

CMPE240
Spring 2023

v/

Jan 25, 23.

Today's Topics:

1° Syllabus, "Greensheet" of the Class.

San José State University
College of Engineering/Computer Engineering
Department
CMPE240 Advanced Microcomputer Design

S2023

Professor Hua Harry Li
Engineering Building, Rm 267A
Phone: (650) 400-1116 for Text Message Only
Email hua.li@sjsu.edu

Class Time: Mondays and Wednesdays 1:30-2:45 PM

Office Hours: Mondays and Wednesdays 4:30 – 5:30 PM Zoom

Zoom link: Join Zoom Meeting [https://us04web.zoom.us/j/9841607683?](https://us04web.zoom.us/j/9841607683?pwd=UIA3aEkITnV4bjNLQk5CQkw0dDk4UT09)

pwd=UIA3aEkITnV4bjNLQk5CQkw0dDk4UT09 Meeting ID: 984 160 7683 Passcode: 121092

Lecture Room: Engineering Building Room 303

Lab facility: Engineering Building Room 268

Prerequisites

5°

Good Throughout the entire School Session.
But it will expire by the Last Day of the
Class. (May 15th).

Note: 1° Attendance; 2° Bring your Laptop Computer; 3° Prototype Board in Class Use/Inspection.

Prototype System { Board A: CPU module.

Board B:
For Graphics Display

4° Board A: CPU NXP LPC11C24 to Replace LPC1769 (Near the End of Its Life, By 2024)

6° Lab Access Code / Policy for the Lab Usage.

Engineering Building Room 268

Prerequisites

CmpE 180D for non CMPE or non EE undergraduate major. Students who do not provide documentation of satisfied the class prerequisite requirements by the second class meeting will be dropped from the class.

Faculty Web Page and MYSJSU Messaging (Optional)

Copies of the course materials such as the syllabus, major assignment handouts, etc. can be found from github <https://github.com/hualili/CMPE240-Adv-Microprocessors/tree/master/2018F> and on SJSU CANVAS.

Note: Homeworks / Projects Announcement will be posted on CANVAS. Submission on CANVAS only.

Course Description

Architecture of a computing system including system bus, memory subsystems and peripherals. Uni-directional and bidirectional bus architectures, SRAM and FLASH memories and their interfaces with the system bus. Design of Graphics Processing Engines, interrupt controller, transmitter, timers, display adapter, and other system peripherals and bus interfaces.

Required Texts/Readings

Textbook

- NXP LPC17xx datasheets;
- LPC1768/1769 CPU Module schematics;
- Dave Jaggar, ARM Architectural Reference Manual, Prentice Hall, ISBN 0-13-736299-4;
- Reference: ARM11 data sheets and on-line web materials on line <https://github.com/hualili/>, or at the SJSU CANVAS provided copyright permitted;
- (Optional) Nvidia Jetson NANO datasheet and user menu (online from Nvidia developer website);
- (Optional) RISC-V tutorial (the link to be given in the lecture) and FPGA verilog implementation guide (the link to be given in the lecture).

Note: Find the Datasheet on the Class github. CMPE244

Other Readings

- The reference material for ARM CPU hardware features, application notes, class handouts and lab assignments and reports, please see Professor Li's lecture notes, PPT, sample C code etc on line <https://github.com/hualili/CMPE240-Adv-Microprocessors> ;
- Professor Li's book materials, ARM Microprocessor Systems (in preparation for publication) <https://github.com/hualili/CMPE240-Adv-Microprocessors>

Other equipment / material requirements

32Bit RISC Prototype/Development Board.

Ref:

1º CPU Datasheets

2021F-107-lpc-cpu-UM... Add files vi

2º "SCH" Design.

2021F-107b-sch-LPC... Add file

3º Lecture Notes

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

deadlines and penalties for adding and dropping classes

Homework/Projects

Assignments and Grading Policy

Laboratory	30%
Midterm Examination	30%
Final	40%

0 to 59 F
60 to 69 D
70 to 79 C
80 to 89 B
90 to 100 A

Option 1. Target CPU Module Board

NXP LPC 11C24 ARM CPU Module (recommended as this course), NXP LPC1769 ARM CPU Mc

Jan 29 (Monday).

1. Check the CANVAS for Homework, Honesty Pledge

2. Target platform.

Background: X86, MIPS, ARM.
CISC RISC
RISC-V

NXP LPC Family.

LPC1769 — End of Life
By 2024.

LPC1768 — mbed

LPC11C24

Option 1: Jetson NAND Zglo.

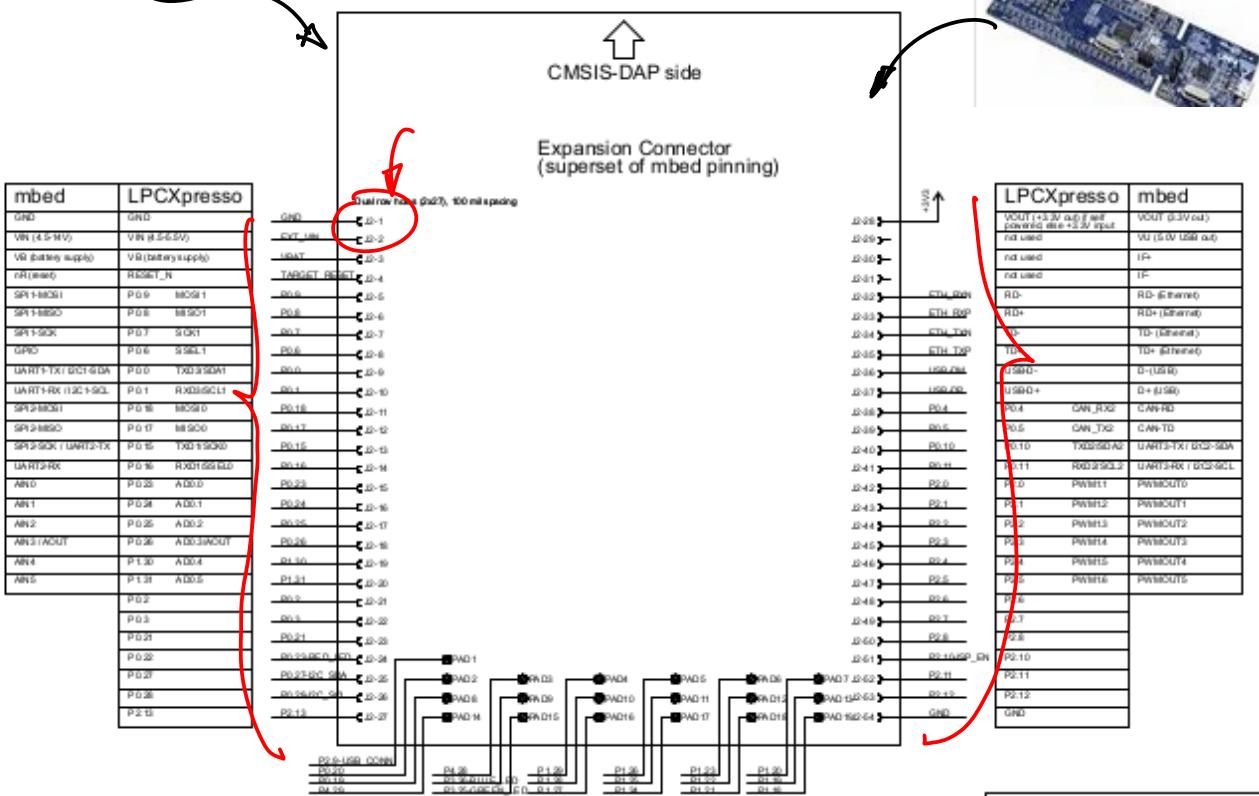
Option 2: RISC-V FPGA Board

BOM (Bill of Material) on github.

Anchor Electronics.

Example: SCH for LPC1768

Note: CPU module, see Fig.1, its Two Connectors match the Pins in the Schematic



Homework: Due A week from today.

1. Download NXP MCU Xpresso,
 2. Install MCU Xpresso;
 3. Start MCU Xpresso, then Screen
Capture of your MCU Xpresso Start
Page, make sure it has your
Personal identifier on it.

Feb 1st (Wed)

Note: 1^o Attendance Sheet
2^o Ref. from the Class github.

 2023S-102-MCUXpresso_IDE_Installation_...

3^o LPC module to be finalized today by the Class. Purchasing CPU module by the end of the day today.

functions. → Init & Config.

More than one part.

2. Connection to GPU Datasheet.

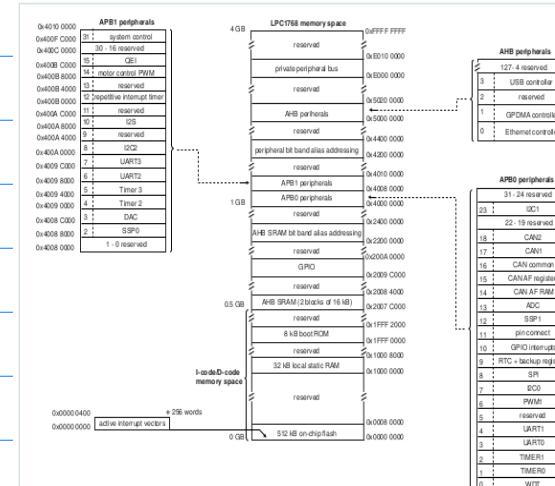


UM10360

LPC176x/5x User ma

User manual

PP.14 Memory Map.



P1Z.

LPC11C24

CPU module Board B

for Graphics
Engine Design
Emulation .
pre-fab

Example: Continuation of the SCH .

1. $P\psi, q \rightarrow "P"$ Port, General

Purpose part \rightarrow ARM multiplexing
Each pin can have more than one

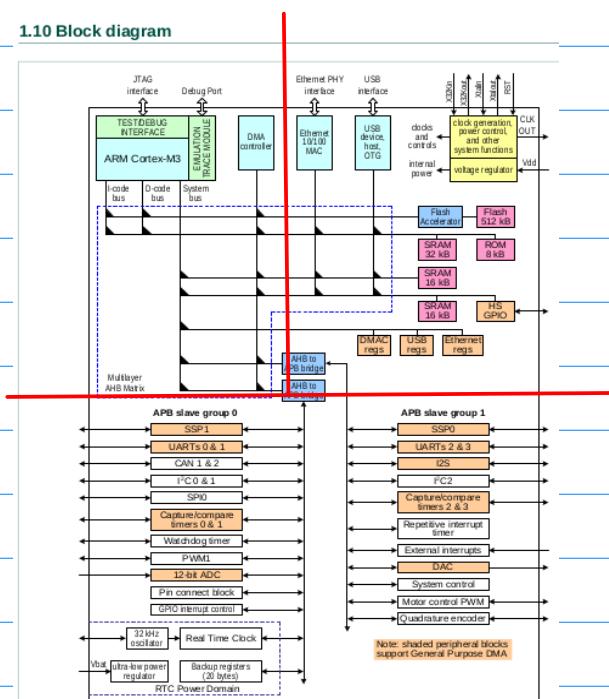
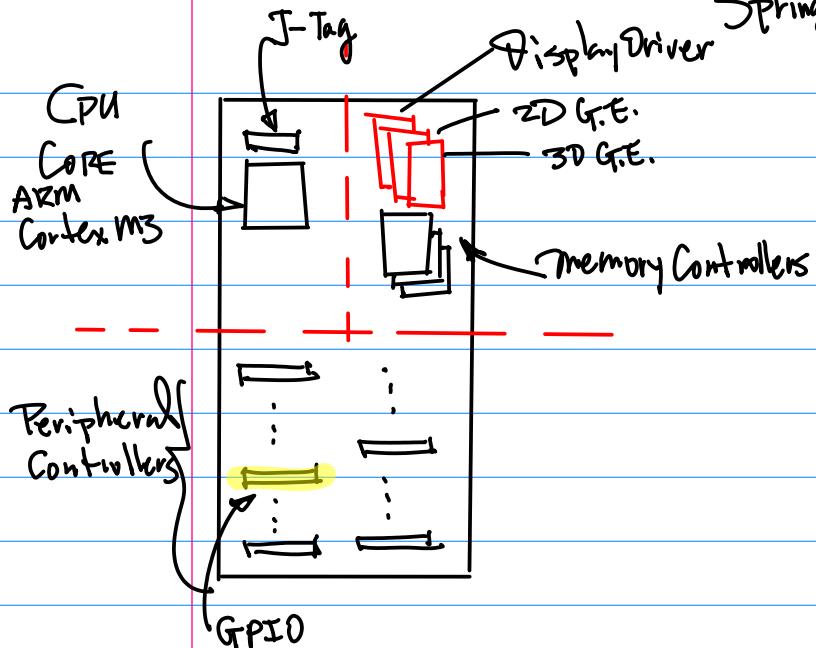


Fig 2. LPC1768 block diagram, CPU and buses



3° 3D Graphics Processing Engine. I.T.U.
 4° Video Codec. & Encoder MPEG4/H.264
 Decoder H.265
 Pixel Graphics.

Vector Graphics. (Display Drive, 2D G.E.,
3D G.E.)

5° General Discussion of the GPU Architecture

(i) 32 Bit RISC Architecture.

All. Bit width. 32bit 32bits
 Register File { Special Purpose Registers,
General Purpose Registers.
Any meaningful AL OPS. }

Init & Config. ✓
32bit

Bus Systems { Address ~ 32bit / Uni-Directional
Data ~ 32bit / Bi-directional.
Control Bus

(2) memory Map. $2^{32} = 2^{10} \cdot 2^{10} \cdot 2^{10} \cdot 2^1$
 | : :
 1K) :
 1Meg) :
 1TByte)

4 G.? Byte

Byte Addressable machine. ~ minimum mem.
 Cell with an unique addr. is a single byte.

Discussion On the CPU Block Diagram.

1° Display Driver.

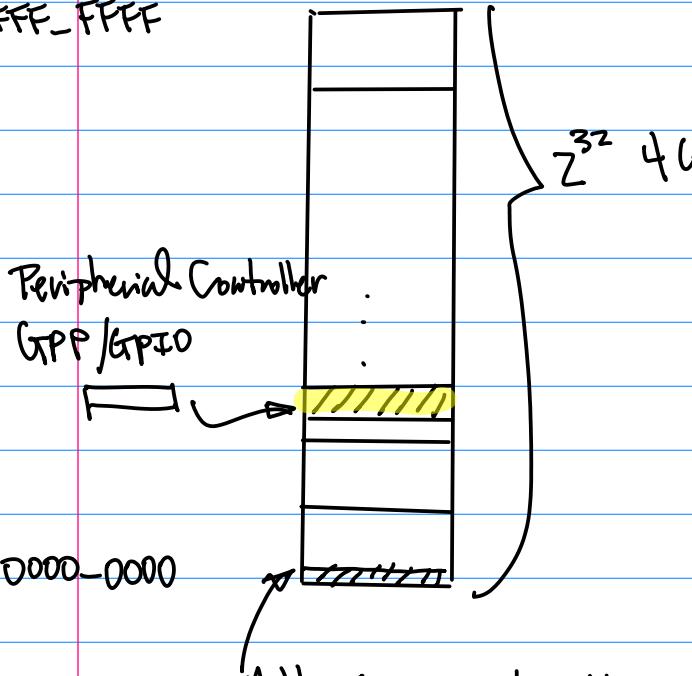
2° 2D Graphics Processing Engine

Example: Continuation of CPU
Architecture

Ref. GPU Datasheet pp.13,

pp.12.

