

CmpE240
Sept. 7

10

Sept. 7.

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

2nd LPC1768 pin-to-pin
Compatible. (Mbed)

a. Step 1. MCUXpresso IDE
1768 Binary Code.

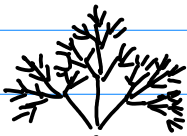
Step 2. "Firmware" Upload
the binary file to the
Flash. Need a prob

Step 3. Interactive Debugging.

3rd LPC1114 Digi-Key in Stock.
LPC1114

GPP/SPI, FLASH (ON-Chip)

1/8 of the size
Comparing to LPC1768/9



Homework (0 pt)

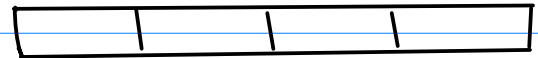
1. Form A Team By Wednesday.
4-Person

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

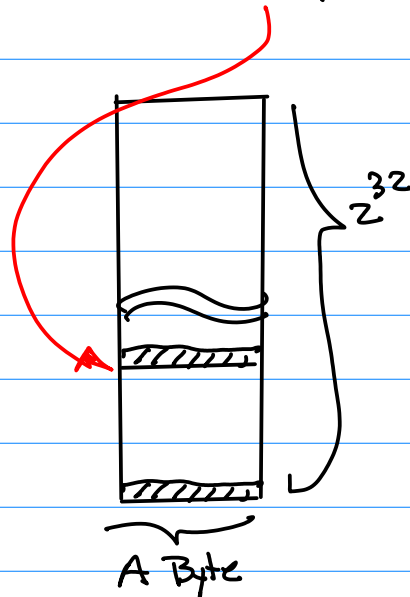
Example: Register File

} Special Purpose Registers
General Purpose Register

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



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



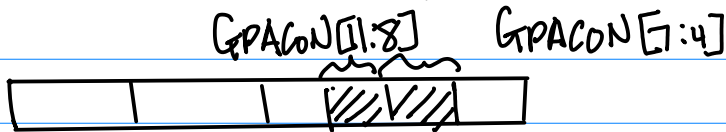
Note: The Task of Init & Config
Can be realized by using HLL
(High Level Language), C/C++,
to deposit A Binary Pattern to that
Memory Location (Addr. is a pointer)

QmpE240

Sept. 7

For Example for Samsung ARM-11.

Consider:
Power Up Address + Power Traces.
Booting.



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

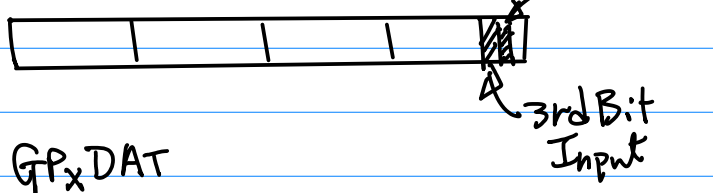
Design Requirements (Spec.)

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

To perform Init & Config.

GPACON[7:4] = 0001 = 0x1

GPACON[1:8] = 0000 = 0x0



GPACON[3:0] = 0x10

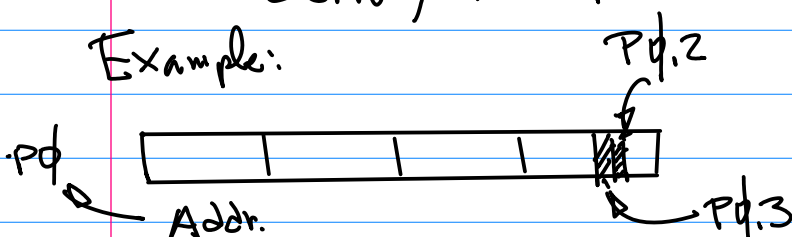
Sept. 12 (Monday)

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

Note: Target platform.

LC17ba, LC11C24

Example:



CMPE240
Sept. 12 (Monday)

12

Homework Due A week from this Wed.
Sample Code: On Github.

"Legs" Layout

Note: Connectivity Table

CPU Pin	J2	Note
GPP/P0.2	A-?	Output $V_{CC} = I \cdot R$
GPP/P0.3	B-?	Input
GND	A-?	GND.

Prototyping Board Option 1.

3.3V

1769

GPPx "1"

GPPy

R

LED

GND

SW

B

Equation: $V_{CC} = I \cdot R + V_{LED}$... (1)

Materials:

- ① LPC 1769 OR 1724
- ② Resistors. 250Ω ~ 1kΩ
- ③ LED.
- ④

A. Output { "1" ON, "0" OFF }

B. Input Testing { "1", "0" }

2° SCH IP/OP

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. →

"Semi-host"

3° Import LPC1769 patch.

1769 patch.zip

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

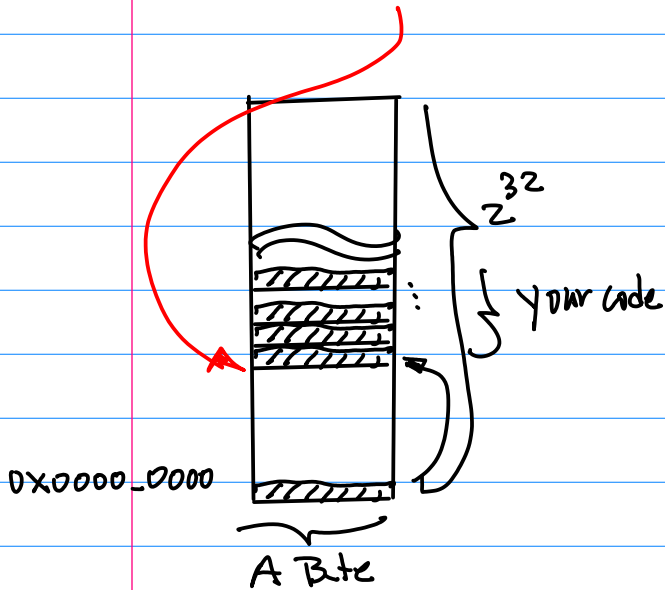
2° prob issue → fix:

Reconnect OR Reboot.

Import GPIO project to your MCU 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:

$$\vec{d}(x, y) \triangleq \vec{P}_{\text{end}}(x_{\text{end}}, y_{\text{end}}) - \vec{P}_i(x_i, y_i) \quad \dots (1)$$

Ending pt. Starting pt.

Definition 2: (Line Eqn. in Vector Form)

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

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

Consider G.E. Design.

Math. Formulation, for Vector Graphics.

Example.

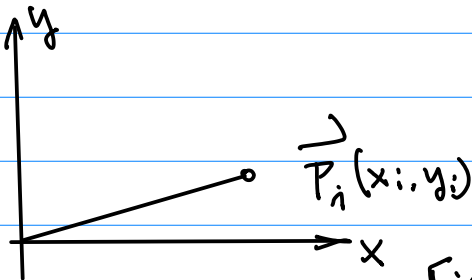


Fig. 1.

