Sept. 7.

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

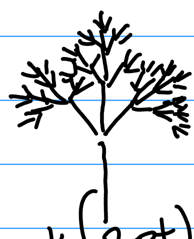
2° LPC1768 pin-to-pin Compatable. (Mbed)

a. Step1. MCUXpresso IDE
1768 Binary Code.

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

Step3. Interactive Debugging.

3° LPC11C24 Digi-Key in Stock.
LPC1114
Size
GPP/SPI, FLASH (ON-Chip)
1/8 of the size
Comparing to LPC1768/9



Homework (0 pt)

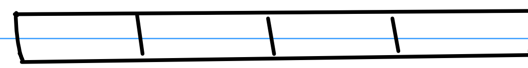
1. Form A Team By Wednesday.
4-Person

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

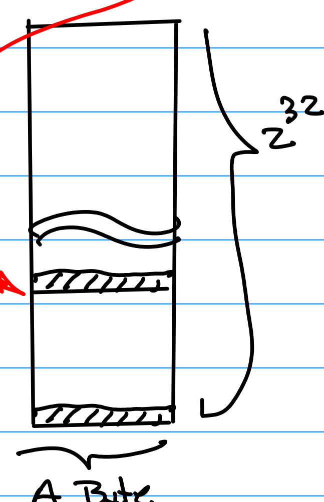
Example: Register File

- Special Purpose Registers
- General Purpose Register

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



GPxCON 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

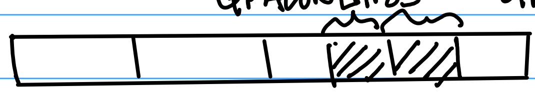
Sept. 7

11

Consider:
Power up Address + Power Process.
Booting.

For Example for Samsung ARM-11.

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



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

Design Requirements (Spec.)

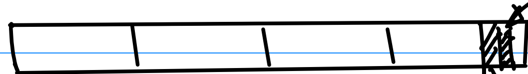
1. 2nd Bit As An Output
2. 3rd Bit As An Input

To perform Init & Config.

GPACON [7:4] = 0001 = 0X1

GPACON [11:8] = 0000 = 0X0

2nd Bit
Output



3rd Bit
Input

GPxDAT

GPACON [31:0] = 0X10

Sept. 12 (monday)

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

Note: Target platform.
LPC1769, LPC11C24

Example:

