

COMP1163 August 20 (Fri)

Organizational Meeting

1) Harry Li Email:

hmla.li@sjtu.edu

(650) 400-1116 Text

Office: M,W, 3:40-4:40pm.

Zoom ID + Passcode

is the same as

what you have today.

Lecture Zoom Link sent to
the class today.

Note: Homework, Projects
Announcements will be made
in Class, posted online as
github

CANVAS, Submission of homework
Projects will be
on CANVAS.

Text Books + References (optional)

a. Unity Tutorial, 3D Graphics

Game Dev. Engine

b. Other Optional Text Books —

Reference Only.

Programming Languages + Software IDE

1. Unity, Student or Personal

Edition. → Karting Game

2. Python for Graphics
Video, Version 3.6 or
higher.

Anaconda; Tool for
Python Programming →

3. C/C++ for 2D & 3D
Graphics, Videos.

4. C# for Interface to
Unity IDE.

→ 5. OpenGL Homework:
Installation of OpenGL,
In 2 weeks Sept. 2nd (Th)

→ 6. OpenGL Installation
of OpenGL. Homework:
Installation, and have it
ready by next week

Aug. 26 (Th) Before
4:00 pm..

→ 7. O.S. Ubuntu 18.04

Installation of Unity
By Aug. 26 (Th),
Before 4:00 pm..

2/

Grading Policy:

30% Projects/Homework etc.
30% midterm (ONE)
40% Final (Comprehensive)

"GAME"-Like Environment

{ Robotics
Self-Driving.

Conduct of the Class

- 1° Lecture
- 2° Show+Tell
- 3° Form A team, 2-3 person Team.

All homework, Coding have to be individual, however teamwork is encouraged, and be required

Projects, Homework: Assigned Projects.
(3 projects)

plus A-Semester-long project (Team)

a 2-3 person team;

b Proposal of A-Semester-long Project;

c Progress Report & Presentation During Class Show+Tell

d Final Presentation (PPT. Demo)

3 projects.

Project to Build 3D Animated Graphics.

Virtual Camera + Video

3D

Graphics + Video

a Scene View Window

b Hierarchy Window

August.26 (Th)

Topics 1° Software Development Tool

2° Vertex Graphics
2D Vertex Graphics.

Reference Link: [github/ahmedali](https://github.com/ahmedali)

Software Tool: First, Unity Up

By Friday

OpenGL Installation on your Machine

Example: Running Unity ,
"Karting" GAME

Start the Unity .

Step1. On the right hand UI. Interactive Tutorial Panel (Window)

Select/Go through 2 Tutorial First.Tutorial — play the Karting GAME

Step2. UI Editor

a Scene View Window

b Hierarchy Window

Comp1b3

3

Hierarchy: "Everything" defined in this Window,

$\stackrel{a}{=} \text{PAN}$;

$\stackrel{b}{=} \text{Zoom In/Out}$, $\stackrel{c}{=} \text{Orbit Movement}$
(Virtual Camera)

Use this platform to modify the "Karting" Game. Removal of Some/all

3D Objects

Re-Building 3D Scene. System
(3D World Coordinate)

2D Vector Definition of a Line

Segment

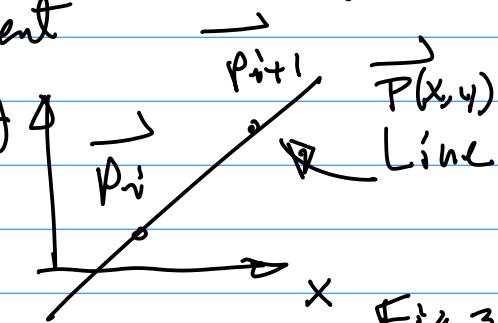


Fig.3

x-y Coordinate System

"Virtual Display" Coordinate System

Introduction to 2D Vector Graphics.