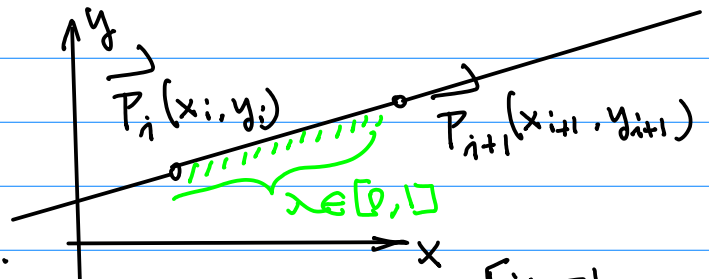


Fig. 2b.

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

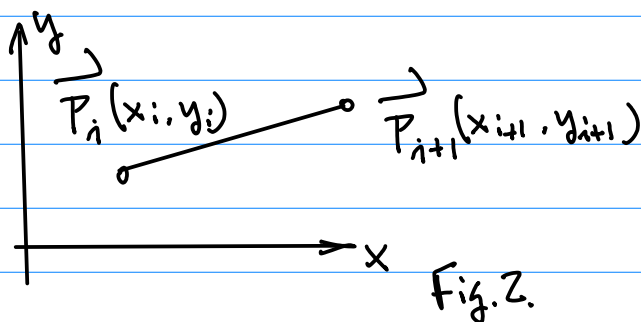


Fig. 2.

Observation 1:

When $\lambda = 0$, Eqn (2) 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 $\vec{P}_{i+1}(x_{i+1}, y_{i+1})$

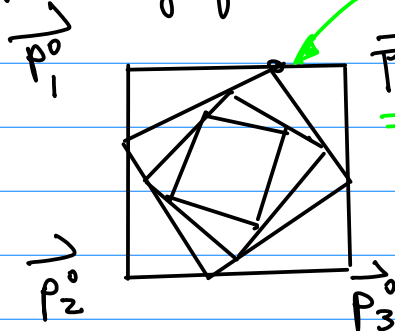
Sept. 21. Today's Topics:

1. Check Canvas for the Submission of the In-Class Exercise;
2. Graphics Lib has been ported to LPC1114, Ref. Code & PPT will be provided.

Example: Given the equation Below,

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

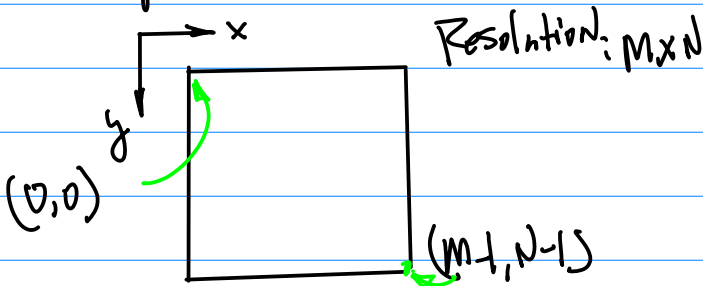
Let's Design A Graphic Algorithm to Create A Set of Counter Clockwise (CCW) Rotating Squares.



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 $m \times n$

$\max(m, n) \leq 200$, Select the Size of the Cube ≈ 50



$m \times n$
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 P15.

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

$$\begin{aligned} \vec{P}_1(x_1, y_1) &= \vec{P}_0(x_0, y_0) + \lambda (\vec{P}_1(x_1, y_1) - \vec{P}_0(x_0, y_0)) \\ &= (60, 60) + \lambda ((10, 60) - (60, 60)) \\ &= (60, 60) + \lambda (-50, 0) \end{aligned}$$

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++ for this purpose.

From Eq(1), we have

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

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

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

Consider Creating A Screen Saver
By Generating A tree.

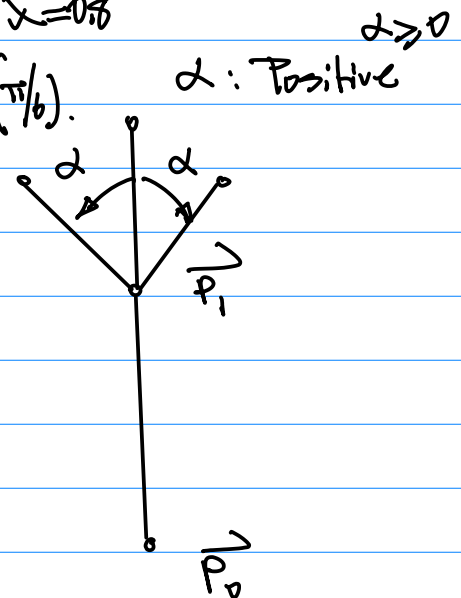
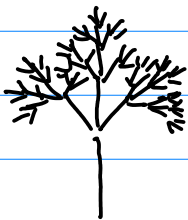
Note: Level 0: Tree Trunk; Same Direction, Directional Vekt. Same.

Level 1: Branch (Main): Mag. Reduction By 20% $\lambda = 0.8$

Side Branch: L (CCW), Rotation by α ($\pi/6$).
R (CW), $\alpha < 0$

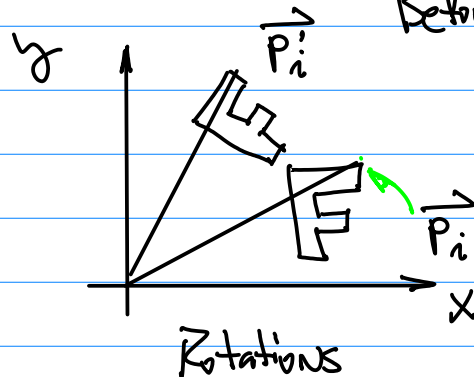
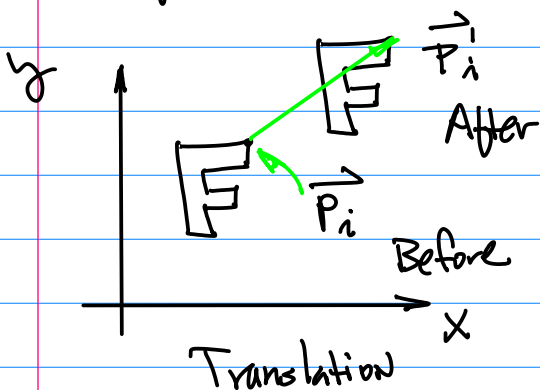
Level 2, 3, ..., k.

Repeat Level 1 with reference vector (pt)
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).