P0.2

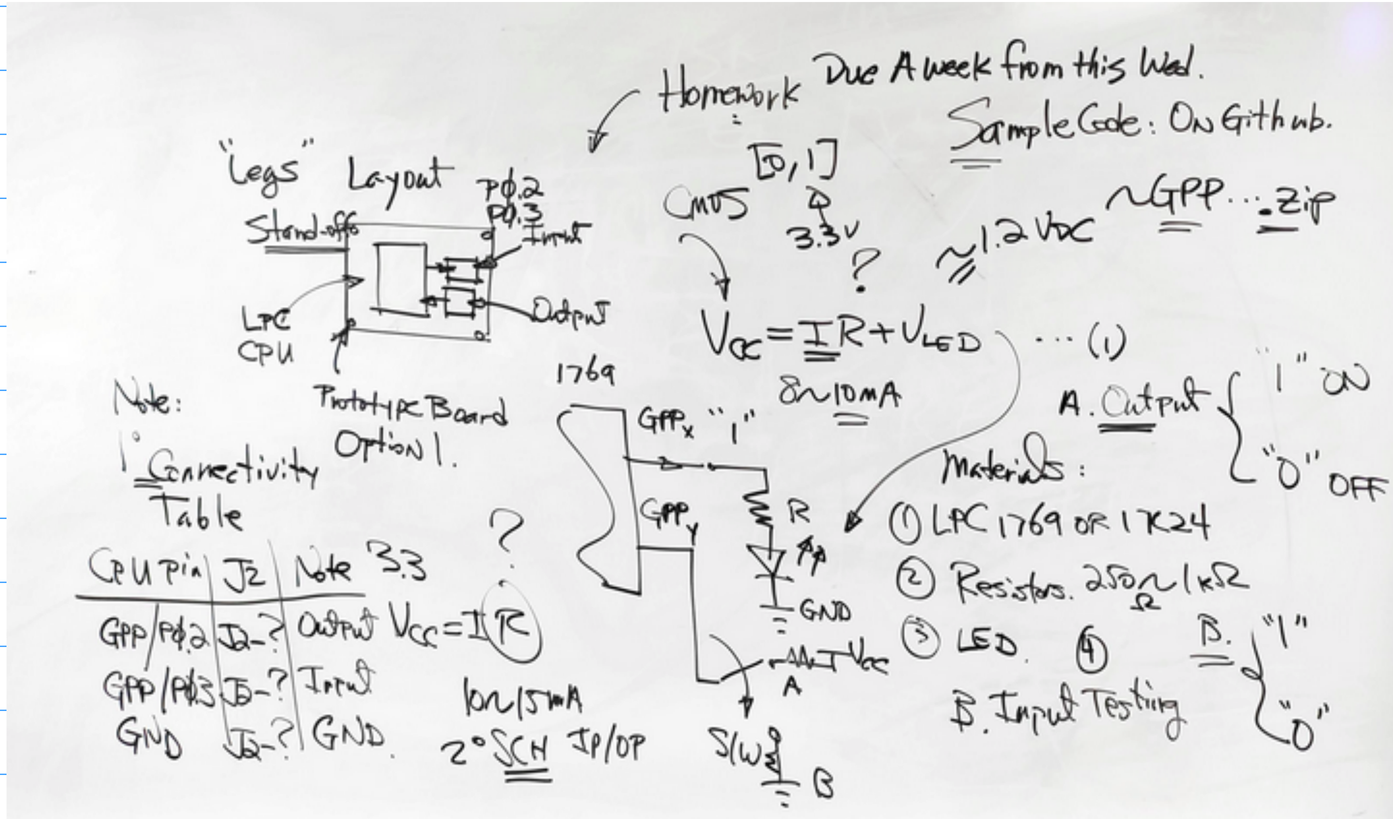
P0

Addr.

P0.3

CmpE240

Sept. 12 (monday) 12




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: Setup 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.

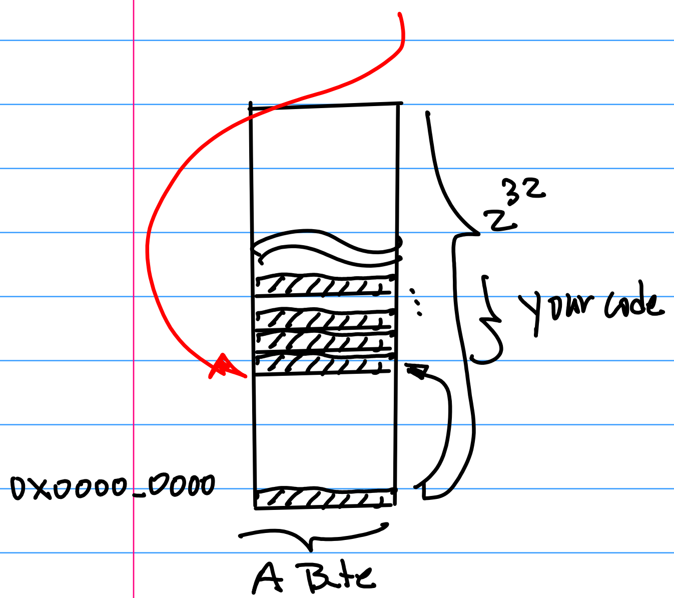
1769 patch.zip

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.

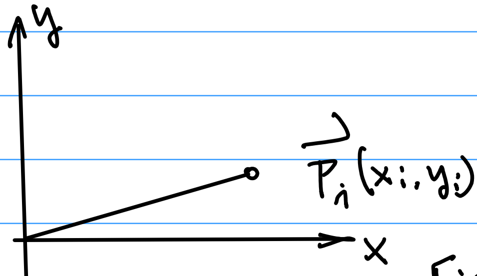



Fig.1.

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

$\downarrow$

$(x_i, y_i)$

Also, a line

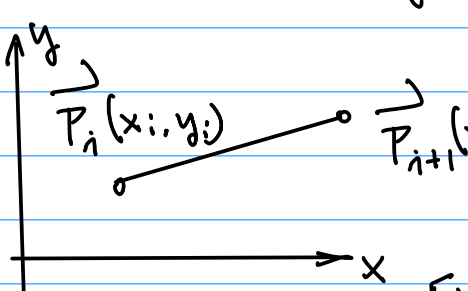



Fig.2.