Dimensional
Description

Vertices (Vertex)
To Define Graphics
Pattern(s)

Primitive Graphics

2 pts to uniquely define a

line

\vec{P}_i, \vec{P}_{i+1}

Notation

\vec{P}_i Short Hand Notation

$P_i(x_i, y_i), x_i-, y_i-$ comp.

$P_i(x_i, y_i) = (x_i, y_i)$ for

Coding in C/C++, Python, ...

Vector \rightarrow Vertex \rightarrow Point



To Define A line

① Direction of the Line

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

\uparrow

\uparrow

Ending Pt. Starting Pt

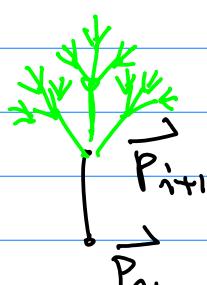


Fig.2

2D Vector
Graphics

Eqn(1), can be written as follows

$$\vec{d}(x_d, y_d) = \vec{P}_{i+1}(x_{i+1}, y_{i+1}) - \vec{P}_i(x_i, y_i) \\ \dots (1-a)$$

For Coding purpose,

$$x_d = x_{i+1} - x_i \quad (1-b)$$

$$y_d = y_{i+1} - y_i \quad (1-c)$$

Write C code for the directional vector in Eqn(1-b), (1-c)

Question: How to find the Ending pt from Eqn(2a) ?

if $\lambda = 1$

$$\begin{aligned} \vec{P}(x, y) &= \vec{P}_i(x_i, y_i) + 1 \cdot (\vec{P}_{i+1}(x_{i+1}, y_{i+1}) - \vec{P}_i(x_i, y_i)) \\ &= \vec{P}_i(x_i, y_i) + \vec{P}_{i+1}(x_{i+1}, y_{i+1}) - \vec{P}_i(x_i, y_i) \\ &= \vec{P}_{i+1}(x_{i+1}, y_{i+1}) \text{ Ending pt.} \end{aligned}$$

$$x_d[i] = x[i+1] - x[i] ; // \text{for } x\text{-Comp of the directional vector}$$

$$y_d[i] = y[i+1] - y[i] ; // \text{for } y\text{-Comp. of the directional Vector.} \\ \dots (1-d), (1-e)$$

(2) Need A pt to make an Unique Line

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

Where λ is scalar

Physical meaning: $\vec{P}(x, y)$ Any pt. on the Line

$\vec{P}_i(x_i, y_i)$ A given pt (Known) on this Line

$\vec{d}(x, y)$, A directional vector of the Line

Let $\lambda = 0$, $\vec{P}(x, y) = \vec{P}_i(x_i, y_i)$ starting pt.

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

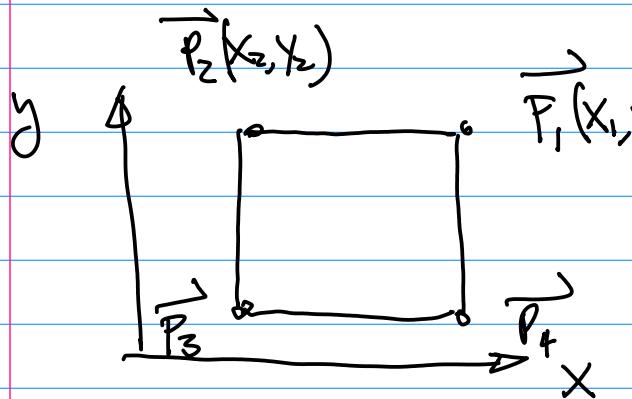
ScreenSaver a collection of 2D

Rotating Patterns. (Squares)

Example: Using Eqn(2a) to Create
2D Rotating Squares as a Screen
Saver.