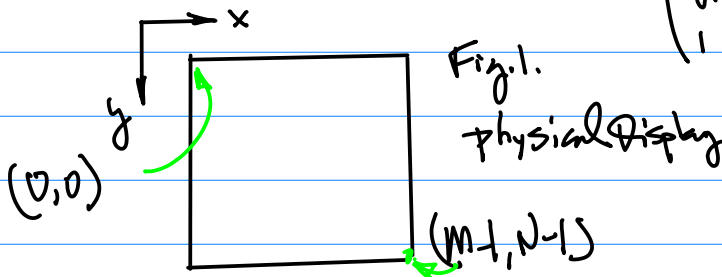
$$P_t(\text{After}) \quad P_t(\text{Before}) \vec{P}_i$$

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

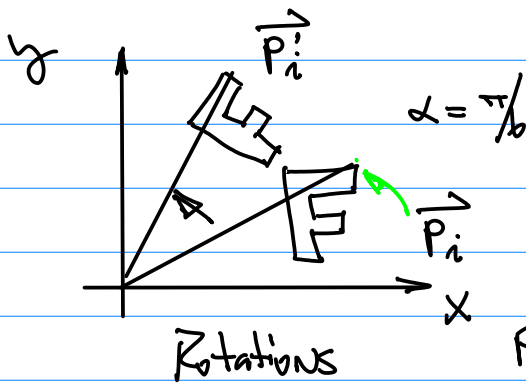
3D Vector, With One "Dummy" Dimension.

Rotation Matrix for Eqn (1b)

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



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



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

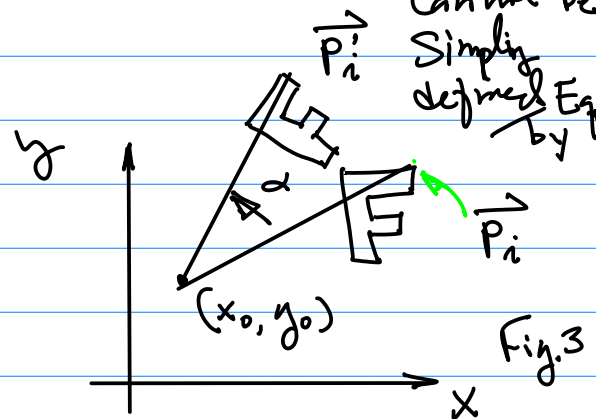
for the Reference of Doing C/C++ Coding.

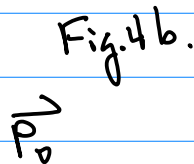
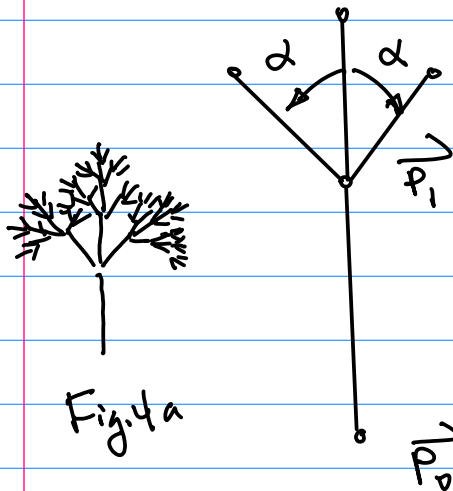
Note: Reference pt. for the Definition of Rotation. This Rotation Can not be Simply defined by Eq (2)

Example: for the Rotation illustrated in Fig. 2.

$$P_t(\text{After}) \quad P_t(\text{Before}) \vec{P}_i$$

$$\begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} \stackrel{?}{=} \phi \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \dots (1)$$





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.

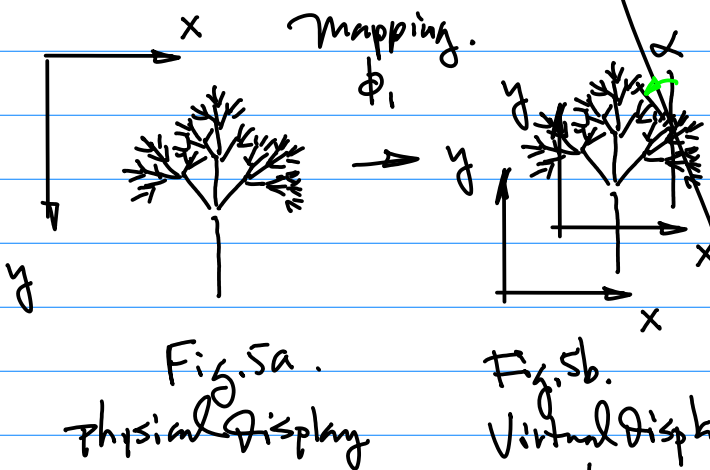


Fig. 5c. Rotation By α . CCW.

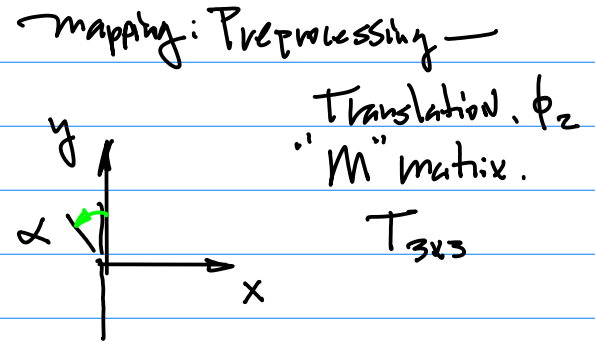
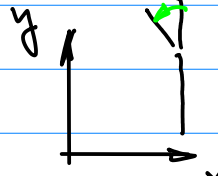
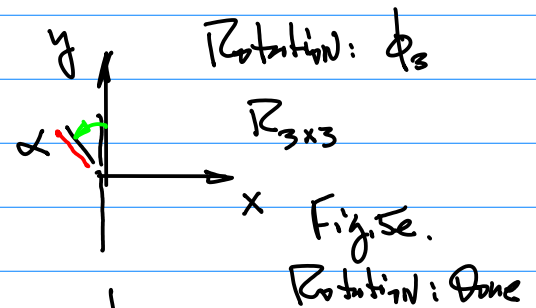
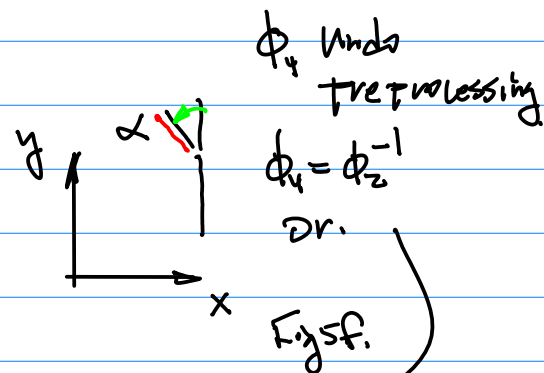


Fig. 5d. Preprocessing By Translation.