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

Definition 1:

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

Definition 2: (Line Eqn. in Vector Form)

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

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

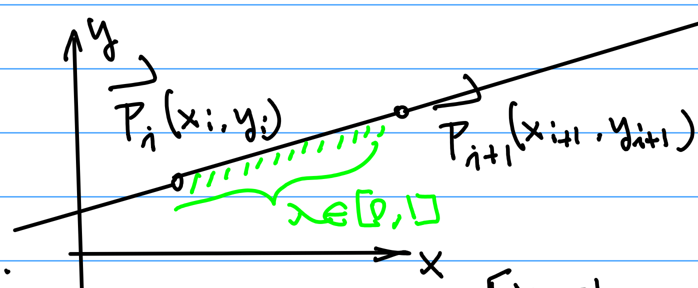



Fig. 2b.

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.

$M \times N$

No. of Pixels per Row (col). No. of Rows

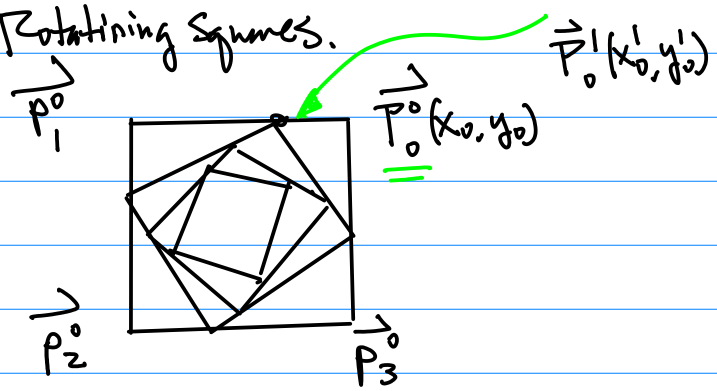
2021S-105-CMPE240-2021-03-22-Note.pdf

$\vec{P}_0^0(x_0^0, y_0^0), \vec{P}_1^0(x_1^0, y_1^0), \dots, \vec{P}_3^0(x_3^0, y_3^0)$ are defined as the Same data pts on pp15.

Example: Given the equation Below,

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

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




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

$$\vec{P}_0^1(x_0^1, y_0^1) = \vec{P}_0^0(x_0^0, y_0^0) + \lambda\left(\vec{P}_1^0(x_1^0, y_1^0) - \vec{P}_0(x_0^0, y_0^0)\right)$$

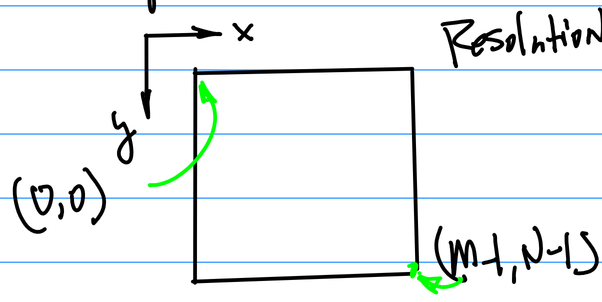
$$= (60, 60) + \lambda\left((10, 60) - (60, 60)\right)$$

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

Step 1. Define the initial set of Data Points $\{\vec{P}_i(x_i, y_i) \mid i = 0, 1, \dots, 3\}$ for Resolution of A Display Device defined as $M \times N$

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

Resolution: $M \times N$




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

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

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

Consider Creating A Screen Saver
By Generating A tree.

Note: Level 0: Tree Trunk;

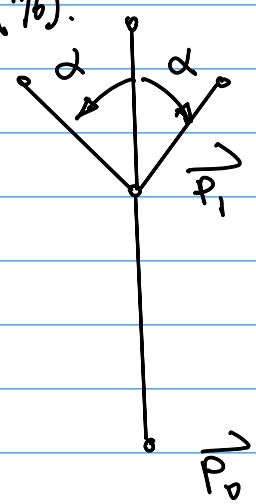
Level 1: Branch (Main): Same Direction, Directional Vect. Same. Mag. Reduction By 20% $\lambda = 0.8$

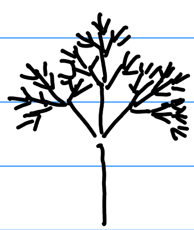
SideBranch: L (CCW), Rotation by $\alpha$ ($\pi/6$). $\alpha$ : Positive $\alpha \geq 0$

R (CW), $\alpha < 0$

level 2, 3, ..., k.