Step 1. Define 2 vectors (\vec{P}_i s)

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



$$\vec{P}_1(x_1, y_1) = (60, 60), \vec{P}_2(x_2, y_2) = (10, 60)$$

And to Define A Line in Parallel with \vec{P}_1 & \vec{P}_2 $(1-d), \frac{1}{2}(1-e)$

$$\vec{P}_3(x_3, y_3) = (10, 10), \vec{P}_4(x_4, y_4) = (60, 10)$$

Connect \vec{P}_2 to \vec{P}_3 , Similarly \vec{P}_1 to \vec{P}_4

Therefore, we have formed A Square

Line Equation for Line (Top Line)

$$\vec{P}(x, y) = \vec{P}_1(x_1, y_1) + \lambda (\vec{P}_2(x_2, y_2) - \vec{P}_1(x_1, y_1)) \dots (3a)$$

Line for $\vec{P}_2(x_2, y_2)$ and $\vec{P}_3(x_3, y_3)$

$$\vec{P}(x, y) = \vec{P}_2(x_2, y_2) + \lambda_2 (\vec{P}_3(x_3, y_3) - \vec{P}_2(x_2, y_2)) \dots (3b)$$

And for the other 2 Lines

$$\vec{P}(x, y) = \vec{P}_3(x_3, y_3) + \lambda_3 (\vec{P}_4(x_4, y_4) - \vec{P}_3(x_3, y_3)) \dots (3c)$$

And

$$\vec{P}(x, y) = \vec{P}_4(x_4, y_4) + \lambda_4 (\vec{P}_1(x_1, y_1) - \vec{P}_4(x_4, y_4)) \dots (3d)$$

These 4 equations define
the Boundary of the Square.

From Coding Aspect:

From Sample/Example

Eqn(3a) becomes

$$\begin{cases} x = x_1 + \lambda (x_2 - x_1) \\ y = y_1 + \lambda (y_2 - y_1) \end{cases} \dots (4a) \quad \dots (4b)$$

Define A buffer for x,

And a buffer for y.

Each x_1, y_1, x_2, y_2 are also

Therefore C/C++ Coding Implementation
for (h-a), (h-b) can be
done accordingly.

Homework: Install OpenGL on your
Machine By Next Lecture,
So we will use it for
Rotating Sphere implementation.



Sept 2nd (Th)

Topics: 1^o 2D Screen Saver
Implementation;

Ref: [github/kanalilj/OpenGL/](https://github.com/kanalilj/OpenGL/)

Homework: (To Be Submitted in
1 week) Submission 4:00 pm.

Sept. 9th (Th) (1 pt)

Visit homework Assignment
on OpenGL, Source code .CPP
has been posted.

Example: OpenGL CPP code

1^o Create A program header
template, Start your
Program with this
Unified template

- a. Program Name
- b. Coded by
- c. Date , d. Version
- e. Status (Debugging,
Release). f. Compilation
and Build; g.

Ref. (URL)

```
b. j glBegin( );  
j glEnd( );  
glClear();
```

GL_POLYGON Keyword.

Vertex (pt)

Homework: GL_LINES

modify the Sample code
to draw a line with

$$\overrightarrow{P_1(x_1, y_1)} = \overrightarrow{P_1(x_1, y_1)} = (\overline{50}, \overline{50})$$

$$\overrightarrow{P_2(x_2, y_2)} = \overrightarrow{P_2(x_2, y_2)} = (60, 100)$$

Compile your program, run it.
E-mail your Screen Capture,
or 5 seconds Video Clips.

Submission
in e-mail,
Before Sept 9
4:10pm.
(No point)

7.

Create Rotating Squares for
a Screen Saver.