Rotation Eqn. (2).



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

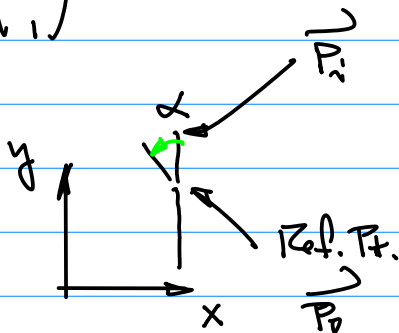


$$\phi_4 = \phi_2^{-1} = T_{3 \times 3}^{-1} \dots (3)$$

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

Start at given
pt. to be Rotated

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



Next, Preprocessing ϕ_2 , $T_{3 \times 3}$
To make $P_0(x_0, y_0)$ to overlap
with the origin $(0,0)$.

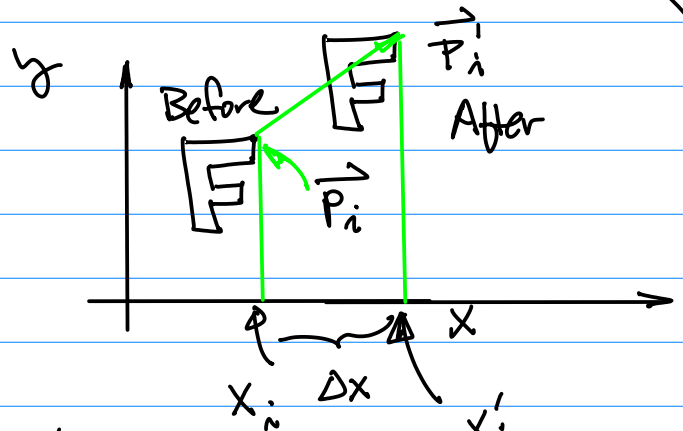
$$T_{3 \times 3} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \quad \dots (5)$$

$$\vec{P}'_i \text{ After} = T_{3 \times 3} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \text{ Before} \quad \dots (6)$$

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

$$x'_i = a_{11}x_i + a_{12}y_i + a_{13} \quad \dots (6a)$$

Where $a_{13} = \Delta x$



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

So $x'_i = x_i + 0 + \Delta x = x_i + \Delta x \quad \dots (6c)$

$$y'_i = y_i + \Delta y \quad \dots (6d)$$

Therefore

$$T_{3 \times 3} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \quad \dots (7)$$

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

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

Put All these together, we have

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

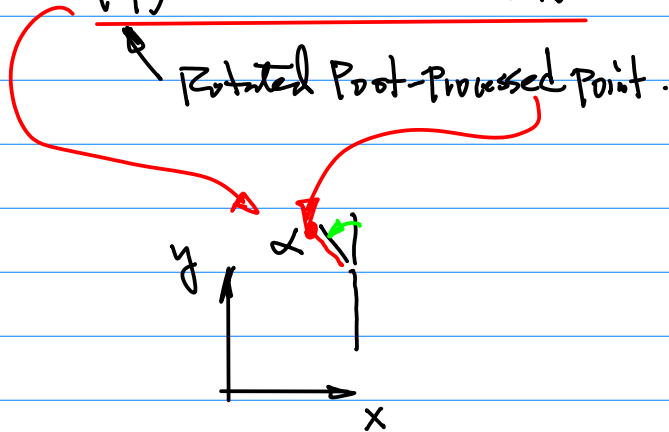


Fig. 7

Sept. 28 Wed.

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

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

T_{Σ} is 3×3 matrix

↓
Last Row of T_{Σ} (3×3) 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.

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

Similarly for y_L ,

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

Note: Work on
Virtual to physical
mapping

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

→ Lead

→ Enumeration of the
Points.

→ $P_{i,l}$ or $P_{i,r}$

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

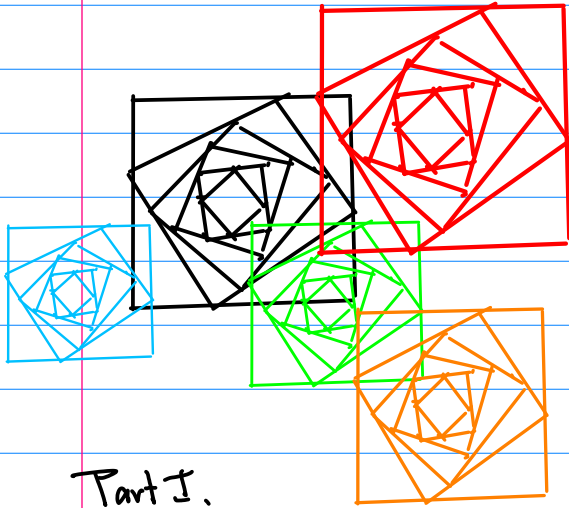
1st project, 2D Graphics Engine

2D Graphics Screen Saver
a collection of Rotating
Squares.

Impe 240
Sept. 28, 22

20

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



Part I.

Fig.1
(Level 10 or higher)



LPC1769
CPU

LCD

Part II. Trees. Optional for Artistic Presentation.

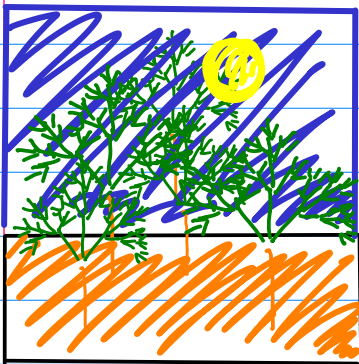


Fig.2

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

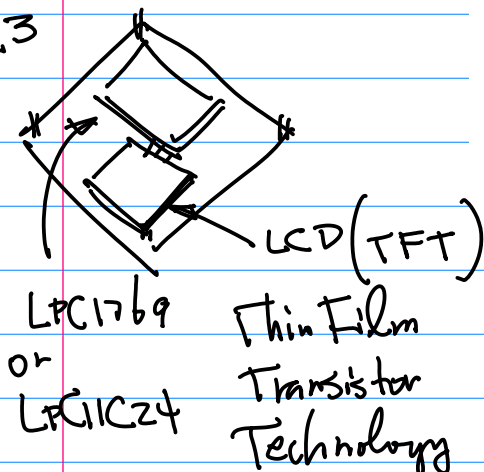
1. MOSI Output
2. MISO Input
3. SCK (Clock) Output
4. SS (CS) Output

Now, consider LPC1769, CPU Datasheet
7p431. CR0

Consider the Graphics Engine Design
from System View.