0xFFFF_FFFF



a. Power-up Address: ~ when the CPU is powered up, it will go to this memory location to fetch the 1st Executable instruction

b. 8 Memory Banks. Equal size of memory blocks.

$$2^{32}/8 = 2^{32}/2^3 = 2^{29} = 2^9 \cdot 2^{20}$$

512 |
 Meg.

1st BANK is enumerated as BANK0, ..., the last BANK is Bank7.

c. We use 3 Bits from the Address Bus to define the Starting Addr. of Each Bank.

"Little
Endian"

$\alpha_3 \alpha_2 \dots \alpha_1 \alpha_0$

$\alpha_3 \alpha_2 \dots \alpha_1 \alpha_0$

$2^{32} = 4 \text{ GB}$	$\alpha_3 \alpha_2 \dots \alpha_1 \alpha_0$...
	0 0 0 . 0	BANK0 0x0000_0000
	0 0 1 . 0	BANK1 0x2000_0000
	0 1 0 . 0	BANK2 0x4000_0000
	:	:
	1 1 1 . 0	:

d.

UM10360

Chapter 2: LPC176x/5x Memory map
Rev. 3.1 — 2 April 2014

User manu

2.1 Memory map and peripheral addressing

The ARM Cortex-M3 processor has a single 4 GB address space. The following table shows how this space is used on the LPC176x/5x.

Table 3. LPC176x/5x memory usage and details

Address range	General Use	Address range details and description
0x0000_0000 - 0x0000_0FFF	On-chip non-volatile	0x0000_0000 - 0x0000_0FFF For devices with 512 kB of flash memory

Address for GPIO Controller. 0xZ---
 0x4---

Feb 8. (Wed).

Note: Inspection of CPU module.

1° LPC1769 or LTC11C24;

2° Layout Design. CPU module
Location
Standoffs.

Prototyping wire

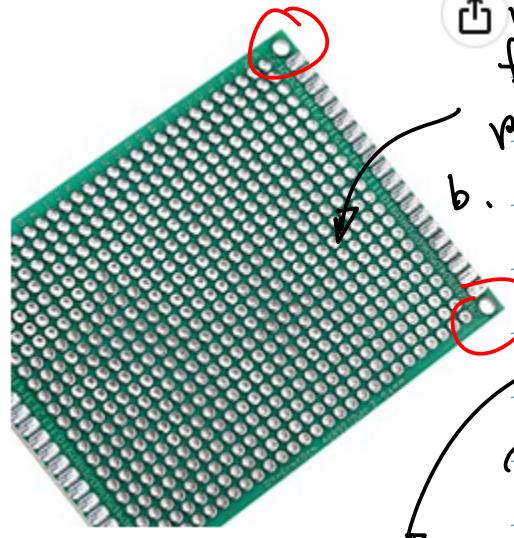
30AWG Blue

28n32 AWG

\$7.50



Prototype Board, Note:



- a. Through-Hole
is uniformly formed with plate coating
- b. Mounting holes at corners.
- c. Standoffs.

~ \$5-\$15



Exquisite Workmanship
Accurate dimensions and production methods ensure the perfect use of each component

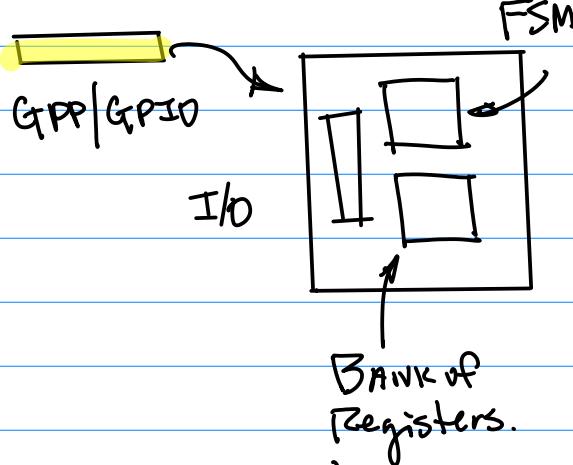
Note: Please Bring Your 1^o LTC CPU Module; 2^o USB Cable; 3^o Laptop with MCU Expresso for Inspection.

Example: Continuation of CPU Architecture

1^o Discussion

Note: A GPP/GPIO Controller is mapped to the memory map.
Read CPU Datasheet to find its Location;

2^o Special Purpose Registers.



Special Purpose Registers.

Task 1: Init & Config.

Naming Convention: URG (Uniformity, Regularity, Orthogonality)

Root (3 letters)

↓ CON

Prefix + Root
3 letters 3 letters

GPI



GPP



GPI, GPI, GPX

GPPCON

Software to Drive the Peripheral Controller

↓
Compiler, gcc. Open Source → Target CPU

Punting

CMPE240

Spring 2023

a/

2nd Special Purpose Register

GPx + DAT

Reset.

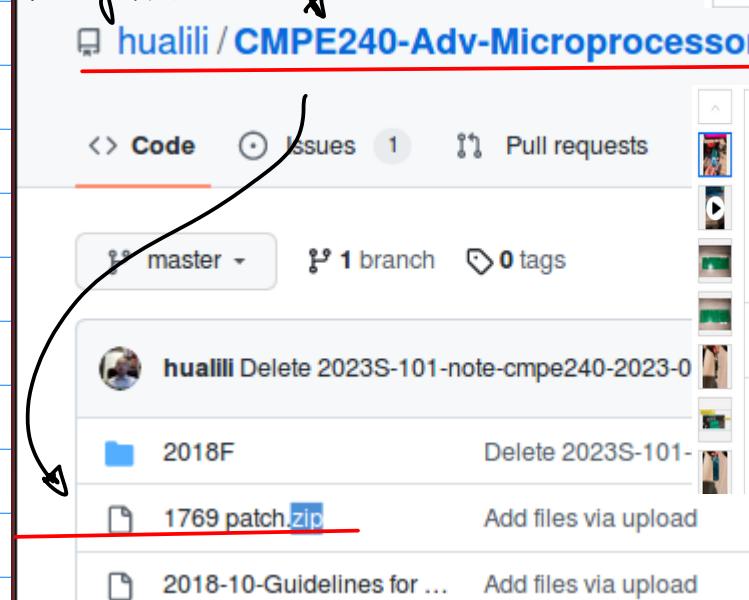
GP&DAT.

Feb 13 (Monday).

Example: Start-up MCU Xpresso.

1. Install MCU Xpresso. Start the MCU. → Project Browser (Top Left Panel) → A panel beneath it, "Import Project". Double Click / Select to open it, then follow the steps to import 17b9 patch.

2. the LPC17b9 Patch is posted on the github.



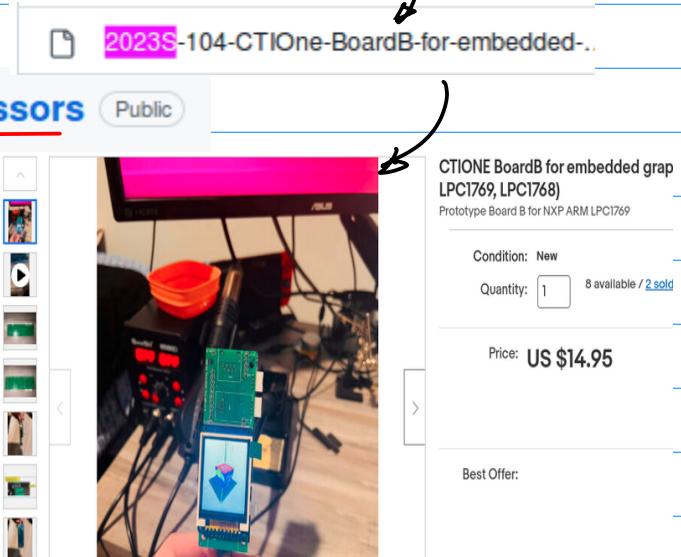
find a project with a keyword "Blanky". Then, use "Debug" option to Build it, And to observe the LED on the GPU module flashing.

4. Then Browse the LPC17b9 Patch, Locate 17xx.h which realizes the task of Porting the target platform(17b9) to the IDE Software.

Resources: The posted zip files.(projects),

LPC17b9 patch → gpio.zip → drawline (IIC24) zip
2D Graphics → 2D+3D (IIC24) projects for IIC24.

Note: for IIC24 Board)



3. Connect your LPC GPU module (LPC17b9)
Browse the imported 17b9 Projects.

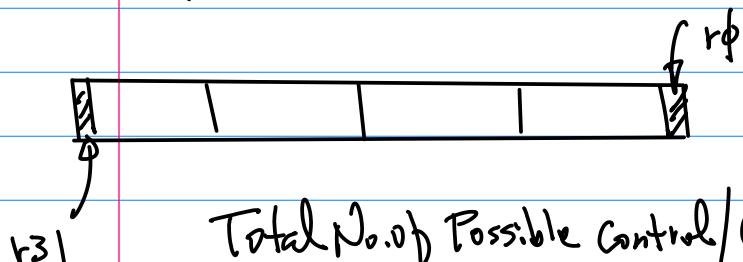


<https://www.digikey.com/en/products/detail/adafruit-industries-lc/358/5801368>



Image shown is a representation only. Exact specifications should be obtained from the product data sheet.

Example: GPxCON 32 bit.



Total No. of Possible Control/Config
 $OPS = 2^{32}$

From CPU Datasheets, Inspection:

from github ~ Cmpe244

2021F-105-#0-cpu-arm...

Notation: Vector Notation

GPACON [3:0]

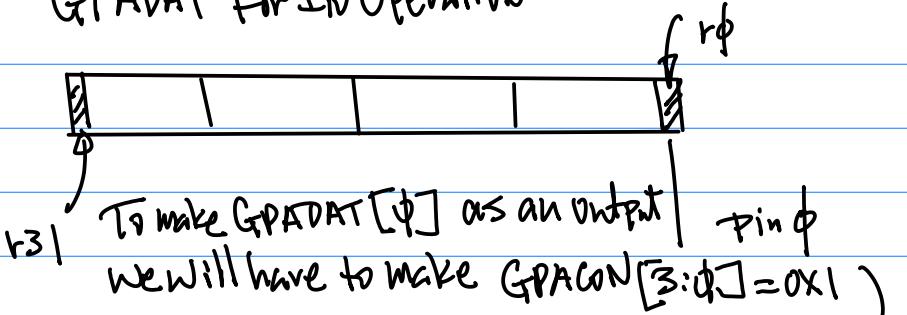
GPACONSLP	0x7F00800C	R/W	Port A Sleep mode Co
GPAPUDSLP	0x7F008010	R/W	Port A Sleep mode Pu

GPACON	Bit	Description	
GPA0	[3:0]	0000 = Input	0001 = Output
		0010 = UART RXD[0]	0011 = Reser
		0100 = Reserved	0101 = Reser
		0110 = Reserved	0111 = Extin
GPA1	[7:4]	0000 = Input	0001 = Output
		0010 = UART TXD[0]	0011 = Reser
		0100 = Reserved	0101 = Reser
		0110 = Reserved	0111 = Reser

The 1st Pin of the GPxIO Port.

CMPREGD

GPADAT For I/O operation



GPAON	Bit	Description	Initial State
GPA0	[3:0]	0000 = Input 0010 = UART RXD[0] 0100 = Reserved 0110 = Reserved	0000
GPA1	[7:4]	0000 = Input 0001 = Output	0000

From the SCD, we can find the physical pins to realize output function.

Feb 15 (wed)

Ref: 1° GPIO Code for LPC11C24

2022F-103i-lpc11c24-gpio-archive-2022-9-20.zip

2022F-103j-lec GPIO-v4-2017-2-22.pdf

For LPC1769, the Code/Project

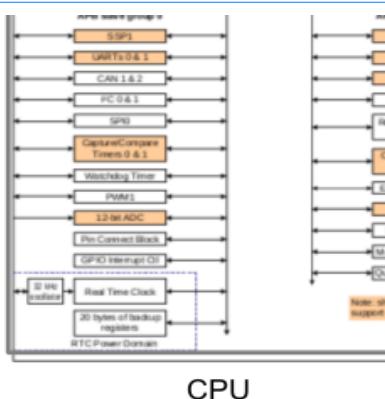
2° GPIO I/O Design.

Special purpose Register

like the following
`#define SPR 0x2000_0000`
map the CPU architecture to the arm gcc compiler

Note: 1° FIODIR, CPU Datasheet.

LPC_GPIO0->FIODIR
LPC_GPIO0->FIOSET
LPC_GPIO0->FIOCLR



2° Naming Convention.
Product Family → Peripheral Controller

Special Purpose Register

Source: 17xx.h

The mapping from CPU Architecture to the C-Code is defined by 17xx.h file with statements like:

LPC_GPIOφ → FIODIR 0x....

Memory Addr.

CPU Datasheet from .h

3° Mapping/Porting Architecture to C-code

LPC_GPIO0->FIOSET

LPC_GPIO0->FIOCLR

```
#include "../../../../tool...  
#ifdef USE_CMSIS  
/* CMSIS: the Cortex M */  
#include "LPC17xx.h"
```

Exercise: Must Do

- ① Locate LPC17xx.h
- ② Find $\text{GPIO}\phi$ part of the Code.
- ③ Locate $\text{LPC_GPIO}\phi \rightarrow \text{FIODIR}$

Find its Address.

- ④ Use CPU Datasheet to verify its Address in LTC17xx.h.

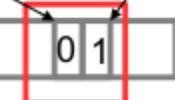
Continuation:

CPU Datasheet Table 102.

Discussion on Bit Settings.