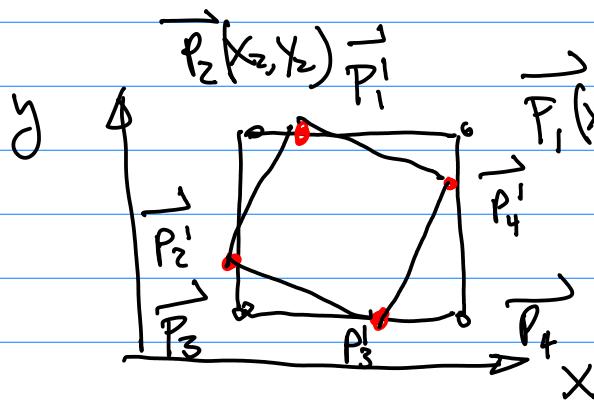


Fig 1. Note: $\vec{P_1}, \vec{P_2}, \dots, \vec{P_4}$ are
defined in A Counter Clock wise
direction (Later in 3D Graphics
we will do Hidden Line/Surface
Removal)

Repeat the Same Process, however
with New set of Points, $\vec{P_1}', \vec{P_2}',$
 $\vec{P_3}', \vec{P_4}'$
Continue this process, we have:

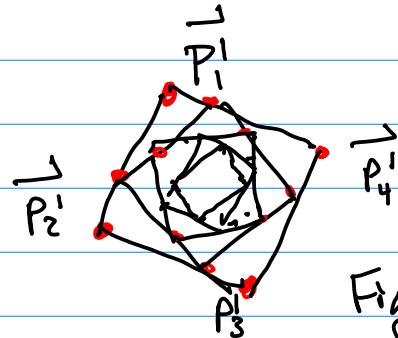


Fig. 2

To generalize this process,
we introduce a superscript
 j as follows,
From Eqn(za)

From Eqn(za),

$$\vec{P}(x, y) = \vec{P}_1(x_1, y_1) + \lambda (\vec{P}_2(x_2, y_2) - \vec{P}_1(x_1, y_1))$$

Let $\lambda = 0.8$ To S

$$\vec{P}(x, y) = \vec{P}_1(x_1, y_1) + \lambda (\vec{P}_2(x_2, y_2) - \vec{P}_1(x_1, y_1)) \quad \dots (za)$$

If $\lambda = 0.8$, \vec{P} is 80% pt on the line
formed by \vec{P}_1 & \vec{P}_2 ;

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

becomes

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

The Above Equation can be written in Explicit form (x - Comp, y - Comp)

for x -Comp.

$$x_i^{j+1} = x_i^j + \lambda (x_{i+1}^j - x_i^j) \dots (2a)$$

$$y_i^{j+1} = y_i^j + \lambda (y_{i+1}^j - y_i^j) \dots (2b)$$

C/C++ Code

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

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

Sample Code Example: [github/hualili/OpenCV.../1-line.cpp](https://github.com/hualili/OpenCV.../1-line.cpp).

https://github.com/hualili/opencv/blob/master/ComputerGraphics_AR/F2018/1_line.c

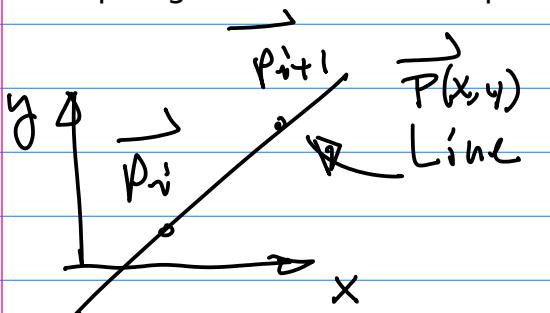


Fig.4

```

1  ****
2  * Program: line.c  Coded by: Harry Li
3  * Version: x1.0;  status: tested;
4  * Compile and build:
5  * gcc main.cpp -o main.o -lGL -lGLU -lglut
6  * Date: Jun 5, 2014
7  * Purpose: Graphics Demo.
8  ****
9  #include<GL/glut.h>
10 #include<stdio.h>
```

① Header

② Libraries

11 void mydisplay()
12 {
13 float p1x=1.0f, p1y=1.0f; //the window coordinates (-1.0, 1.0)
14 float p2x=-1.0f, p2y=-1.0f;