NXP Semiconductors

Fig.3



a. SSP
b. CR0, CR1

UM10360

Chapter 18: LPC176x/5x SSP0/1

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

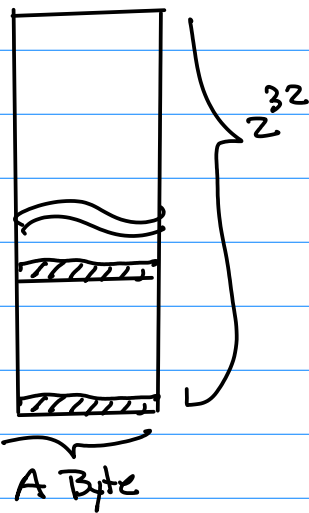
Bit	Symbol	Value	Description	Reset Value
3:0	DSS		Data Size Select. This field controls the number of bits transferred in each frame. Values 0000-0010 are not supported and should not be used.	0000
		0011	4-bit transfer	

(SSP0CR0)

CR0

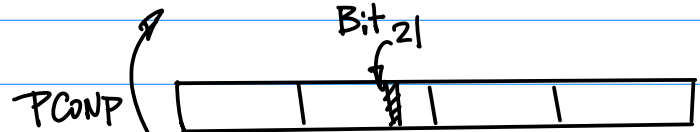


Addr. 0x4008-0000



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.



Note: 2nd STR: PCLKSEL

Oct.5 (Wed)

Example: SPR (for SSP/SPI)
CRP

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, 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
```

7P117.

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 a 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	Reserved
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
23:22	P0.11	GPIO Port 0.11	RXD2	SCL2	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	MOSI0	MOSI
7:6	P0.19	GPIO Port 0.19	DSR1	Reserved	SDA1
9:8	P0.20	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)