Bit	Symbol	Value	Description
31:0	FIO0DIR		Fast GPIO Direction PORTx control bits. Bit 0 in FIOxDIR controls pin Px.0, bit 31 in FIOxDIR controls pin Px.31.
	FIO1DIR		
	FIO2DIR	0	Controlled pin is input. <i>a.</i>
	FIO3DIR	1	Controlled pin is output. <i>b.</i>
	FIO4DIR		

Bit 3 for pin 3 Bit 2 for pin 2



LPC GPIO0->FIO0DIR = 0x4; *c.*

Ref: LCD Display for LPC11C24

2022F-103f-LCD-connectivity-LPC11C24-hl-bj-2022-9-30.pdf

LPC11C24 Connectivity Table to

HL (2022-9-30) corrected this type by replacing
LPC 1769 to LPC11C24

Table 5. Connectivity Table of LPC11C24 and LCD

LPC11C24	Description	LCD
1. J6-28	3VOUT	VCC
2. J6-1	GND	GND
3. J6-8	SSEL0	TFT_CS
4. J6-14	RST	RESET
5. J6-13	D/C	A0
6. J6-5	MOSI0	SDA
7. J6-7	SCK0	SCK
8. J6-28	3VOUT	LED

Broad Walfront

TFT LCD Display:
Screen: <https://www.amazon.com/gp/product/B07BFV69DZ/>
ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1



1.8 inch SPI TFT LCD Display Module for
ST7735 128x160 51/AVR/STM32/ARM
8/16 bit

Visit the Walfront Store

4.5 stars 20 ratings

Amazon's Choice in LCD Graphic Displays by Walfront

-10% \$15.55

Price: \$19.99

prime One-Day

& FREE Returns

Includes \$1.71 Prime savings

Personalized All in One

CMPE240
Spring 2023

13

Ref: CPU Datasheet for l1C24

1. [2022F-103g-lpc11c24-cpu-datasheet-UM10398.pdf](#) Add files via upload

2. [2022F-103h-SCH-LPCXpressoLPC11C24revB.pdf](#) Add files via upload

3. [2022F-103i-lpc11c24-gpio-archive-2022-9-20.zip](#) Add files via upload

Feb 20 (Monday),
Note: 1^o Homework coming on
CANVAS;

Today's Topics:

- 1^o Summary on GPIO
- 2^o Hardware for Graphics Engine ^{2D}

Example: Ref CPU Datasheet
Table 102. for
FIODIR S.P.R.

1^o
Note: Addr. for the S.P.R. Need to Be
Cross-refer'd in
MCUXpresso Code.

2^o FIODIR
multiple.
Registers

access to a group of bits in a single GPIO port independently from other bits in the same port.

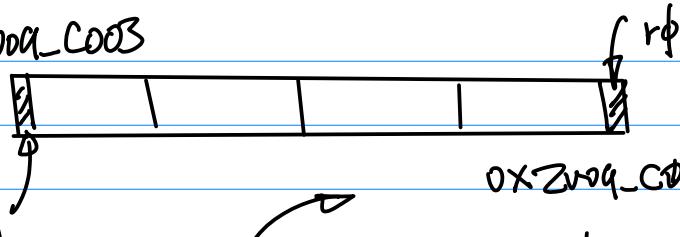
Table 102. GPIO register map (local bus accessible registers - enhanced GPIO features)

Generic Name	Description	Access	Reset value	PORTn Register Name & Address
FIODIR	Fast GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0	FIO0DIR - 0x2009 C000 FIO1DIR - 0x2009 C020 FIO2DIR - 0x2009 C040 FIO3DIR - 0x2009 C060 FIO4DIR - 0x2009 C080
FIOMASK	Fast Mask register for port. Writes, sets, clears, and reads to port (done via writes to FIOPIN, FIOSET, and FIOCLR, and reads of FIOPIN) alter or return only the bits enabled by zeros in this register.	R/W	0	FIO0MASK - 0x2009 C010 FIO1MASK - 0x2009 C030 FIO2MASK - 0x2009 C050 FIO3MASK - 0x2009 C070 FIO4MASK - 0x2009 C090

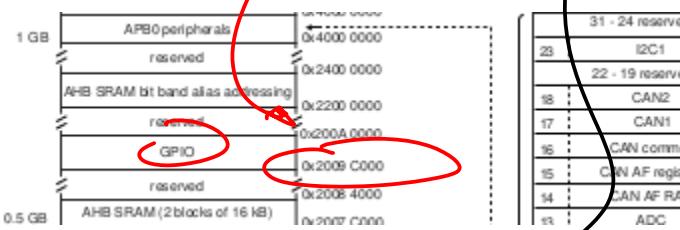
Connection To Coding: C/c++

FIO0DIR → LPC_GPIOφ → FIO0DIR

0x2009_C000 : 0x2009_C003

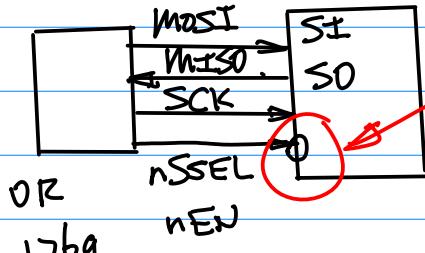


Connection To
Mem map



0x2009_C000

LPC IIC24



LCD
Active
LOW!

Fig. 1.

30. Addr. for Byte Addressable
Machine, 4 Addresses.

FIO0DIR[rx': rx], for Example

FIO0DIR[4:3]

40 Bit (Binary) Pattern for the S.P.R.

Note: Bit Rate Discussion.

Table 105. Fast GPIO port Direction control byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxDIR0	Fast GPIO Port x Direction control register 0. Bit 0 in FIOxDIR0 register corresponds to pin Px.0 ... bit 7 to pin Px.7.	8 (byte) R/W	0x00	FIO0DIR0 - 0x2009_C000 FIO1DIR0 - 0x2009_C020 FIO2DIR0 - 0x2009_C040 FIO3DIR0 - 0x2009_C060 FIO4DIR0 - 0x2009_C080
FIOxDIR1	Fast GPIO Port x Direction	8 (byte)	0x00	FIO0DIR1 - 0x2009_C001

Responsible for P0.0 ~ P0.7 (8 pins).
"1" for OutInt, "0" for the input.

* Consider Hardware Design for
LCD Display.

Background: SPI. Enable LCD

Display Device.

Serial Peripheral Interface.

Bit Rate $\sim 100\text{Mbps}$.

Graphics
Display
Resolution:

(0,0) 1024×768



Scans:

Top Left Corner (0,0),

From left to Right

Top to Bottom.

No. of pixels/Row

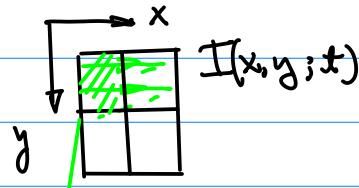
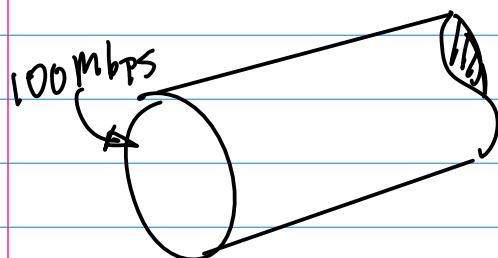
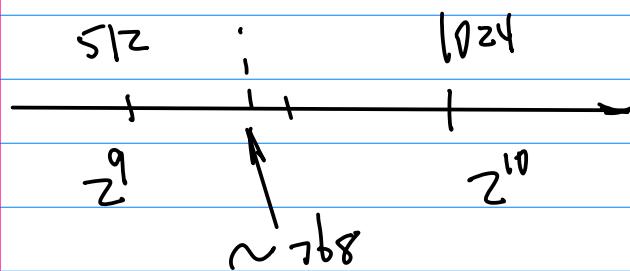
No. of Rows
Per Frame.

30 Frames Per Second (FPS)

24 Bits of color. (Color Depth)

$(0.24 \times 768 \times 30 \times 24)$ bit/sec.

$$\begin{aligned} & 2^{10} \quad 2^9 \quad 2^5 \quad 2^5 \\ & \quad \quad \quad \underbrace{\quad}_{2^{10}} = 1K \cdot 1K \cdot 2^9 \\ & 1 \quad \quad \quad \quad \quad = 1M \text{eg} \times 512 \\ & 1K \quad \quad \quad 1K \quad = 512 \text{ Mbps}, \end{aligned}$$



$$\frac{1}{4} \times 512 > 100 \text{ Mbps}$$

$\frac{1}{4}$ of the Resolution $1024 \rightarrow 1024 / 2 = 512$

SPI I/F: "3+"

$$768 \rightarrow 512 / 2 = 256$$

MISO (Master Input / Slave Output)
MOSI (Master Out / Slave Input)
SCK (Clock / Output)
SS (Slave Select / Enable): Active Low, Output from the master

Now, Let's go to Sot of the Target GRL.

2022F-103h-SCH-LPCXpressoLPC11C24revB.pdf

LPCXpresso	
GNDX	J6-1
VIN (4.5-5.5V)	
not used	J6-3
PIO0_0 RESET	J6-4
PIO0_9 MOSIO/SWO	J6-5
PIO0_8 MISO0	J6-6
PIO2_11 SCK0	J6-7
PIO0_2 SS0	J6-8
A PIO1_7 TXD	J6-9
!L PIO1_6 RXD	J6-10
PIO0_7	J6-11
PIO2_0	J6-12
X PIO2_1	J6-13
PIO2_2	J6-14
PIO0_11 AD0	J6-15

Dual row holes (2x27), 100 mil spacing

Build Connectivity Table

CPU pin	Connector	Note
pin No. / Name	Info	
PIO0_9/MOSIP	J6-5/MOSIP	

Feb 22 (Wk 6).

1° In-Class Homework 1 Pt.
Next Monday.

Homework : SPR & GPP.
In-Class.

~15 min.

Submission on-line to CANVAS.
You will need Laptop Computer.

Example: Continuation of Build GPIO to LCD Display Interface

CPU pin	Connector Info	Note
Pin No./Name		LCD Device
P100P-9/MOSI#	J6-5/MOSI#	SPI

Note: for LCD pin Connectivity Information
for LPC11C24

2022F-103f-LCD-connectivity-LPC11C24-hl-bj-2022-9-30.pdf

HL (2022-9-30) corrected this typo by replacing
LPC 1769 to LPC11C24

Table 5. Connectivity Table of LPC11C24 and LCD

LPC11C24	Description	LCD
1. J6-28	3VOUT	VCC
2. J6-1	GND	GND
3. J6-8	SSEL0	TFT_CS
4. J6-14	RST	RESET
5. J6-13	D/C	AO
6. J6-5	MOSIO	SDA
7. J6-7	SCK0	SCK
8. J6-28	3VOUT	LED

3° Control Signal from the Master, CPU,
D : Data ; C : Command
Binary Value ; 4° SDA is S.I.
Serial Inpt ; 5° for Backlight

Note: 8 pin v.s. 10 pins. if for Wirewrapping

Build, then you can use any one of the
two. Check the Latest pin Layouts
for IIC24 Prefab Board.

Software Design: Theoretical Background
CPU Architecture Coding.

Coding: MCUXpresso Installation.

SDK for the target CPU
Configuration

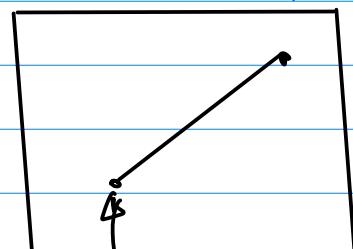
Import the existing code
LPC1769 patch.

LPC11C24 gpi0.zip

Test Drive, "Dry Run"

Ready for "DrawLine"
project (Code for
LPC1769, and code for
LPC11C24.)

$$\overrightarrow{P_i} (x_{i+1}, y_{i+1})$$



$$\overrightarrow{P_i} (x_i, y_i)$$

CPU Datasheet, pp. 43)

Note: 1^o Control Register, SSP0CR0

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

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

2^o Addr. for SSP0CR0 : 0x4008_8000.

Where is it on the Memory Map?

How is it reflected in C/C++ code,

e.g. LPC17xx.h

3^o SSP0CR0 [3:0]

Least Significant Bit
Most Significant Bit

"Little Endian"

r31

SSP0CR0

SSP0CR0 [3:0] : DataSizeSelect. (DSS).

For 8-bit Communication.

Bit	Symbol	Value	Description
3:0	DSS	0011	Data Size Select. This field controls the number of bits transferred in each frame and should not be used.
		0010	4-bit transfer
		0100	5-bit transfer
		0101	6-bit transfer
		0110	7-bit transfer
		0111	8-bit transfer
		1000	9-bit transfer
		1001	10-bit transfer
		1010	11-bit transfer
		1011	12-bit transfer
		1100	13-bit transfer

$$\text{SSP0CR0 [3:0]} = 0111 = 0 \times 7$$

Tech. Spec.
(Technical Specification)

Binary Pattern
for Init &
Config.

For SPI

$$\text{SSP0CR0 [5:4]} = 00$$

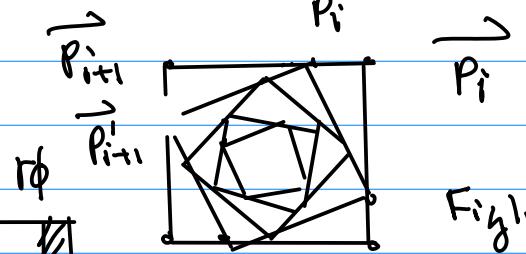
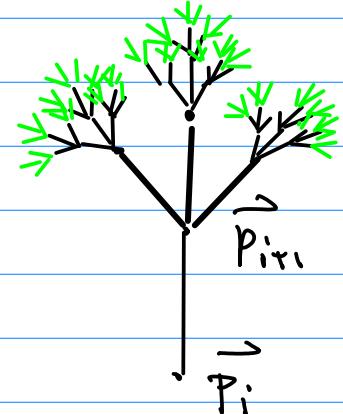
$$f_{\text{SPI}} = \frac{\text{PCLK}}{(PSDVSR * (SC2+1))}$$

$(2 \sim 255)$ ↑ ↑ ... (1)

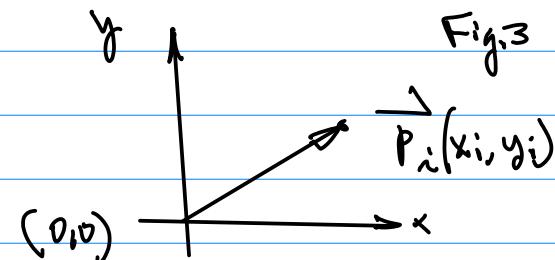
Feb 27 (Monday)

Project 1. One 2^{1/2} weeks.

Design & implement 2D Vector Graphics.