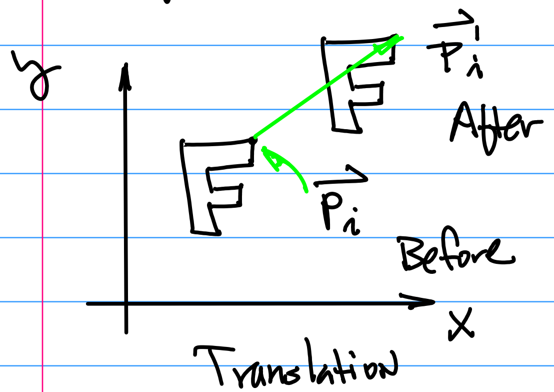
Repeat Level 1 with Reference Vector ($\vec{p_t}$) Updated Accordingly.

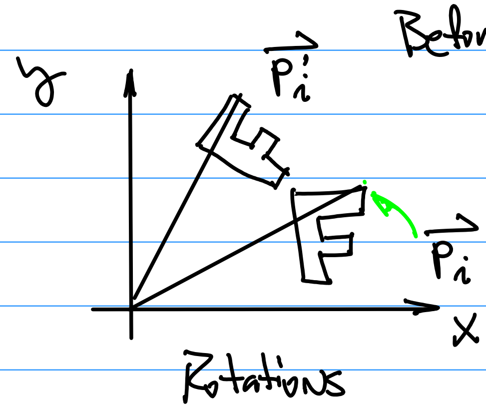


Background (2D Transformations)

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).

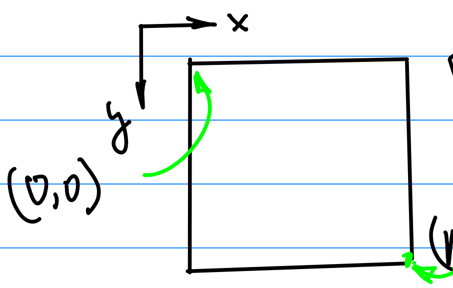
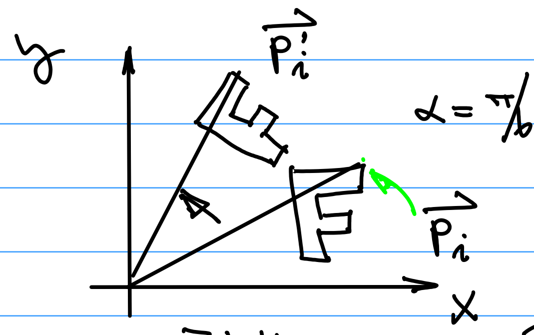



Fig.1. physical Display




Rotations Fig.2 Virtual Display.

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

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & \cdots & a_{23} \\ a_{31} & \cdots & a_{33} \end{pmatrix}_{3\times3} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \cdots (1b)$$

3D Vector, With One "Dummy" Dimension.

Rotation Matrix for Eqn (1b)

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

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

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

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

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.

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

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

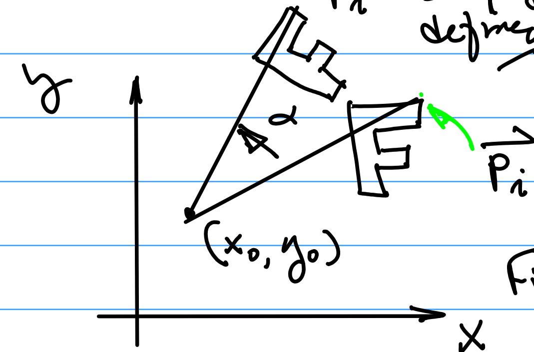



Fig.3

CmpE240

Sept.26.22



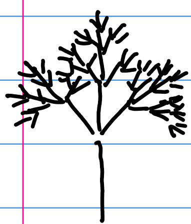


Fig.4a

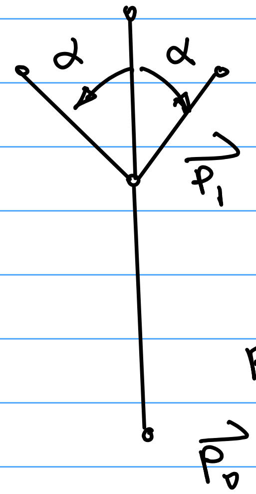



Fig.4b.