```

171
172 /* Set DSS data to 8-bit, Frame format SPI, CPOL = 0, CPHA = 0, and SCR is 15 */
173 LPC_SSP0->CR0 = 0x0707;
174

```

0000 0111 0000 0111
 SCR SPI 8 bit

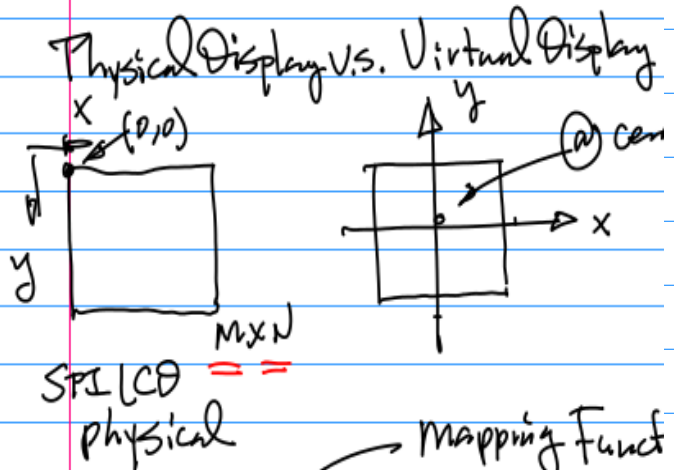
$$f_{SPI} = \frac{PCLK}{CPSDVSR * (SCR + 1)}$$

Consider Display Device Interface.

Background: Low Level Design of I/O
 Hardware Driver Design.
 Physical Display vs. Virtual Display.

Example: Physical Display vs. Virtual Display.

PP19. from ref.
 (github. Notes ~ 105 ~)



No. of
 M: Row
 N: Col.

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

CMPE240

Oct. 10, 22

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

$\left\{ \begin{array}{l} \text{Direction} \\ \text{Offset} \end{array} \right.$

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

(half of the No. of Rows)

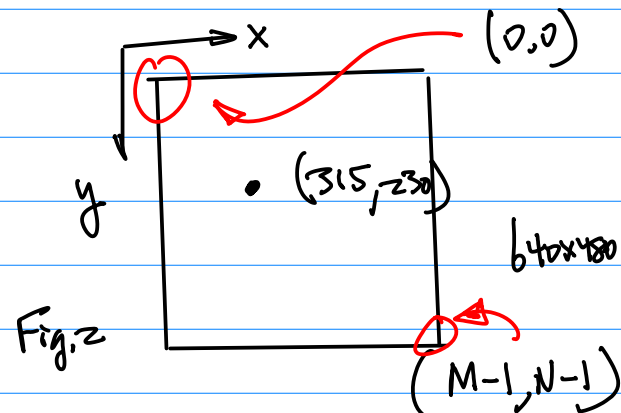
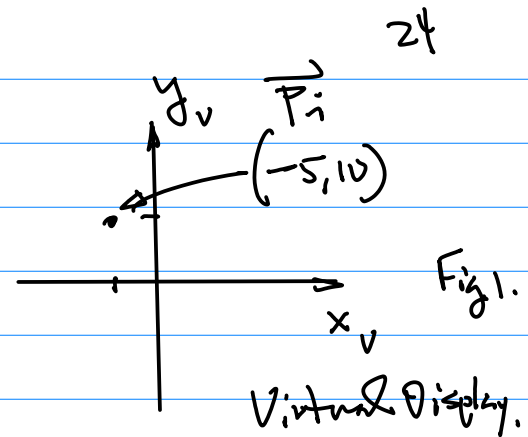
$$\left\{ \begin{array}{l} x_p = x_v + \frac{M}{2} \dots (7) \\ y_p = -y_v + \frac{N}{2} \dots (8) \end{array} \right.$$

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 $\vec{T}_i(x_i, y_i) = (-5, 10)$ in the virtual coordinate system.
Given: 1. Physical Display with Resolution $M \times N$ (640×480),
Find the New point $\vec{T}_i'(x_i', y_i')$
After the transformation from the Virtual Display to the physical Display.



Sol: From Eqn (7), we have

$$x_p = x_v + \frac{M}{2} \quad | \quad M=640$$

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

Where we have $x_v = -5$, so

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

$$= 315 ;$$

For y_p from Eqn (8).

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

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

$y_v = 10$, hence,

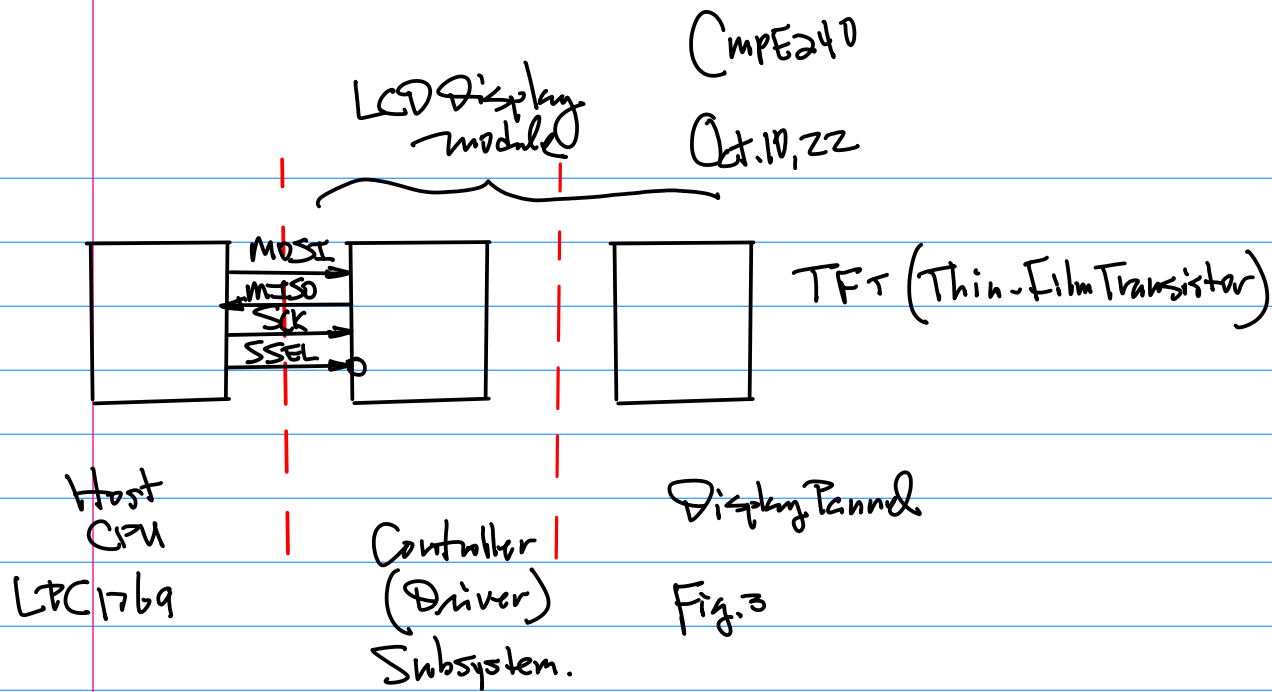
$$y_p = -10 + 240 = 230 //$$

Now, Consider the Design of Display Driver (Hardware)

CmPE240

Oct. 10, 22

25