1^o Screen Saver.2^o

Background DN 2D Vector Graphics



$$\vec{P} \rightarrow \vec{P}(x, y) \rightarrow (x, y)$$

$$\vec{P}(x, y) = (x, y)$$

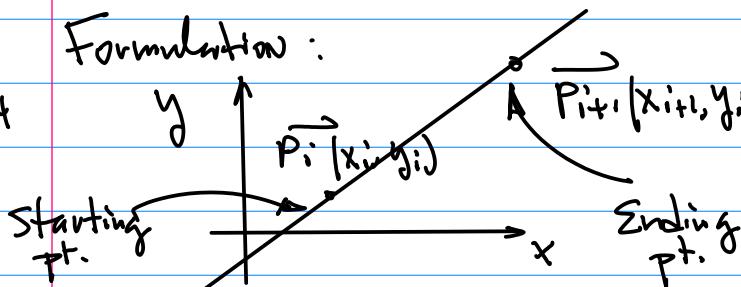
if $x < 0$, then we have all the pts defined by Eqn(z-b).

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

if $\lambda > 1$, then we have all the pts beyond the Ending Pt.

Visualize the vector $\vec{P}_i(x_i, y_i)$

Fig 4



Marchist (Wed)

Note: Project Due in 2 weeks
10 pts.

Example: Design Rotating Square

(Given $\vec{P}_i(x_i, y_i)$ and $\vec{P}_{i+1}(x_{i+1}, y_{i+1})$).

Define a directional vector.

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

$$= (x_{i+1} - x_i, y_{i+1} - y_i) \quad \dots (1)$$

Therefore,

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

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

where $\lambda \in (-\infty, +\infty)$

Question 1: Find the starting pt by Eqn(z-b)?

Question 2: Find the ending pt by Eqn(z-b)?

$$\lambda = 1$$

Question: Find All the pts Between \vec{P}_i and \vec{P}_{i+1} ? $0 < \lambda < 1$

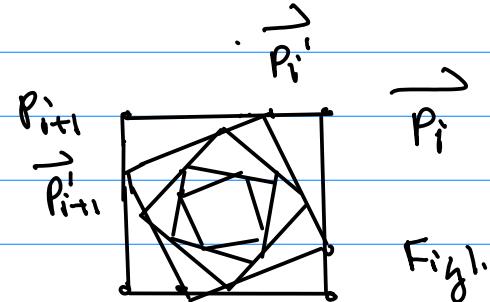


Fig.1

$$\vec{P}_{i+2}$$

Fig.1

First, To create a set of points,
 $\{\vec{P}_i(x_i, y_i) | i=0, 1, 2, 3\} \dots (1)$

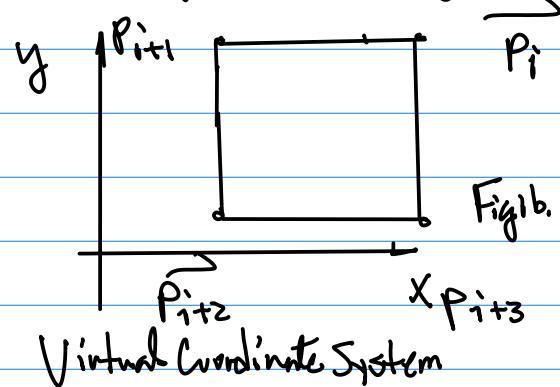


Fig.1b.

Virtual Coordinate System

Let's choose $\vec{P}_i \Big|_{i=0} = (30, 25)$

$$\vec{P}_1(10, 25), \vec{P}_2(10, 5), \vec{P}_3(30, 5)$$

Then use DrawLine Sample Code
to Draw Lines to Connect Each Pair
of points.

$$\vec{P}_0, \vec{P}_1; \vec{P}_1, \vec{P}_2; \vec{P}_2, \vec{P}_3;$$

$$\text{and } \vec{P}_3, \vec{P}_0$$

Note: in C/C++, Use mod() Function.

Then, Define the 2nd set of points.

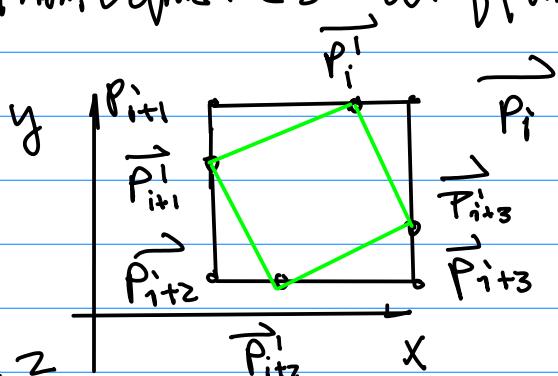


Fig. 2

Define A set of pts as

From Eqn.(2b), we have

$$\vec{P}(x, y) = \vec{P}_i(x_i, y_i) + \lambda (\vec{P}_{i+1}(x_{i+1}, y_{i+1}) - \vec{P}_i(x_i, y_i)) \quad \dots (2)$$

$$\vec{P}_0'(x_0', y_0') = \vec{P}_0^0(x_0^0, y_0^0) + \lambda (\vec{P}_1^0(x_1^0, y_1^0) - \vec{P}_0^0(x_0^0, y_0^0)) \quad \dots (3)$$

Coding it in C/C++

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

$$= (x_0^0, y_0^0) + \lambda (x_1^0 - x_0^0, y_1^0 - y_0^0) \quad \dots (3b)$$

Hence,

$$\left\{ \begin{array}{l} x_0' = x_0^0 + \lambda (x_1^0 - x_0^0) \\ y_0' = y_0^0 + \lambda (y_1^0 - y_0^0) \end{array} \right. \quad \dots (4a)$$

$$\left\{ \begin{array}{l} x_0' = x_0^0 + \lambda (x_1^0 - x_0^0) \\ y_0' = y_0^0 + \lambda (y_1^0 - y_0^0) \end{array} \right. \quad \dots (4b)$$

$$\text{lambda} = 0.2$$

$$\times [0][0] = x[0][0] + \text{lambda} * (x[1][0] - x[0][0]); \quad \dots (5)$$

Next Step, generalize it for level $j+1$
for all the points i , ($i=0, 1, 2, 3$)

$$\vec{P}_i^{j+1} = \vec{P}_i^j + \lambda (\vec{P}_{i+1}^j - \vec{P}_i^j)$$

$$\vec{P}_i^{j+1}(x_i^{j+1}, y_i^{j+1}) = \vec{P}_i^j(x_i^j, y_i^j) + \lambda (\vec{P}_{i+1}^j(x_{i+1}^j, y_{i+1}^j) - \vec{P}_i^j(x_i^j, y_i^j)) \quad \dots (6)$$

for the Coding :

$$\left\{ \begin{array}{l} x_i^{j+1} = x_i^j + \lambda (x_{i+1}^j - x_i^j) \\ y_i^{j+1} = y_i^j + \lambda (y_{i+1}^j - y_i^j) \end{array} \right. \quad \dots (7a)$$

$$\left\{ \begin{array}{l} x_i^{j+1} = x_i^j + \lambda (x_{i+1}^j - x_i^j) \\ y_i^{j+1} = y_i^j + \lambda (y_{i+1}^j - y_i^j) \end{array} \right. \quad \dots (7b)$$

Example: Design & Implement
Tree Drawing Algorithm

Next we "grow" the tree to Level 1

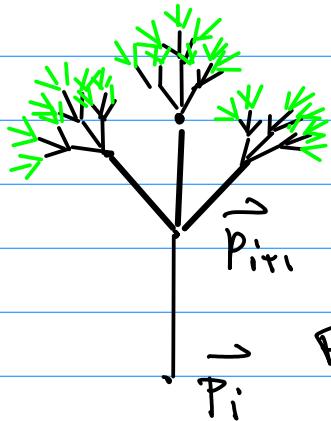


Fig.3a

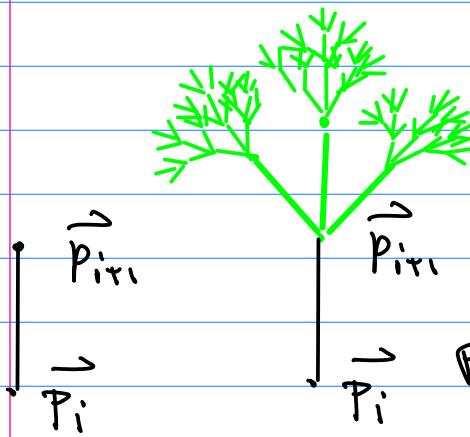
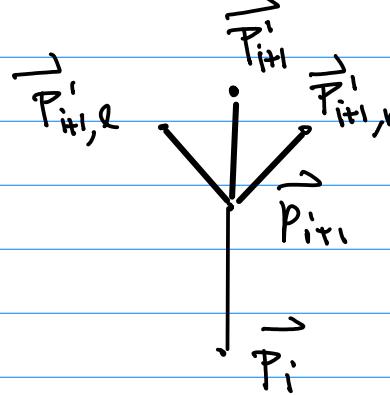


Fig.3b

Define \vec{P}_i and \vec{P}_{i+1} as points for the tree trunk.

$$\vec{P}_i(x_i, y_i) \Big|_{i=0} = \vec{P}_0(x_0, y_0) = (10, z)$$

$$\vec{P}_{i+1}(x_{i+1}, y_{i+1}) \Big|_{i=0} = \vec{P}_1(x_1, y_1) = (10, z)$$

Fig.3c

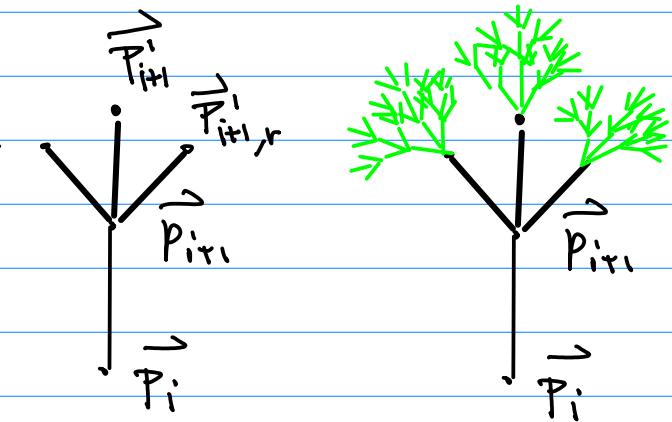


Fig.4.

where \vec{P}_{i+1}^i is the point

with 20% Reduction of its previous magnitude.

$$\vec{P}_{i+1}^i = \vec{P}_i + \lambda(\vec{P}_{i+1} - \vec{P}_i) \quad | \lambda = 0.8$$

To find pts to the right & left, Let's define 2D Transforms.

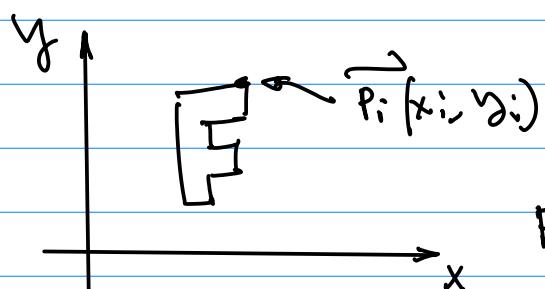


Fig.5

Define A Pattern $\{\vec{P}_i(x_i, y_i)\}_{i=0, \dots, n}\}$

$$= (10, z)$$

1. Translation.

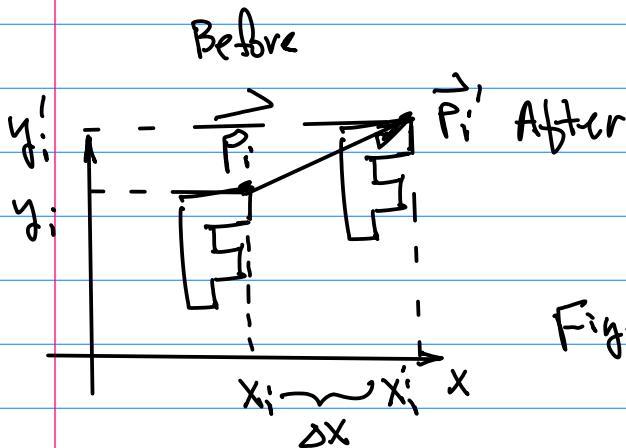


Fig.b

$$\begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \dots(1)$$

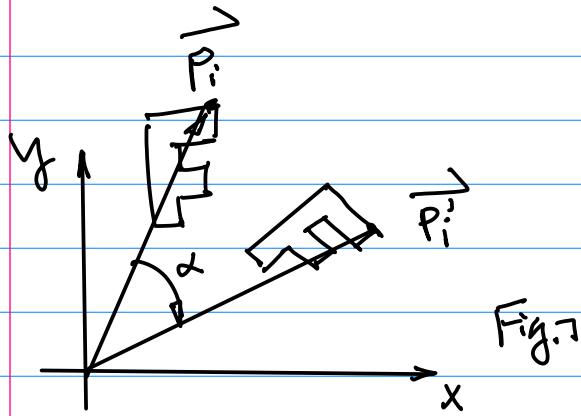
Rotation.

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

$$\{P_i(x_i, y_i)\} \xrightarrow{\phi} \{P'_i(x'_i, y'_i)\}$$

ϕ^{-1}

2. Rotation

Clockwise Rotation $\alpha < 0$ Counter Clockwise Rotation $\alpha > 0$

Let's consider the translation.

$$(x_i, y_i) \xrightarrow{\phi=?} (x'_i, y'_i)$$

Before After

$$x'_i = ? \quad x'_i = x_i + \Delta x \quad \dots(8a)$$

Where $\Delta x = x'_i - x_i$

Similarly

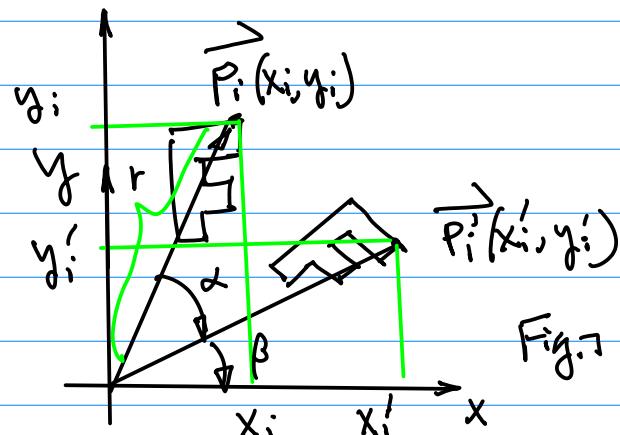
$$y'_i = y_i + \Delta y \quad \dots(8b)$$

2018F-112-lec4-2D-vectors-v3-2018-10-1.pdf

March 6 (Monday).

Note: for people who choose Pre-fab Board for the project, (1) you have 2 weeks to work on the project from the receiving of the Board; (2) Do submission of the photo of the proj.

