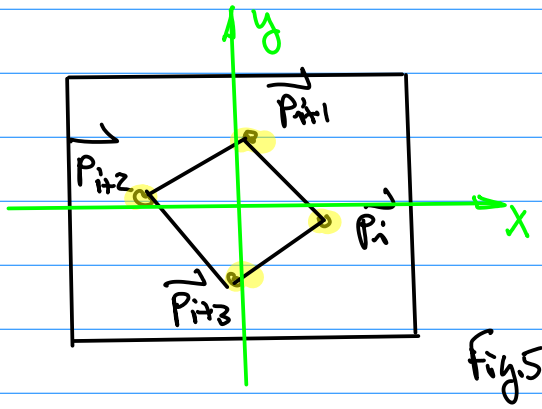
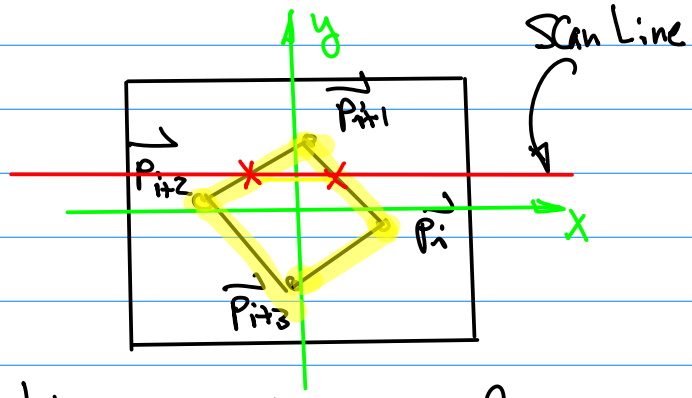


Fig.5

Step 3. Compute the Color Fig.6




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. $\{\vec{P}_i \mid i=1,2,\cdots,k\}$

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

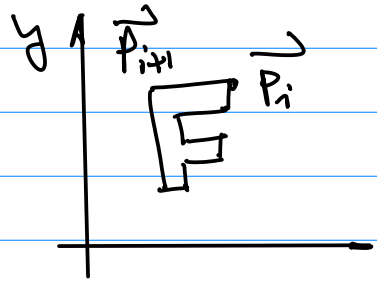



Fig.7a

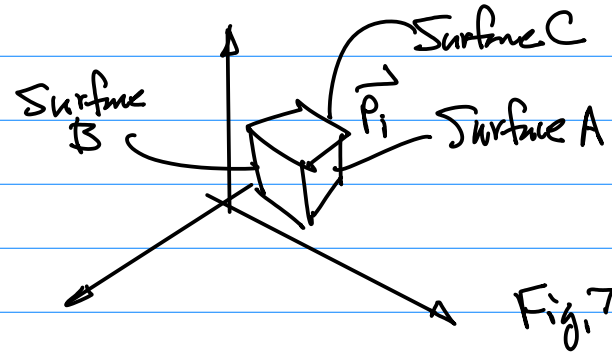



Fig.7b

First, to move 2D Pattern $\{\vec{P}_i(x_i, y_i) \mid i=1,\cdots,k\}$ 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 $X_w$-$Y_w$-$Z_w$

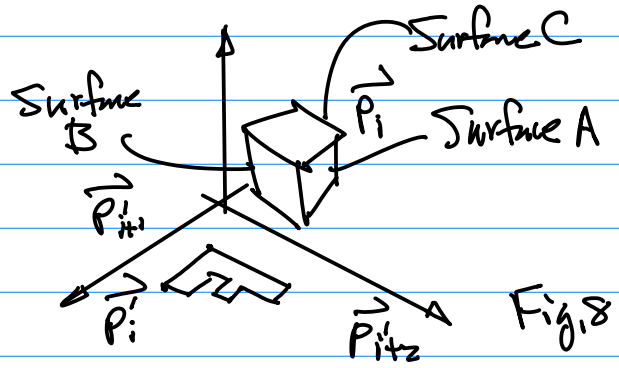



Fig.8

Step 2. Formulation to Decode Surface C.

| After | | Before |
|---|---|---|
| $x_i'$ | = | $x_i$ |
| $y_i'$ | = | $y_i$ |
| $z_i'$ | = | $C + H$ ...(1) |