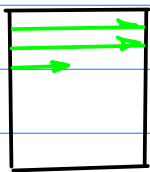


2nd LPC1769 For Example
(Not most Cost Effective Solution)

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

1. Sync. Signals

- (1) Frame Sync. Example: 30 F.P.S. $\text{Sync. } F \rightarrow f_F = 30 \text{ Hz} \dots (1)$
- (2) Horizontal Sync. for $M \times N$ Resolution, it Repeats N Times Per frame.
- (3) 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 $\text{Sync. } F$ & $\text{Sync. } H$, Denoted as f_H

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

$$\text{Sync. } H = N \cdot \text{Sync. } F \quad \dots (2b)$$

CMPE240
Oct. 10, 22

26

Consider Sync. v.s. Sync. H.

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

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

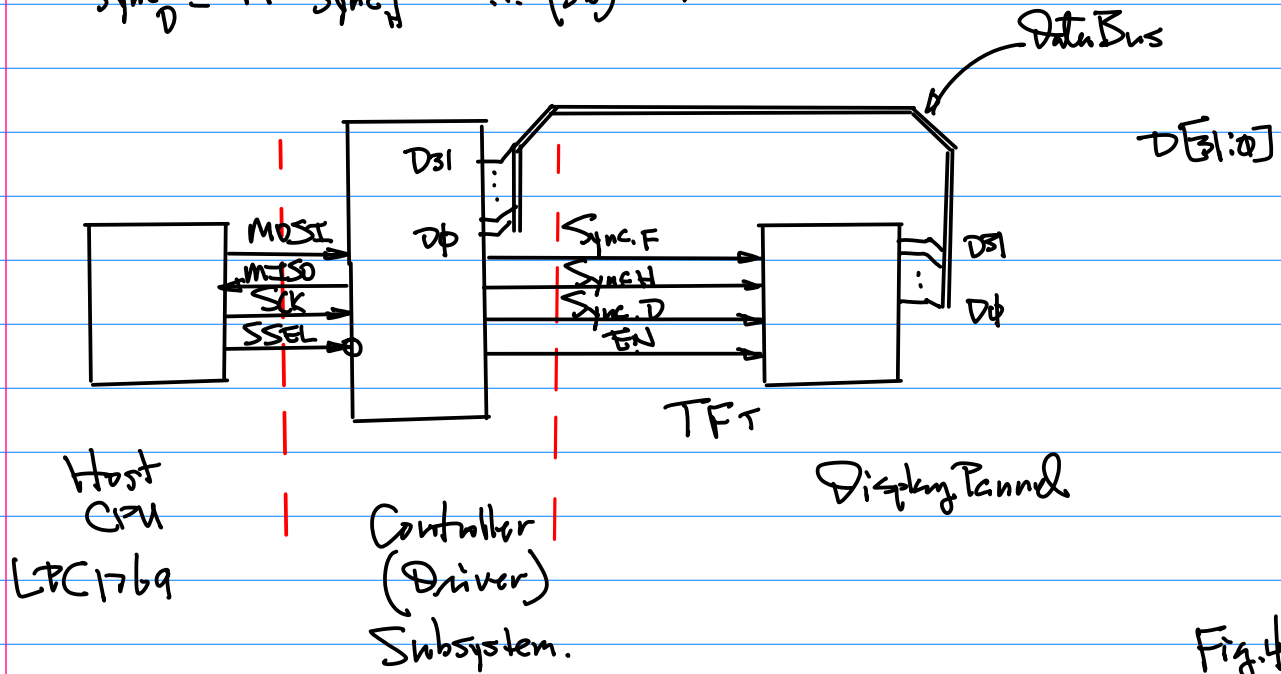
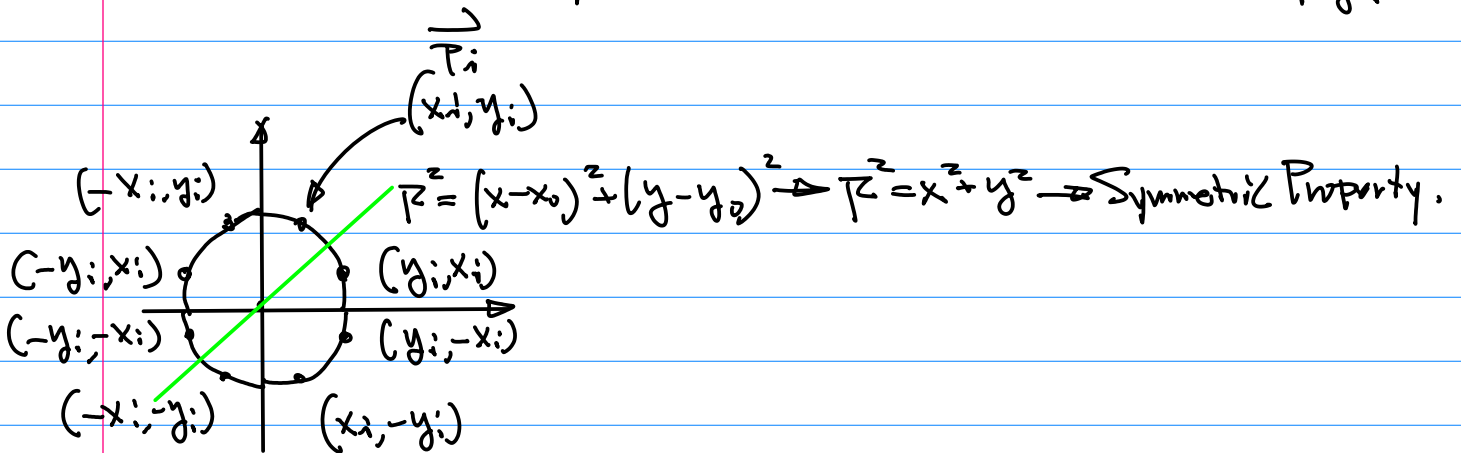
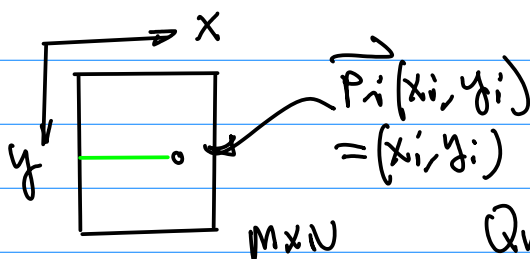


Fig. 4



Oct. 12 (Wed)

Example:



Assume FrameRate 30FPS.

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}

$$T_H = 1/\text{Sync}_H \quad \dots (1)$$

$$\text{And } T = (y_i - 1) T_H \Big|_{T_H = 1/\text{Sync}_H} \\ = (y_i - 1) / \text{Sync}_H \quad \dots (2)$$

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

$$T_D = 1/\text{Sync}_D \quad \dots (3)$$

$$\text{Sync}_D = M \text{Sync}_H$$

$$\text{So } T' = (x_i - 1) T_D = (x_i - 1) / M \text{Sync}_H \quad \dots (4)$$

Hence

$$T_\Sigma = T + T' \\ = (y_i - 1) / \text{Sync}_H + (x_i - 1) / (M \text{Sync}_H) \quad \dots (5)$$

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

$$y = ax + b$$

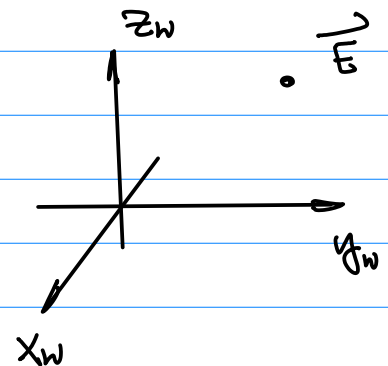
No. Syst Computations
 $R^2 = (x - x_0)^2 + (y - y_0)^2$

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

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

...note... May...

Fig. 1.



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

World Coordinate System.

RH: Right-Hand