Example: Continuation.



$$\begin{cases} x_i = r \cdot \cos(\alpha + \beta) & \dots(1a) \\ y_i = r \cdot \sin(\alpha + \beta) & \dots(1b) \end{cases}$$

$$x_i = r \cdot \cos(\alpha + \beta) \\ = r \cos \alpha \cos \beta - r \sin \alpha \sin \beta$$

where $r \cos \beta = x'_i$, $r \sin \beta = y'_i \dots (z)$.

$$\begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \dots (4)$$

Question: How to Apply the Rotation Matrix to find the pt. of the left Branch & the pt of the Right Branch.

$$\vec{P}'_{i+1,L}, \vec{P}'_{i+1,R}$$

Suppose $\vec{P}'_{i+1} = (12, 20)$

From $\vec{P}(x, y) = \vec{P}_i + \lambda(\vec{P}_{i+1} - \vec{P}_i)$

$$R_{3 \times 3}$$

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

Where $\alpha = -30^\circ$.

Note: The Rotation matrix only good w.r.t. the reference point at the origin (0,0).

Therefore, Translation of the pattern to the origin by Translation is Need.

The Translation Matrix is from Eqn [a], pp.21.

$$T_{3 \times 3} \text{ (pre-Processing)} \dots (3)$$

Where

$$\begin{aligned} \Delta x &= -12 \\ \Delta y &= -20. \end{aligned} \dots (5)$$

For the pre-processing.

Post processing using the following math. Formulation

$$T_{3 \times 3}^{-1} = \begin{pmatrix} 1 & 0 & ? \\ 0 & 1 & ? \\ 0 & 0 & 1 \end{pmatrix} \begin{matrix} -\Delta x \\ -\Delta y \\ 0 \end{matrix} \dots (b)$$

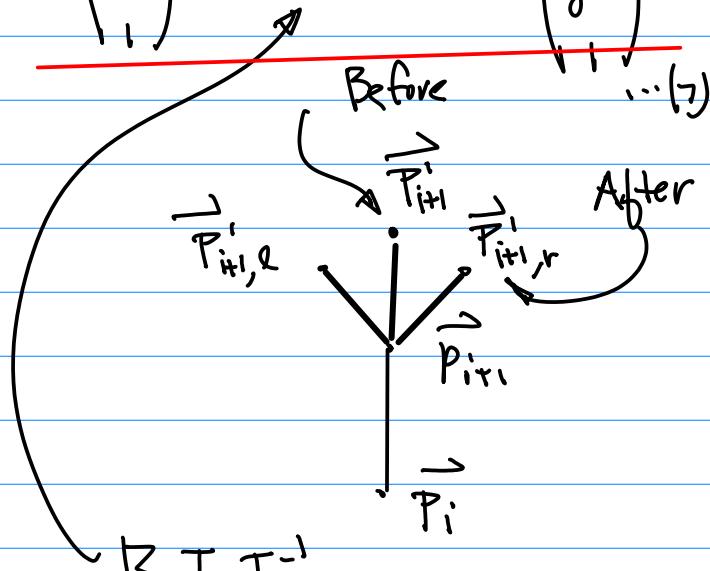
Now, Let's Combine the above steps.

After

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

Before

$$T^{-1} R T \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \dots (7)$$



$$R, T, T^{-1}$$

$$\begin{aligned}
 T^T R T &= \begin{pmatrix} 1 & 0 & -\Delta x \\ 0 & 1 & -\Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \\
 &= \begin{pmatrix} 1 & 0 & -\Delta x \\ 0 & 1 & -\Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\alpha & -\sin\alpha & \cos\alpha \cdot \Delta x - \sin\alpha \cdot \Delta y \\ \sin\alpha & \cos\alpha & \sin\alpha \cdot \Delta x + \cos\alpha \cdot \Delta y \\ 0 & 0 & 1 \end{pmatrix} \\
 &= \begin{pmatrix} \cos\alpha & -\sin\alpha & \cos\alpha \cdot \Delta x - \sin\alpha \cdot \Delta y - \Delta x \\ \sin\alpha & \cos\alpha & \sin\alpha \cdot \Delta x + \cos\alpha \cdot \Delta y - \Delta y \\ 0 & 0 & 1 \end{pmatrix} \dots (8).
 \end{aligned}$$

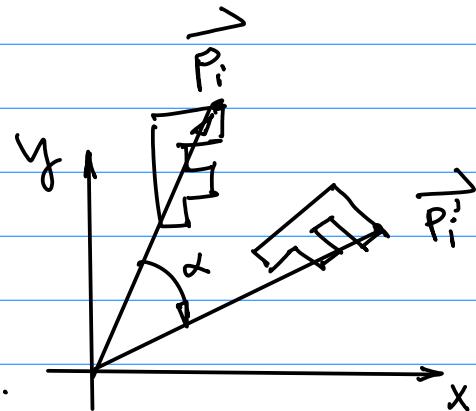
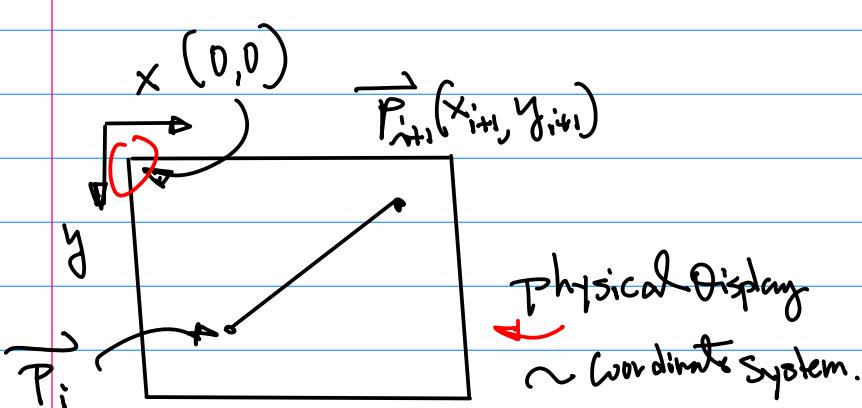
Now,

$$\begin{cases} x'_i = \cos\alpha \cdot x_i - \sin\alpha \cdot y_i + \cos\alpha \cdot \Delta x - \sin\alpha \cdot \Delta y - \Delta x & \dots (aa) \\ y'_i = \sin\alpha \cdot x_i + \cos\alpha \cdot y_i + \sin\alpha \cdot \Delta x + \cos\alpha \cdot \Delta y - \Delta y & \dots (ab) \end{cases}$$

$$\begin{aligned}
 x_{\text{prim}}[i] &= \cos(\alpha) * x[i] - \sin(\alpha) * y[i] + \\
 &\quad \cos(\alpha) * \Delta x - \sin(\alpha) * \Delta y - \Delta x
 \end{aligned}$$

where $\Delta x = -r_z$, $\Delta y = -z_0$, $x_i = rz$, $y_i = z_0$

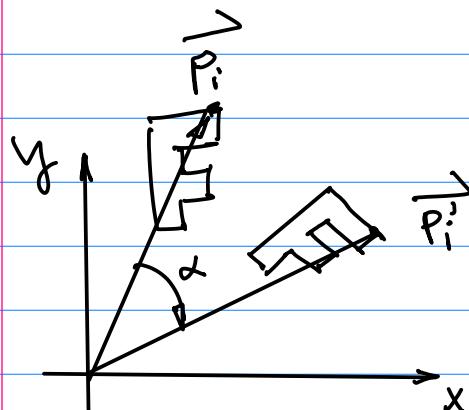
Consider the Hardware of A
Display Device.



CMPE240

Spring 2023

24



$$x \in (-\infty, +\infty), y \in (-\infty, +\infty)$$

↓ Normalization.

$$x \in (-1, +1), y \in (-1, +1)$$

Virtual Display / Coordinate System

Physical Display
(x_p, y_p)

Virtual Display
(x_v, y_v)

From PP. 23

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

↑ Direction
↓ Offset

Half of the Resolution along x-direction.

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

(Half of the No. of Rows)

After
 y_p Before
 y_v

$$y_p = y_v$$

$$y_p = -y_v$$

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

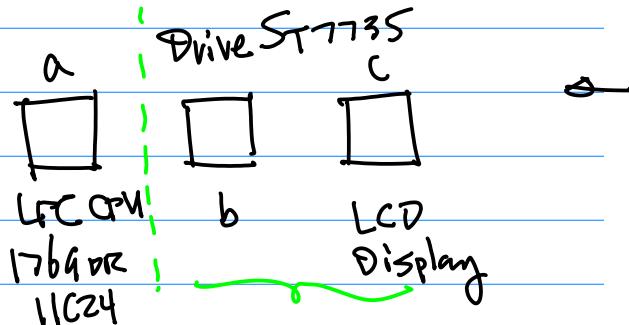
Ref:

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

March 8 (Wed)

Midterm Exam is scheduled on March 2nd (Wed).

Example: Lecture Notes pp. 23



Note: MxN Resolution

↑
No. of Col.
X
↓
No. of Rows
Y

Example: Sample Code Pending.

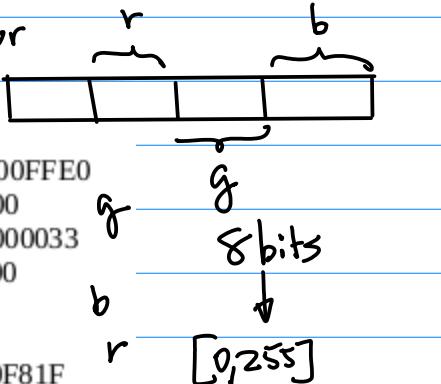
2021F-111b-handout-drawline-dotC.pdf

Note: 1^o Resolution

#define ST7735_TFTWIDTH 127
#define ST7735_TFTHEIGHT 159

#define ST7735_CASET 0x2A
#define ST7735_RASET 0x2B
#define ST7735_RAMWR 0x2C
#define ST7735_SLOUT 0x11
#define ST7735_DISPON 0x29

Note 2: 24 Bit Color



```
// defining color values
#define LIGHTBLUE 0x00FFE0
#define GREEN 0x00FF00
#define DARKBLUE 0x000033
#define BLACK 0x000000
#define BLUE 0x0007FF
#define RED 0xFF0000
#define MAGENTA 0x00F81F
#define WHITE 0xFFFFF
#define PURPLE 0xCC33FF
```

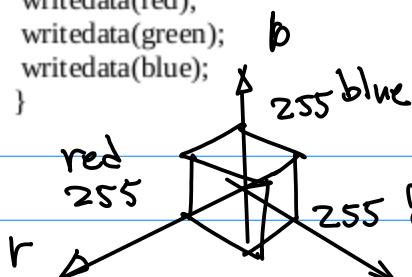
Note: SPI About

Communication
Between a $\frac{1}{2}$ b.`void spiwrite(uint8_t c)`

```
{
    int pnum = 0;
    src_addr[0] = c;
    SSP_SSELToggle( pnum, 0 );
    SSPSend( pnum, (uint8_t *)src_addr, 1 );
    SSP_SSELToggle( pnum, 1 );
}
```

Note: Related to Color Space.

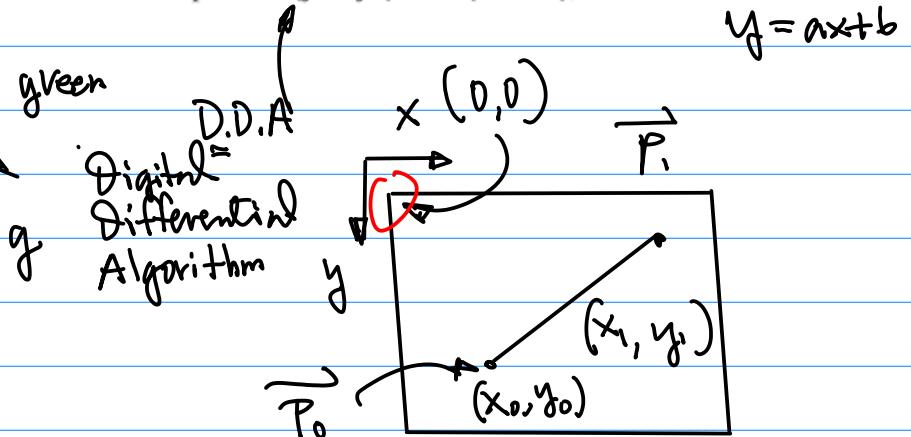
```
void write888(uint32_t color, uint32_t repeat)
{
    uint8_t red, green, blue;
    int i;
    red = (color >> 16);
    green = (color >> 8) & 0xFF;
    blue = color & 0xFF;
    for (i = 0; i < repeat; i++) {
        writedata(red);
        writedata(green);
        writedata(blue);
    }
}
```



```
void drawPixel(int16_t x, int16_t y, uint32_t color)
{
    if ((x < 0) || (x >= _width) || (y < 0) || (y >= _height))
        return;

    setAddrWindow(x, y, x + 1, y + 1);
    writecommand(ST7735_RAMWR);
    write888(color, 1);
}
```

```
void drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint32_t color)
{
    int16_t slope = abs(y1 - y0) > abs(x1 - x0);
```



Note: More discussion later.

```
void lcd_init()
{
    int i;
    printf("LCD Demo Begins!!!\n");
    // Set pins P0.16, P0.21, P0.22 as output
    LPC_GPIO0->FIODIR |= (0x1<<16);
```

```
LPC_GPIO0->FIODIR |= (0x1<<21);
LPC_GPIO0->FIODIR |= (0x1<<22);
```

```
// Hardware Reset Sequence
LPC_GPIO0->FIOSET |= (0x1<<22);
lcddelay(500);
```

```
LPC_GPIO0->FIOCLR |= (0x1<<22);
lcddelay(500);
```

```
LPC_GPIO0->FIOSET |= (0x1<<22);
lcddelay(500);
```

Note: Together with Draw Line function,
we can build different Apps.

Background DN 3D Graphics.

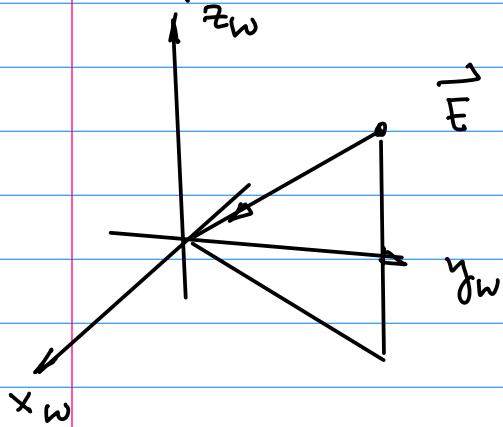


Fig.1

$x_w - y_w - z_w$ World Coordinate System.

R.H. Right hand System.