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.

Mapping. $\phi_1$

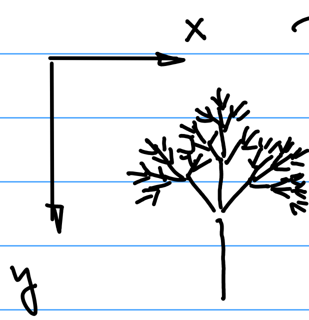



Fig.5a. Physical Display

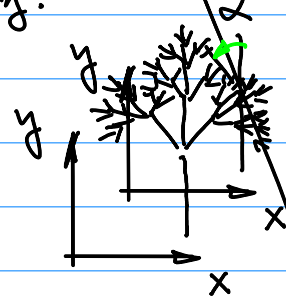



Fig.5b. Virtual Display

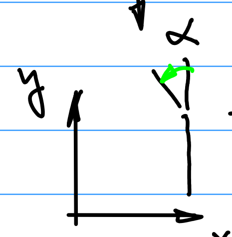



Fig.5c. Rotation By α. CCW.

Mapping: Preprocessing —

Translation. $\phi_2$

"M" matrix.

$T_{3\times3}$

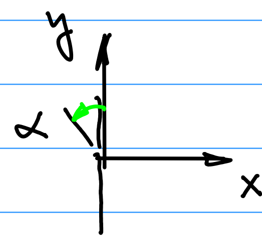



Fig5.d. Preprocessing By Translation.

Rotation Eqn.(2).

Rotation: $\phi_3$

$R_{3\times3}$

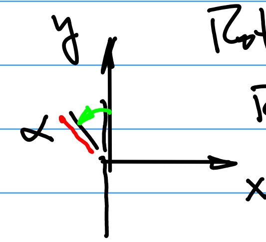



Fig.5e.

Rotation: Done

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

$\phi_4$ Undo Preprocessing

$\phi_4 = \phi_2^{-1}$

or.

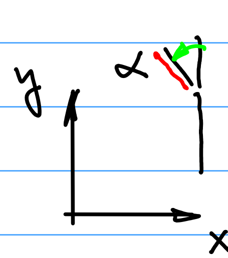


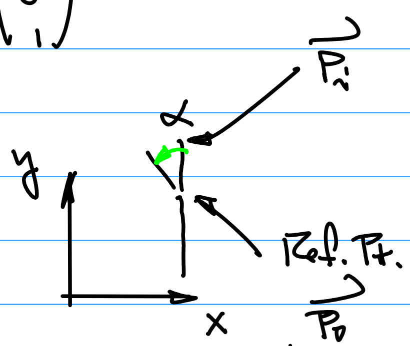

Fig5f.

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

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 $(0,0)$.

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

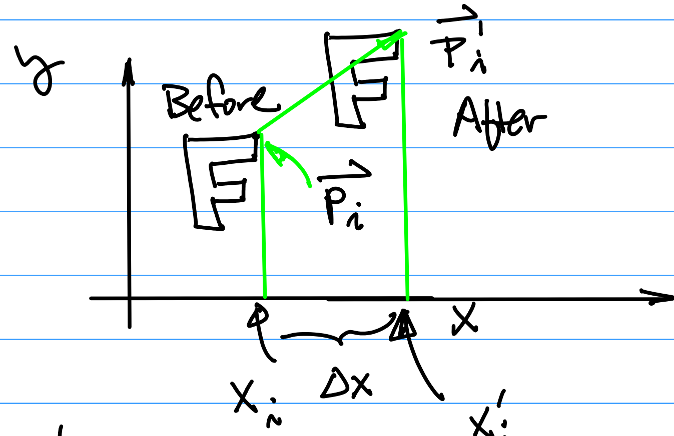
$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = T_{3\times3} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \ldots (6)$$

$P_i'$ 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. $\Phi_4$

$$M_{\Phi_3} = T_{3\times3}^{-1} = \begin{pmatrix} 1 & 0 & -\Delta X \\ 0 & 1 & -\Delta Y \\ 0 & 0 & 1 \end{pmatrix} \quad \ldots (8)$$

Put All these together, we have

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

Rotated Post-Processed Point.

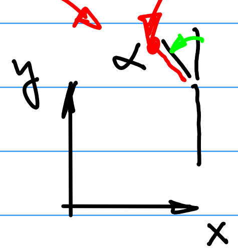



Fig. 7