System Defined
By Vector
Cross Product
 $\vec{x}_{w0} \times \vec{y}_{w0}$

Omprakash

Oct. 12, 22

28

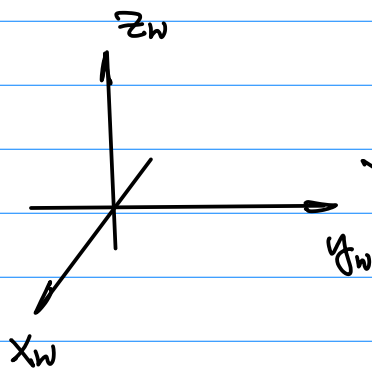


Fig. 2

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

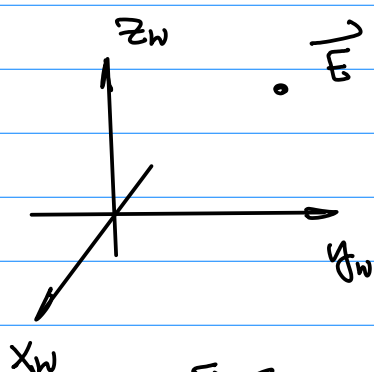
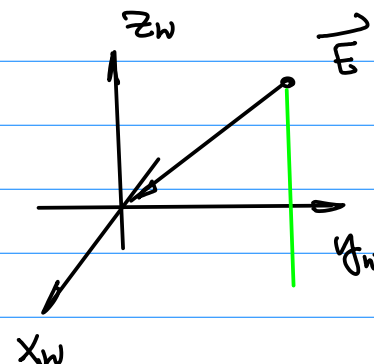


Fig. 3

"Pin-Hole" Camera for its simplicity. Its properties include:

1° An Enclosure, and a pin-hole, and projection plane. Fig. 6

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.

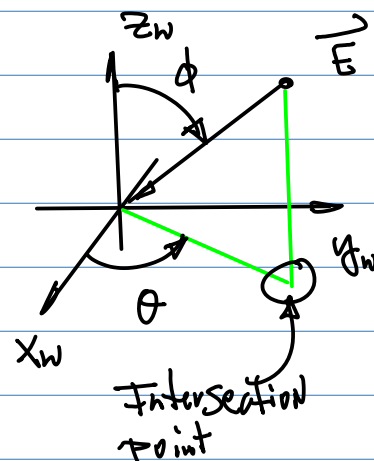
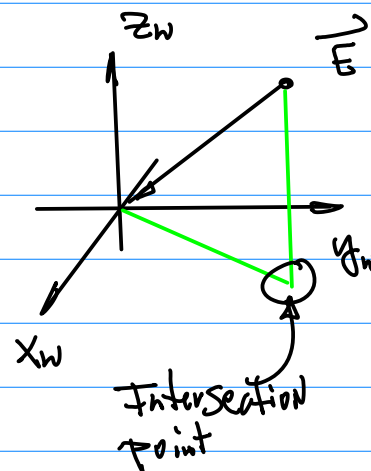


Fig. 7

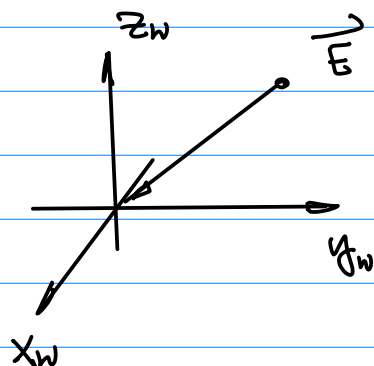
Angle θ : ON x_w - y_w plane C.C.W as a Positive Angle

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



CMPG240

Oct. 12, 22

29

Oct. 17 (Monday).

Example: Perspective Projection.

Projection of 3D objects onto 2D plane.

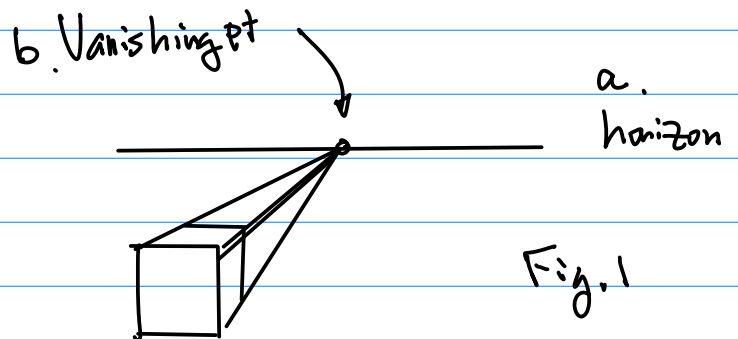


Fig. 1

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.

Fig. 8

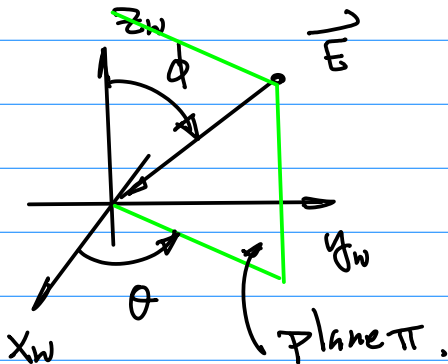
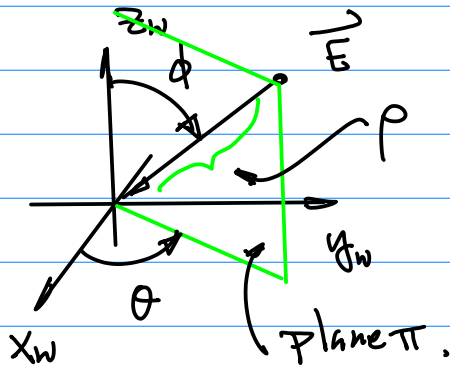


Fig. 9



Let's consider the Viewer Coordinate

$x_v - y_v - z_v$

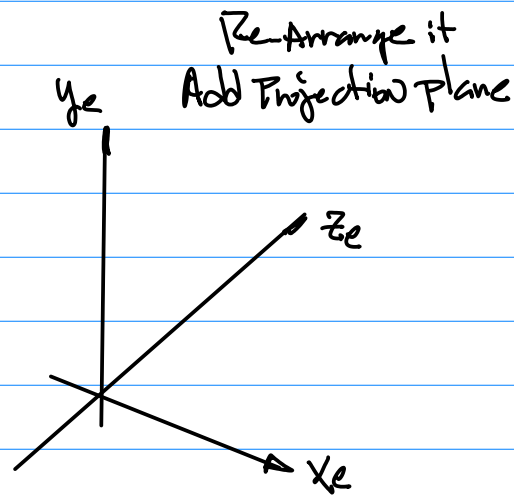
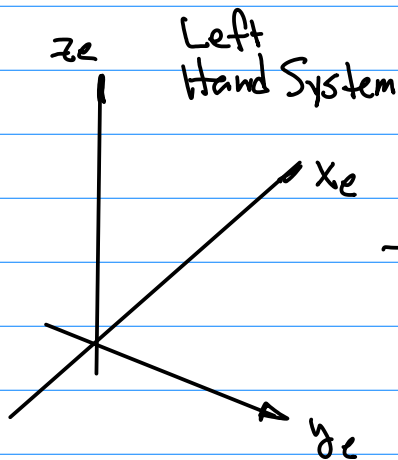
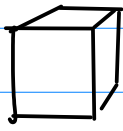


Fig. 10

OmPE240

Oct. 17, 22



Perspective
Fig. 2. Projection.

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

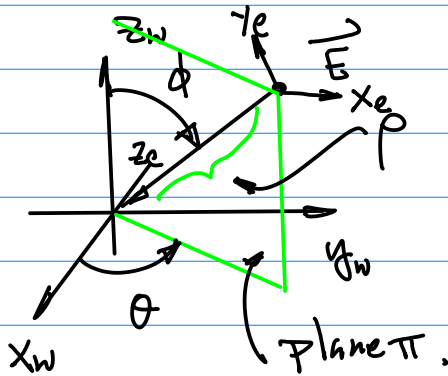
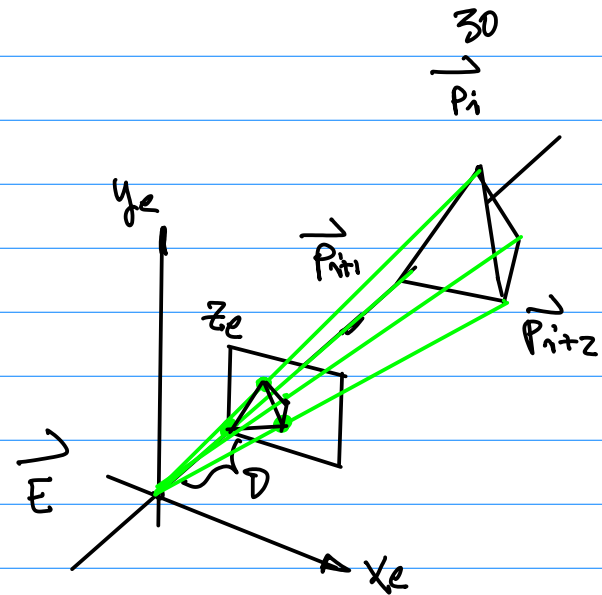


Fig. 3

Transformation pipeline $\left\{ \begin{array}{l} x_w - y_w - z_w \text{ to } x_v - y_v - z_v \\ \text{World} \rightarrow \text{Viewer} \\ \text{Perspective Projection} \end{array} \right.$

$$x_w - y_w - z_w \rightarrow x_v - y_v - z_v \rightarrow P.P.$$

$$\vec{P}_i(x_i, y_i, z_i) \rightarrow \vec{P}'_i(x'_i, y'_i, z'_i) \rightarrow \vec{P}''_i(x''_i, y''_i, D) \text{ In Viewer}$$

Drop D in z
so, 2D with
"Depth".