Define A virtual Camera:

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

Direction of the Virtual Camera:

points to $(0,0,1)$;

Virtual Camera model: "Pin-Hole"

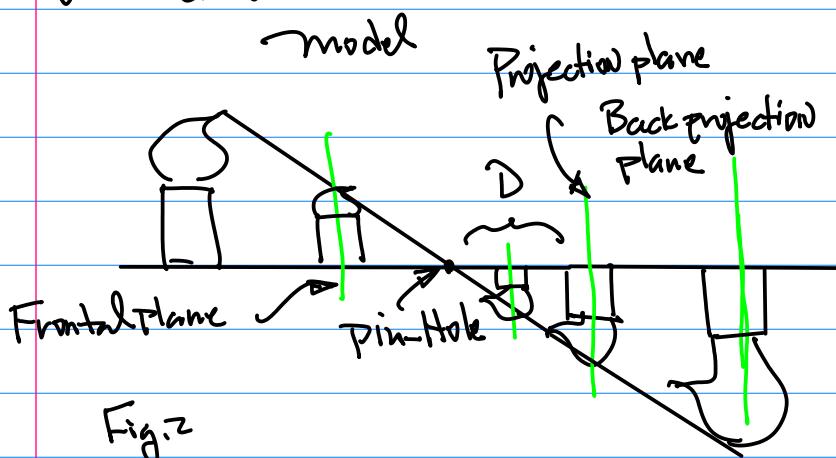


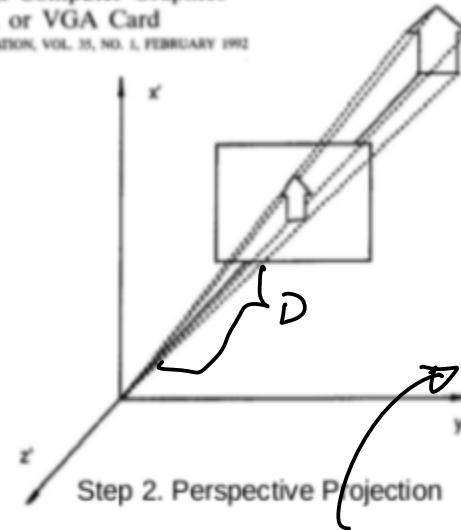
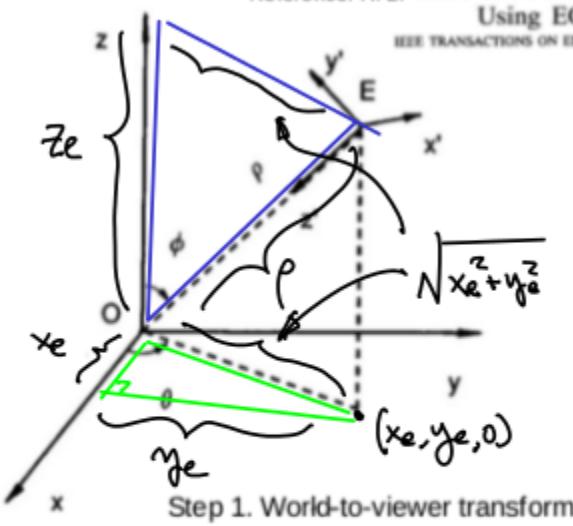
Fig.2

Design A Technique Called Transformation Pipeline to Be Able to generate 3D Graphics.

Note: This project (Perspective) is characterized by (1) "D" focal length, (2) the "-" sign of the projected image. See Eqn(z).

From pp.32, github, Lecture Notes

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



$$\frac{x_p}{x_e} = \frac{D}{z_e}$$

$$\frac{y_p}{y_e} = \frac{D}{z_e}$$

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

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

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

... (z)

Huiy Li, Ph.D.

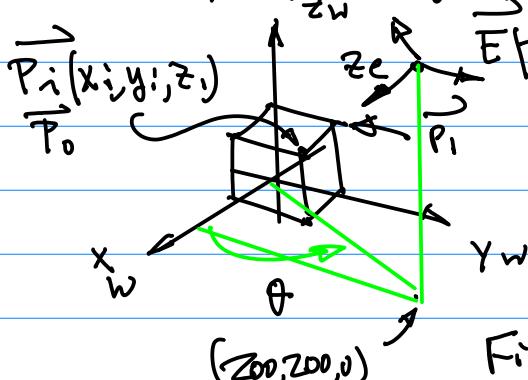
March 13 (Monday).

1) Midterm Exam 1 hr. Sharp.
(5 min. for the submission.)

Example: Continuation.

1° $x_w - y_w - z_w \rightarrow x_v - y_v - z_v$
World Virtual Camera

By Eqn(1), the mapping from the "world" to the "viewer" Coordinate System is defined.



$$\begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} ? \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

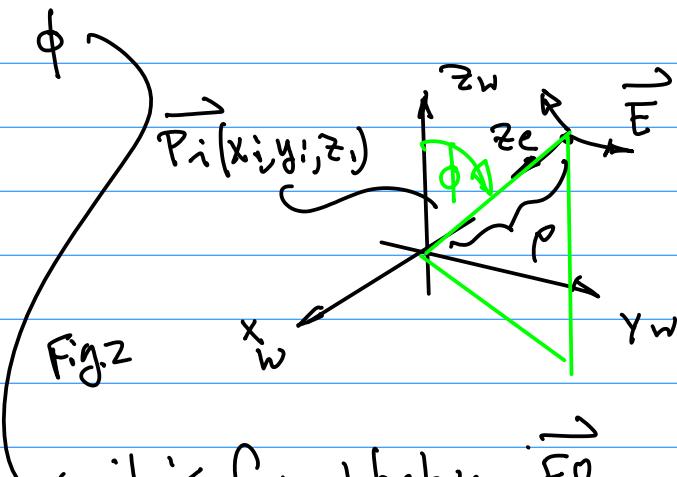
Consider

$$\begin{pmatrix} - & 0 & . & - & - \\ - & - & & & \\ - & - & - & - & 0 \\ 0 & . & . & . & 1 \end{pmatrix}$$

$$\begin{pmatrix} S & C \\ C & S \\ C & C \\ . & . \end{pmatrix}$$

Note: θ Angle is defined ON X_w-Y_w Plane, it is formed by connecting the intersection pt from E to the X_w-Y_w plane to the origin of $X_w-Y_w-Z_w$

$\theta \geq 0$ if it is
Counterclockwise



it is formed between \vec{E} &
vector \vec{Z}_w -axis.

$\phi \geq 0$ if Counter Clockwise

Then,

$$\left(\begin{array}{ccc} C & S \\ S & C \end{array} \right)$$

$$x'_i = -\sin\theta \cdot x_i + \cos\theta \cdot y_i \quad \dots (3)$$

$$y'_i = -\cos\phi \cos\theta x_i - \cos\phi \sin\theta y_i + \sin\phi z_i \quad \dots (4)$$

$$z'_i = -\sin\phi \cos\theta x_i - \sin\phi \cos\theta y_i - \cos\phi z_i + p \quad \dots (5)$$

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

Example: Project Related Design.

Step1. Design & Define A Cube

$$\{\vec{P}_i(x_i, y_i, z_i) \mid i=0, 1, \dots, 7\}$$

Size = 100, Elevate the cube
by 10.

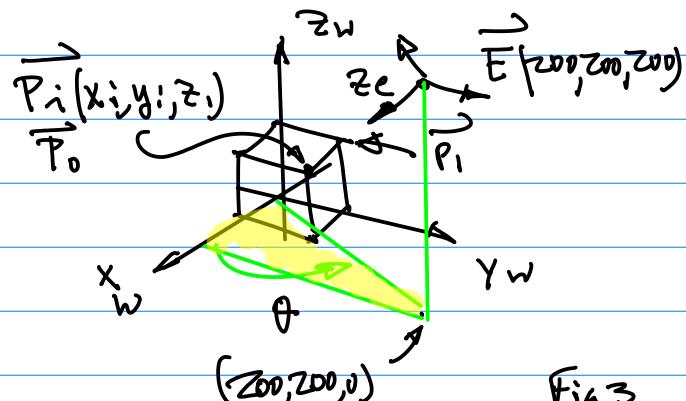
$$\vec{P}_0(x_0, y_0, z_0) = (100, 100, 110)$$

$$\vec{P}_1(x_1, y_1, z_1) = (0, 100, 110)$$

Counterclockwise when viewing
the object from the outside

$$\text{Step2. } \vec{E}(x_e, y_e, z_e) = (200, 200, 200)$$

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



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

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

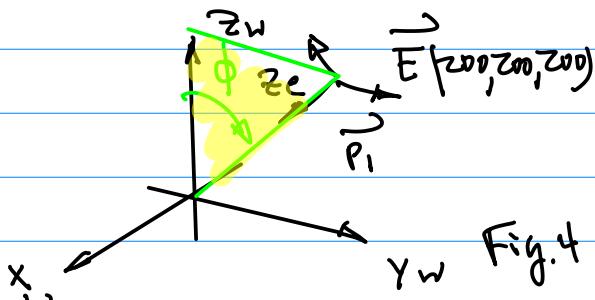
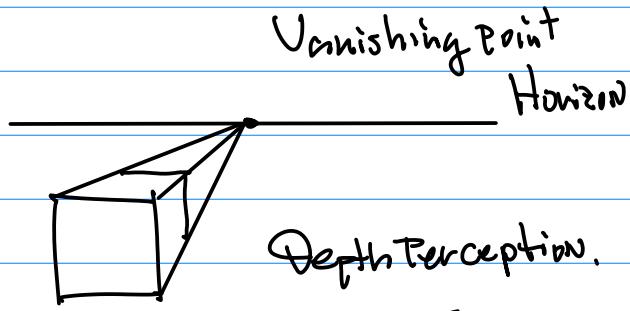


Fig.4



Vanishing Point

Horizon

Depth Perception.

Fig.5

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

$$= \frac{200}{200\sqrt{3}} = \sqrt{3}/3$$

$$\sin\phi = \frac{\sqrt{x_e^2 + y_e^2}}{\sqrt{x_e^2 + y_e^2 + z_e^2}} = \sqrt{2}/\sqrt{3}$$

Step 3. Let's make $\theta = 50$

From Eqn (2), Perspective Projection
we have

$$\left\{ \begin{array}{l} \frac{x_p}{x_e} = \frac{D}{z_e} \\ \frac{y_p}{y_e} = \frac{D}{z_e} \end{array} \right.$$

or

$$\left\{ \begin{array}{l} x_p = \frac{D}{z_e} \cdot x_e \\ y_p = \frac{D}{z_e} \cdot y_e \end{array} \right.$$

or

$$\left\{ \begin{array}{l} x_i'' = \frac{D}{z_i'} \cdot x_i' \\ y_i'' = \frac{D}{z_i'} \cdot y_i' \end{array} \right.$$

March 15 (W.e.d)

Q, Q & A on SPI Architecture

1° CPU Datasheet pp 41-31.

Table of C120

C[Rd[31:1]] Fields of the Register.

Clock Rate

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

(
[2, 254]
Sample Code

Tech. Spec. 120MHz? 50MHz

System Clock 200 MHz \rightarrow PCLK 120MHz

$$\text{Since } f_{SPI} = 50 \times 10^6 \quad PCLK = 120 \times 10^6$$

$$50 \times 10^6 = \frac{120 \times 10^6}{DVSZ * (SCR+1)}$$

$$DVSZ * (SCR+1) = \frac{120 \times 10^6}{50 \times 10^6} = \frac{12}{5}$$

Let DVSZ = 2,
then $= 2.4$

$$\text{SCR} + 1 = \frac{12}{5} \times \frac{1}{2} = \frac{6}{5}$$

$$\text{SCR} = \frac{6}{5} - \frac{5}{5} = \frac{1}{5}$$

Choose $\text{SCR} = 0$

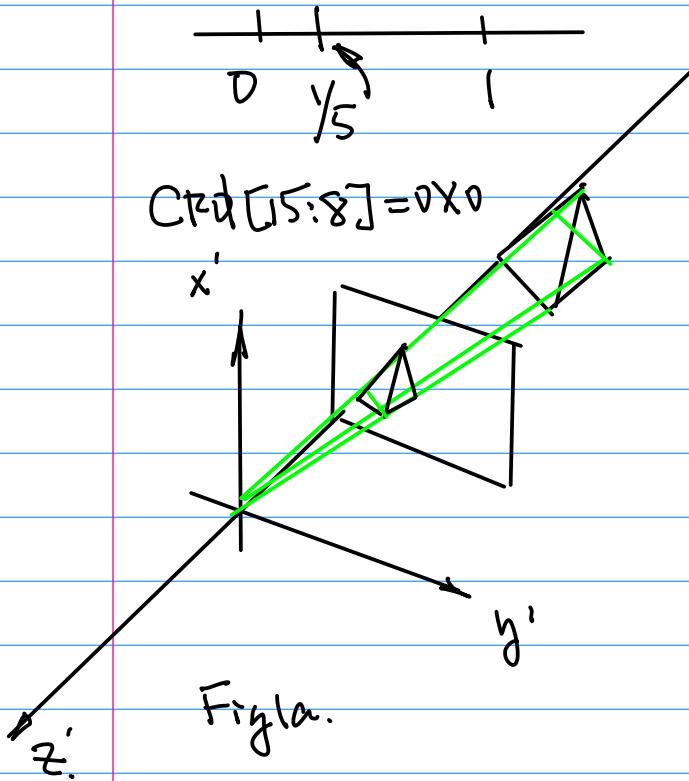


Fig 1a.

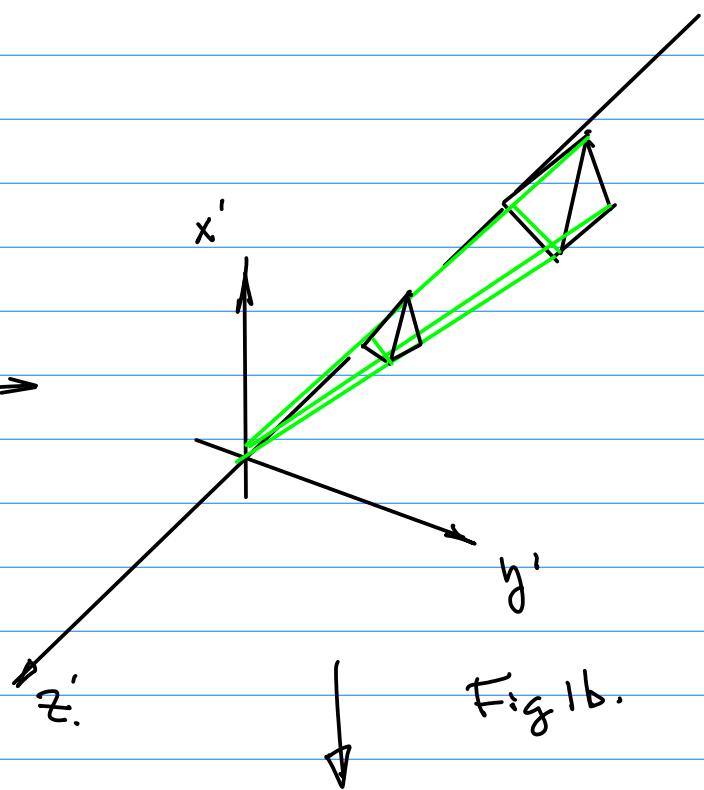


Fig 1b.