$C$: Side of the Cube; $H$: Height of the floatation

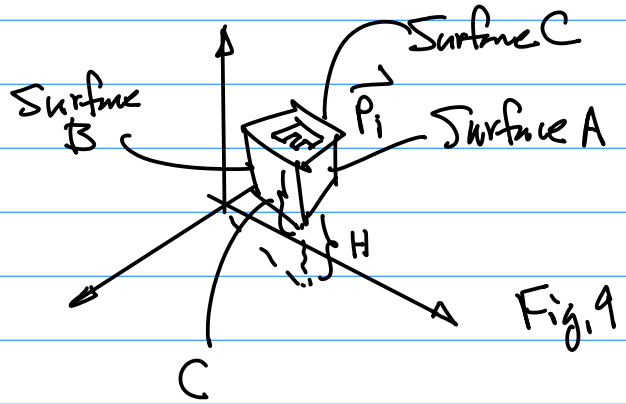



Fig.9

Example:
Decorate Surface B

After = Before

$y_i'$ = $X_i$ Ind.

$z_i'$ = $y_i$ Func

$x_i'$ = C (side of the cube) ... (2)

For Surface A:

After = Before

$z_i'$ = $X_i$ Ind.

$x_i'$ = $y_i$ Func

$y_i'$ = C ... (3)

Nov.16 (Wed).
Discussion On Ray Equations for 3D G.E., for Shadow Computation

Ref:

Formulation of the Ray Equation

Line Equation

$$\vec{P} = \vec{P_i} + \lambda(\vec{P_{i+1}} - \vec{P_i}) \quad \cdots (1)$$

Connecting Eqn (1) to Fig.1.

A Point Light Source

$$\vec{P_S} = (x_s, y_s, z_s) = \vec{P_{i+1}}$$

therefore,

$$\vec{R} = \vec{P_i} + \lambda(\vec{P_S} - \vec{P_i}) \quad \cdots (1\text{-}b)$$

$$(x_R, y_R, z_R) = (x_i, y_i, z_i) + \lambda (x_s - x_i, y_s - y_i, z_s - z_i)$$

OR for Coding

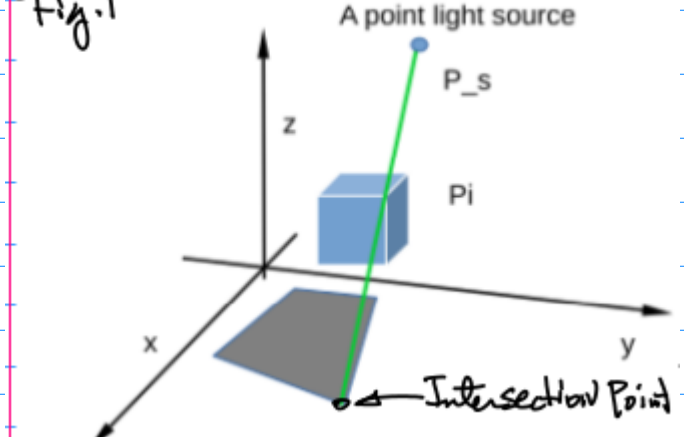
$$\begin{cases} x_R = x_i + \lambda(x_s - x_i) \\ y_R = y_i + \lambda(y_s - y_i) \\ z_R = z_i + \lambda(z_s - z_i) \end{cases} \quad \cdots (1\text{-}c)$$

Formulation of the plane Equation. $\pi$

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

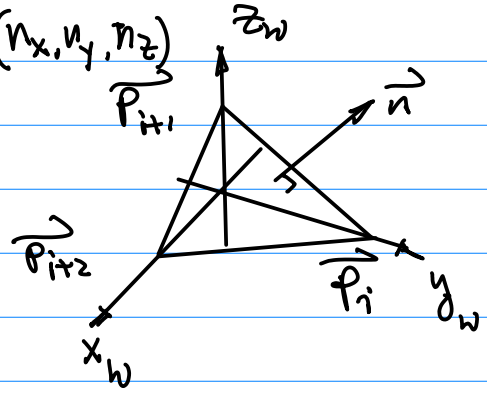
CMPE240-Adv-Microprocessors / 2018F / 2021F-101b-notes-cmpe240-2021-12-1.pdf