Example: Shadow Computation $\frac{D}{z_i}$

Given: 1' A Cube
 $\frac{D}{z_0}$

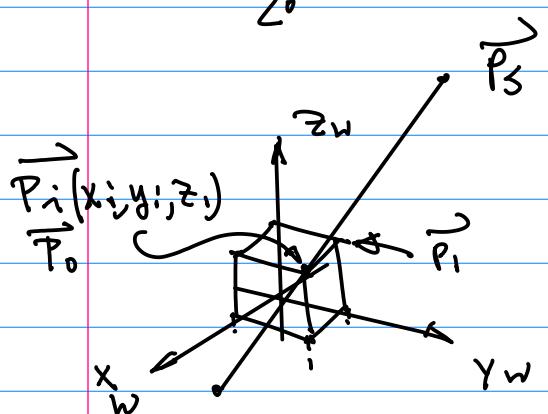


Fig 1c.

March 20 (Monday)

Q & A Session:

1° CRP (PD.43)

2° Init & Config. of SPI

Controller — In the Scope
of the Required material.

Ref 2018F-110-SPILCD-Software-HL-2018-9-19.pdf

Coding for 3D, Not in the Coming midterm.

Example: ppb4, Table 4b.

Note: To correlate the PCONP (SP.R) in the CPU Datasheet with the C-Code.

① C-Code Naming Convention of the S.P.R. → LPC17xx.h⁽²⁾
(Under defined "struct")

register is possible only if the peripheral is enabled in the PCONP register.

Table 46. Power Control for Peripherals register (PCONP - address 0x400F C0C4) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved.	NA
1	PCTIM0	Timer/Counter 0 power/clock control bit.	1
2	PCTIM1	Timer/Counter 1 power/clock control bit.	1
3	PCUART0	UART0 power/clock control bit.	1
4	PCUART1	UART1 power/clock control bit.	1
5	-	Reserved.	NA
6	PCPWM1	PWM1 power/clock control bit.	1

① Addr. ② Ptr. C-Code



③ Bit Pattern to Turn-on/off SPI Controller.

7	PCI2C0	The I ² C0 interface power/clock control bit.
8	PCSPI	The SPI interface power/clock control bit.
9	PCRTC	The RTC power/clock control bit.
10	PCSSP1	The SSP 1 interface power/clock control bit.
11	-	Reserved.
12	PCADC	A/D converter (ADC) power/clock control bit. Note: Clear the PDN bit in the AD0CR before clearing this bit, and this bit before setting PDN.
13	PCCAN1	CAN Controller 1 power/clock control bit.
14	PCCAN2	CAN Controller 2 power/clock control bit.
15	PCGPIO	Power/clock control bit for IOCON, GPIO, and GPIO interrupts.
16	PCRIT	Repetitive Interrupt Timer power/clock control bit.
17	PCMCPWM	Motor Control PWM
18	PCQEI	Quadrature Encoder Interface power/clock control bit.
19	PCI2C1	The I ² C1 interface power/clock control bit.
20	-	Reserved.
21	PCSSP0	The SSP0 interface power/clock control bit.
22	PCTIM2	Timer 2 power/clock control bit.
23	PCTIM3	Timer 3 power/clock control bit.

Example:

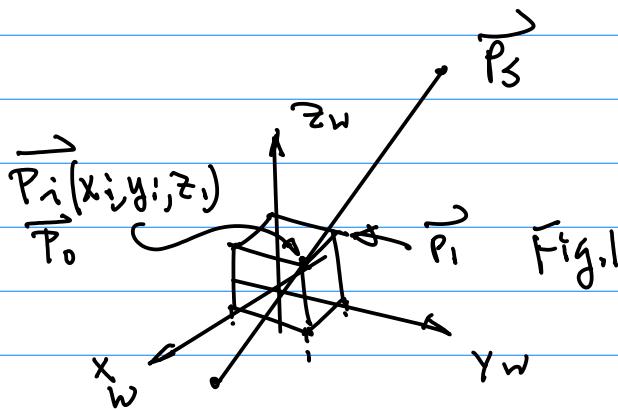


Fig.1

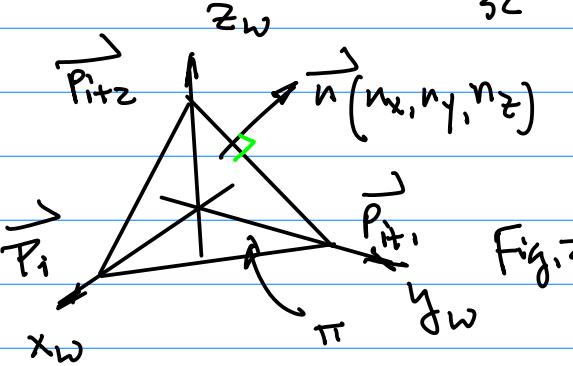


Fig.2

$\vec{n}(n_x, n_y, n_z)$: Normal Vector.

Perpendicular to the plane π

$$(\vec{P}_{i+1} - \vec{P}_i) \times (\vec{P}_{i+2} - \vec{P}_i) \quad \dots (2)$$

$$\text{So } \vec{n} = \frac{(\vec{P}_{i+1} - \vec{P}_i) \times (\vec{P}_{i+2} - \vec{P}_i)}{\|(\vec{P}_{i+1} - \vec{P}_i) \times (\vec{P}_{i+2} - \vec{P}_i)\|_2} \quad \dots (2-b)$$

From Eqn(2-b), Let's Define

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

Now, Let's Define A Plane Equation
Using the Normal Vector.

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

Any Arbitrary Point On the Plane

Arbitrary Known Point
On the Plane.

1. Define A point Light Source

$$\vec{P}_s(x_s, y_s, z_s) = (-100, 100, 350)$$

2. Create 4 Ray Equations

By Linking \vec{P}_s to \vec{P}_i of the top Surface of the Cube

$$\vec{R}(x, y, z) = \vec{P}_i(x_i, y_i, z_i) + \lambda (\vec{P}_s - \vec{P}_i)$$

... (1)

$$\text{For } \lambda=1, \vec{R}(x, y, z) = \vec{P}_s$$

$$\lambda=0, \vec{R}(x, y, z) = \vec{P}_i$$

Find λ , which defines the intersection pt. of the Ray

Equation with the x_w - y_w plane.

For that purpose, define A plane equation.

April 3rd (Mon).

Example: Find the intersection

Point of the Ray Equation
and the plane Equation.Sol: An Intersection Point↓
A Common pt Shared
By Both Equations.

$$\vec{n} \cdot \vec{R} = 0$$

$$\vec{R} = \vec{P}_i + \lambda (\vec{P}_s - \vec{P}_i)$$

$$\vec{n} \cdot (\vec{P}_i + \lambda (\vec{P}_s - \vec{P}_i)) = 0$$

$$\vec{n} \cdot \vec{P}_i + \lambda \vec{n} \cdot (\vec{P}_s - \vec{P}_i) = 0$$

$$\lambda = -\frac{\vec{n} \cdot \vec{P}_i}{\vec{n} \cdot (\vec{P}_s - \vec{P}_i)} \quad \dots (4)$$

$$\begin{cases} \vec{R}(x, y, z) = \vec{P}_i(x_i, y_i, z_i) + \lambda (\vec{P}_s - \vec{P}_i) \\ \vec{n} \cdot (\vec{v} - \vec{a}) = 0 \end{cases} \quad \dots (1a) \quad \dots (1b)$$

 \vec{x} is unknown in Egn(1a). \vec{v} is unknown in Egn(1b).the common point for both Equations is
 \vec{R} from the Ray, and also on the
plane π , Hence,

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

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

$$\dots (4b)$$

Ref: Calculation of An Intersection

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

$$\vec{v} = \vec{R} \quad \dots (2)$$

From Egn(1b). we have

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

$$\vec{n} \cdot (\vec{R} - \vec{a}) = 0 \quad \dots (3)$$

 \vec{a} is any known pt on the
 X_w-Y_w plane. $\vec{a} = (1, 1, 0)$ works, But choose
 $\vec{a} = (0, 0, 0)$, So Egn(3):

CMPE240 April 4, 2023

Example: Ref (see pp.1. github

Lecture Notes, 2021F-101b-n)

Given $\vec{P}_s(-20, 110, 200)$,

vertex of a given cube

 $\vec{P}_i(100, 100, 100)$. Find the intersection

pt to draw shadow.

$$\lambda = \frac{\vec{n} \cdot (\vec{a} - \vec{P}_i)}{\vec{n} \cdot (\vec{P}_s - \vec{P}_i)} \quad \dots (5)$$

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

$$\lambda = \frac{-\vec{n} \cdot \vec{P}_i}{\vec{n} \cdot (\vec{P}_s - \vec{P}_i)}$$

Sol: First, Ray Equation, Egn(1): Now, hand Calculation, alsoNote: the Result is in $X_w-Y_w-Z_w$ world.
So to be able to plot it, we will need

to perform Transformation pipeline,
then Conduct mapping from the
Virtual to physical Coordinate.

④ Decoration (Texture Mapping)

Trees
Squares → on the surface of A cube.

April 5 (Wed)

Project (20 pts). April 26 (Wed)

1° Detailed Requirements

↳ Rubrics will be posted
on CANVAS.

Font Design. Place An initial on the
Surface.

Example: Source Code Walk-Through

[/2018F/2018F-116-11diffuse20181114.cpp](#)

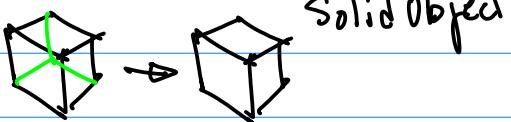
2° Highlights of Requirements.

① 2 3D Objects < in Cube
a top half
of a sphere.

↳ Implemented in OpenGL Portable

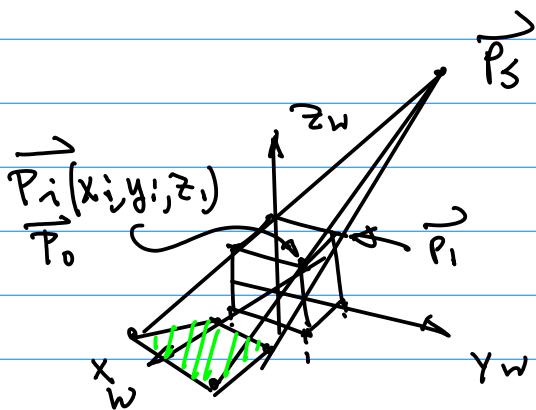
2°

Wireframe model

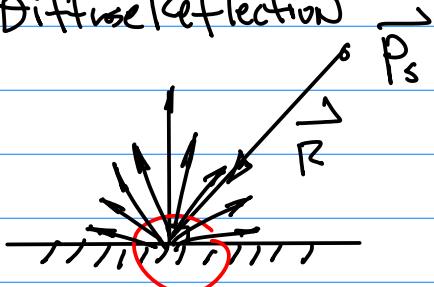


Solid Object

② Shadow Computation & Visualization



③ Diffuse Reflection



```

--  

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;  

37 //-----point light source defined in line 160

```

Projection Plane in Perspective Projection

```

39 typedef struct {  

40     float X[UpperBD], Y[UpperBD], Z[UpperBD];  

41 } pworld;      pts in Xw-Yw-Zw  

42  

43 typedef struct {  

44     float X[UpperBD], Y[UpperBD], Z[UpperBD];  

45 } pviewer;      pts in Xe-Ye-Ze  

46  

47 typedef struct{  

48     float X[UpperBD], Y[UpperBD];  

49 } pperspective;  

50     pts in (x'', y'')  

51 typedef struct{  

52     float X[30], Y[30];  

53 } letter;      for Font Design.  

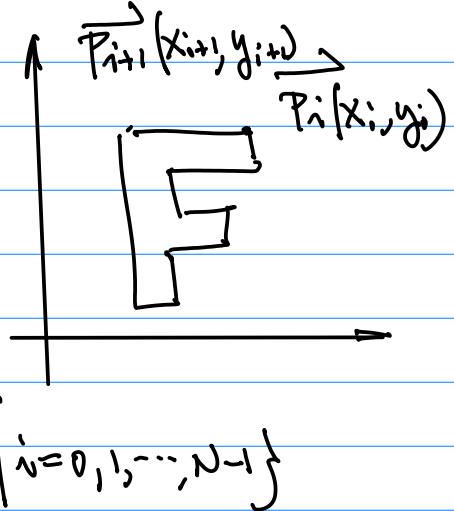
54  

55 typedef struct {  

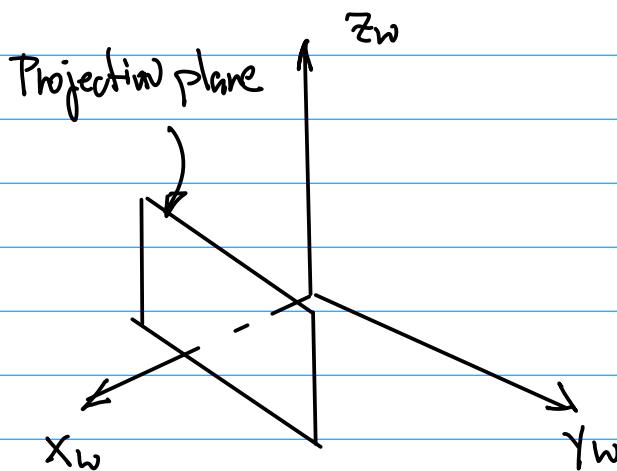
56     float r[UpperBD], g[UpperBD], b[UpperBD];  

57 } pt_diffuse;    r:red, g:green, b:blue

```



$$I(x, y) = (r(x, y), g(x, y), b(x, y))$$



Note: Define $x_w = y_w = z_w$.

```

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

```

Font Design \rightarrow

```

91
98     //-----letter-----
99     letterL.X[0] = 10.0; letterL.Y[0] = 10.0;
100    letterL.X[1] = 20.0; letterL.Y[1] = 10.0;
101    letterL.X[2] = 20.0; letterL.Y[2] = 40.0;
102    letterL.X[3] = 40.0; letterL.Y[3] = 10.0;
103    letterL.X[4] = 50.0; letterL.Y[4] = 10.0;
104    letterL.X[5] = 30.0; letterL.Y[5] = 50.0;

```

For $\sin\theta, \cos\theta, \sin\phi, \cos\phi$ in the matrix of the transformation pipeline.

```

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

```

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

```

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

```

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

```

171 //-----lambda for Intersection pt on xw-yw plane-----
172 float temp = (world.X[47]*(world.X[46]-world.X[45]))
173 +(world.Y[47]*(world.Y[46]-world.Y[45]))
174 +(world.Z[47]*(world.Z[46]-world.Z[45]));
175 float lambda = temp / ((world.X[47]*(world.X[45]-world.X[7]))
176 +(world.Y[47]*(world.Y[45]-world.Y[7]))
177 +(world.Z[47]*(world.Z[45]-world.Z[7])));
178 float lambda_2 = temp / ((world.X[47]*(world.X[45]-world.X[6]))
179 +(world.Y[47]*(world.Y[45]-world.Y[6]))
180 +(world.Z[47]*(world.Z[45]-world.Z[6]));
181

```

for \vec{R} Ray Equations

Find the intersection points.

```

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

```

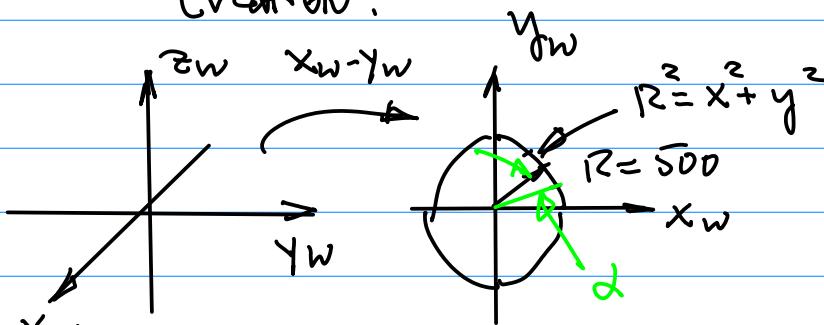
April 7 (Monday).