Fig.1




First, Define A Normal vector $\vec{n}(x_n, y_n, z_n)$

Or $(n_x, n_y, n_z)$



CMPE240

Nov. 16, 22

Using 2 vectors (Lines), each is

$\vec{P_i} - \vec{P_{i+2}}$,

$\vec{P_{i+1}} - \vec{P_{i+2}}$

To perform Cross Product,

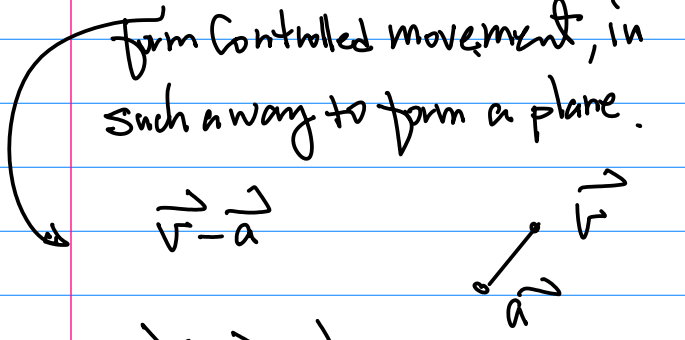
$$\vec{n} = (\vec{P_i} - \vec{P_{i+2}}) \times (\vec{P_{i+1}} - \vec{P_{i+2}}) \quad \cdots (2)$$

Now, Simplification, Consider $x_w$-$y_w$ plane.

$$\vec{n}(n_x, n_y, n_z) = (0, 0, 1) \quad \cdots (3)$$

Secondly,

A line and normal vector to form Controlled movement, in such a way to form a plane.




$$\vec{n} \cdot (\vec{v} - \vec{a}) = 0 \quad \cdots (4)$$

where $\vec{a}$ is known

$\vec{v}$ is an Arbitrary point on $\pi$

To Find the Intersection pt on $x_w$-$y_w$ plane,

$$\begin{cases} \vec{r} = \vec{P_i} + \lambda(\vec{P_s} - \vec{P_i}) \cdots (5a) \\ \vec{n} \cdot (\vec{v} - \vec{a}) = 0 \quad \cdots (5b) \end{cases}$$

Substitute $\vec{r}$ into the plane equation (5b), Solve for $\lambda$, we have

$$\lambda = \frac{\vec{n} \cdot \vec{a} - \vec{n} \cdot \vec{P_i}}{\vec{n} \cdot (\vec{P_s} - \vec{P_i})} \quad \cdots (5c)$$

(see Ref. pp. 51)

Coding:

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

$$\frac{(n_x, n_y, n_z) \cdot (a_x, a_y, a_z) - (n_x, n_y, n_z)(x_i, y_i, z_i)}{(n_x, n_y, n_z) \cdot (x_s - x_i,\ y_s - y_i,\ z_s - z_i)}$$

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

Note: $\lambda$ is Not the intersection Point, plug $\lambda$ Back to Eqn (1b). Find A Intersection Point

Example: Given a Single point light Source $\vec{P_s}(-20, 110, 200)$, $\vec{P_i}(100, 100, 110)$ Find intersection Point to plot Shadow.

Sol

From Eqn (5c), find $\lambda$.

Since

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

$\vec{P_i} = (100, 100, 110)$

$$\lambda = \frac{n_x q_x + n_y \cdot q_y + n_z \cdot q_z - (n_x x_i + n_y y_i + n_z z_i)}{n_x(x_s - x_i) + n_y(y_s - y_i) + n_z \cdot (z_s - z_i)}$$

$n_x = 0, n_y = 0, n_z = 1$.

Therefore, the above equation becomes

$$= \frac{0+0+0 - (0+0+1 \cdot z_i)}{0+0+1 \cdot (z_s - z_i)}$$

$$= -\frac{z_i}{z_s - z_i}$$ from the given condition

$z_i = 110, z_s = 200,$

$$\therefore \lambda = -\frac{110}{200-110} = -\frac{110}{90} = -\frac{11}{9}$$

Substitute $\lambda$ into the Ray Equation (3)

$$\overrightarrow{R} = \overrightarrow{P_i} + \lambda(\overrightarrow{P_s} - \overrightarrow{P_i})$$

$$= (100, 100, 110) + \lambda((-20, 110, 200) - (100, 100, 110))$$

$$= (100, 100, 110) - \frac{11}{9}(-130, 10, 90)$$

$$= (100 + \frac{11 \times 130}{9}, 100 - \frac{11 \times 10}{9}, 0)$$ On $X_w$-$Y_w$ plane $z$ must be 0!

https://github.com/hualili/CMPE240-Adv-Microprocessors/blob/master/2018F/2018F-118-13diffuseInterpolation20181127.cpp