Note: Project in 3D is
Due in 2 Weeks.

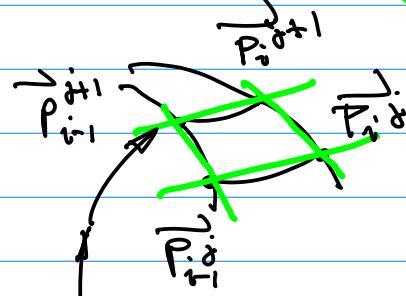
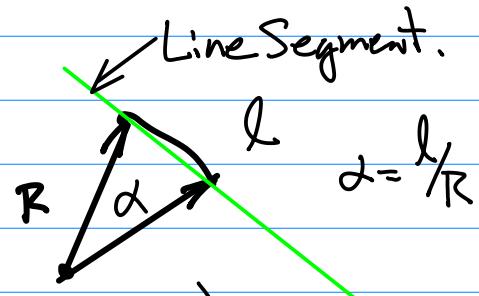
See the previous announcement

(CANVAS Posting By the
end of the Day Today),

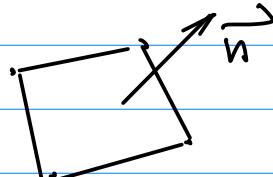
A&A. Spherical Surface
Creation.



Define incremental α
make smaller $\alpha \approx 5^\circ$



R Reduced By Predefined
proportion. Make at least
10 layers for Better
Visualization.



In Summary, we'll Create a Collection of Points

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

Example:

Ref:

- 2018F-115-lab-DiffuseReflection-R...
- 2018F-116-11diffuse20181114.cpp

Previous Project

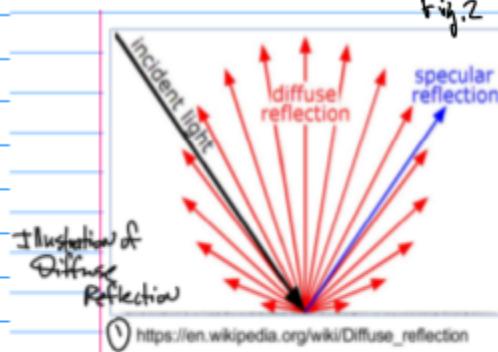
Sample code

Digital Differential Algorithm

- 2018F-117-12dda.cpp
- 2018F-118-13diffuseInterpolation20...

1. Definition.

Directions.



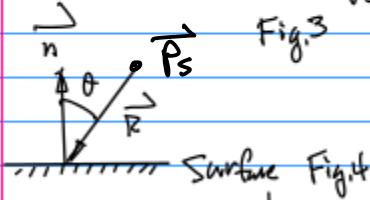
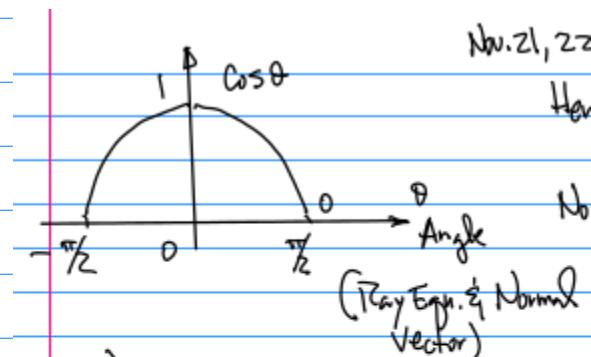
2. Intensity of the Diffuse Reflection, The Intensity of

$$I(x, y) = (r(x, y), g(x, y), b(x, y))$$

red green blue

depends on the incoming angle of the Ray Equation.

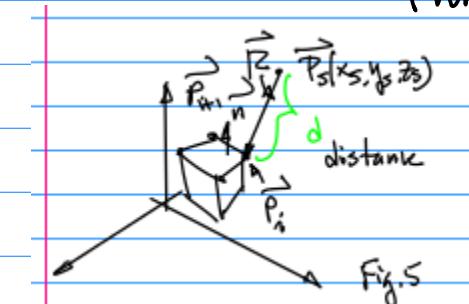
From Ref:



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

Note: \vec{n} Normal Vector of the Surface
 \vec{r} Ray Equation from the Light Source $P_s(x_s, y_s, z_s)$ to the point of Interests

From PP. 48



Note: Ray Equations:

\vec{r}_j from P_s, \vec{P}_i
 $\vec{r}_{i+1} \dots \vec{P}_s, \vec{P}_{i+1}$
 \vdots
 $\vec{r}_{i+3} \dots \vec{P}_s, \vec{P}_{i+3}$

From the Ray Equation.

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

$$\vec{n} \cdot \vec{r} = \|\vec{n}\| \|\vec{r}\| \cos\theta \dots \text{ (2)}$$

$$\therefore \cos\theta = \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} \dots \text{ (3)}$$

$I_{diff}(x, y)$ OR $I_d(x, y, z)$

↑ "World"

$$I_d(x, y, z) \approx \cos\theta = \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} \dots \text{ (4)}$$

Next, Consider the distance (Squared)

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

then, update Eqn (4),

mode

$$I_d(x, y, z) \approx \frac{1}{\|\vec{r}\|^2} \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} \dots \text{ (5)}$$

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

April 12 (Wed).

Note: Project Assignment is posted on CANVAS.

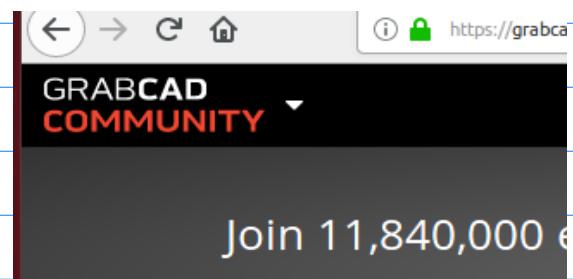
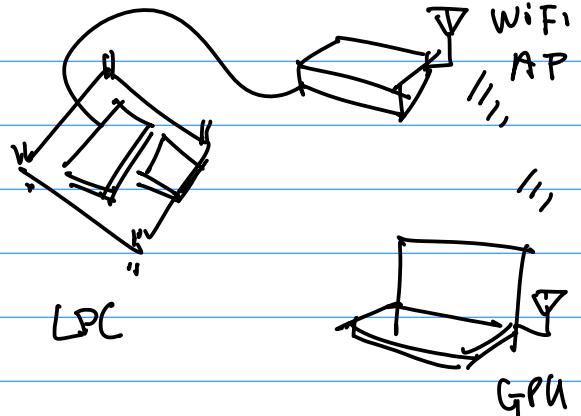
2. 5% Bonus for Using/Impacting Real 3D CAD Data.

 FreeCAD
<https://www.freecad.org>

FreeCAD: Your own 3D parametric modeler

FreeCAD is an open-source parametric 3D modeler made primarily to design real-life objects of any size. Parametric modeling allows you to easily modify your ...

[Download](#) · [Installing on Linux](#) · [Your own 3D parametric modeler](#) · [User hub](#)



Reflectivity $\vec{R} = (K_{dr}, K_{dg}, K_{db})$

Red Green Blue

... (6)

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

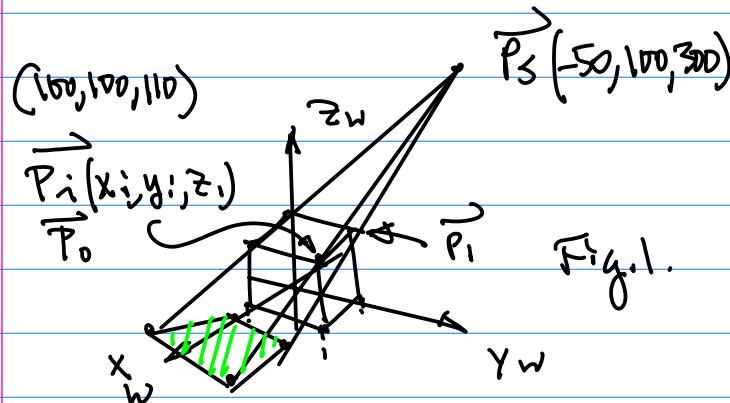


Fig.1.

define

Let's a Linear mapping
function

Let

$$(I_{d\min}, I_{off}) = (x_1, y_1) \dots (za)$$

$$(I_{d\max}, 255) = (x_2, y_2) \dots (zb)$$

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

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

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

which makes $I_d(x, y) < \delta$

Therefore, Suppose 8 bits per pixel

then,

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

$$y = b x + c \dots (4)$$

Now, Suppose we want to display diffuse reflection for a pixel location (x_i, y_i) Step 1. Use Eqn (1), pp 39, to find $I_{diff}(x_i, y_i)$

Step 2. Substitute

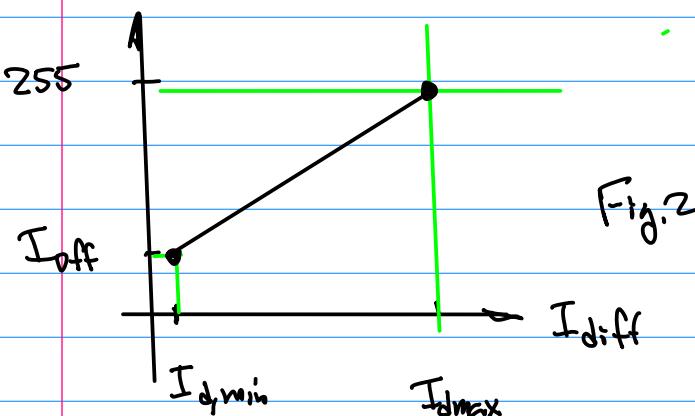
 $I_{diff}(x_i, y_i)$ into this
Eqn (4)

Fig.2

Where $I_{off} = Z_o$. $(I_{d\min}, I_{off})$ is a point on

Fig. 2.

 $(I_{d\max}, 255)$ is the other point

$$y = b x + c \quad |$$

$$x = I_{diff}(x_i, y_i)$$

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

Spring 2023

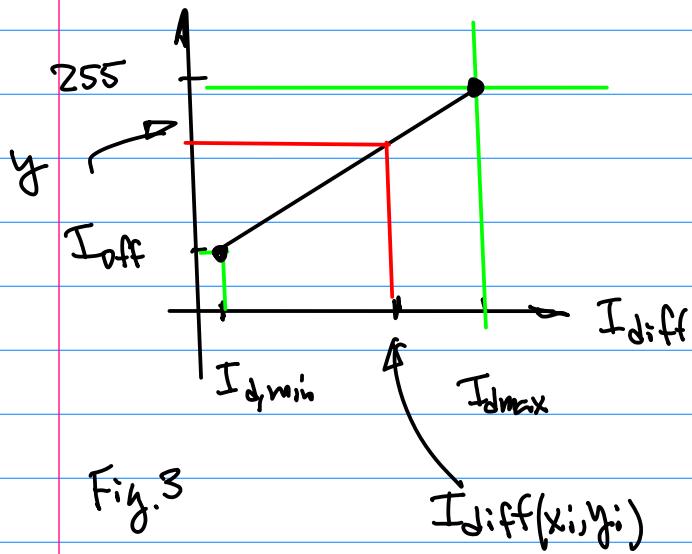


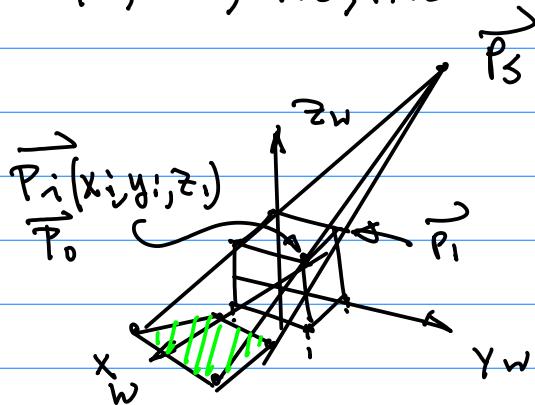
Fig. 3

y is the final intensity
Level for the color.

Now, we have Vertices
with Diffuse Reflection

Result:

$$\vec{P}_i, \vec{P}_{i+1}, \vec{P}_{i+2}, \vec{P}_{i+3}$$



Now, After Perspective Project

Fig. 4

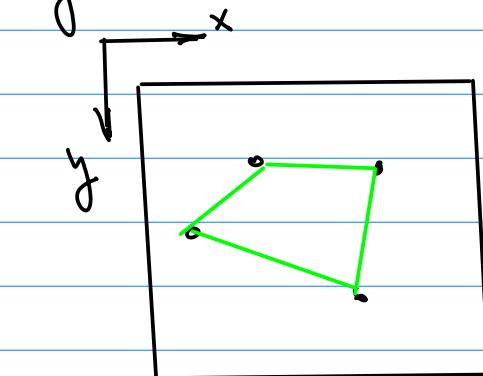


Fig. 5
Find Diffuse Reflection on the
Boundary Lines (Green).