(3) $\vec{P}_1(x_1, y_1), \vec{P}_2(x_2, y_2), \vec{P}_1(x_1, y_1) = (1, 1), \vec{P}_2 = (-1, -1)$

```
15 glClear(GL_COLOR_BUFFER_BIT);
16 glLoadIdentity();
```

(4)

Note: House Keeping for 2D Graphics

(5) { glBegin();
 glEnd(); } GL-LINES

```
17 glBegin(GL_LINES);
18 glVertex2f(p1x, p1y);
19 glVertex2f(p2x, p2y);
20 glEnd();
```

$glVertex2f(x, y) \rightarrow \overrightarrow{P_i}(x_i, y_i) = (x_i, y_i)$

Note: In your homework, please
2D Sample code,

Sept. 9 (Th)

1. Show-And-Tell
2. Today's Topics: 2D Vector Graphics for Screen Saver Application, Squares (Rotating Pattern), Trees

Show+Tell Ken, Bill.

Patrick, Anh

Homework: Implementation of Rotating Squares Based on Eqn (a)-(b), or pp 8.

The A week from Today.

Show+Tell

Optional Homework: Write a script CS (C#) to control your Design.
~ 2 weeks

Example: Creating a Tree By 2D vector graphics.

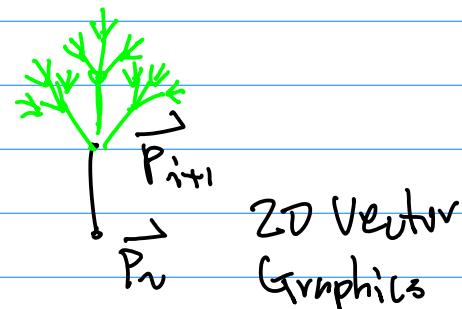


Fig. 1

2D Vector Graphics

- Note:
- 1° The levels of Iteration should be at least 7 or higher;
 - 2° Random Function Generator to Allow Each tree to be placed Random Locations. at rand();
 - 3° Python Version Dremel for the implementation is encouraged.

Description of the Algorithm:

1. You Design the length of A Tree Trunk By 2D Vectors. P_0, P_{i+1} , P_i : Starting pt; P_{i+1} Ending pt.

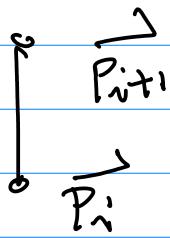


Fig. 2

$$\vec{P}_i(0, -10) \quad \vec{P}_{i+1}(0, 0)$$

--- (x)

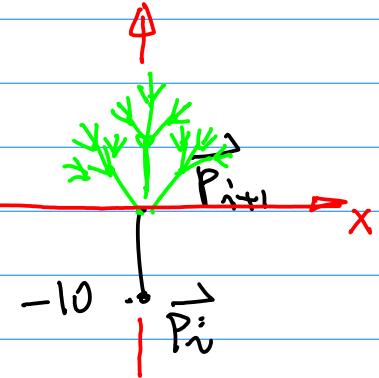


Fig. 4

2. Generate Another (Next Level)

Tree Branch \rightarrow Main Branch(Same Direction) as its
previous level;

a. Same direction
b. Reduction By 20%

From Eqn.(1a), (1b),
with the given condition (x),
we have

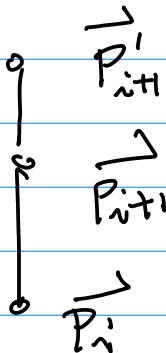


Fig. 3

$$\begin{aligned}
 \vec{P}'_{i+1} &= \vec{P}_{i+1} + \lambda (\vec{P}_{i+1} - \vec{P}_i) \\
 &= (0, 0) + \lambda ((0, 0) - (0, -10)) \\
 &= (0, 0) + \lambda (0, 10) \dots (2)
 \end{aligned}$$

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

Starting Pt.

Direction
Vector of the
previous level

To find New Ending pt \vec{P}'_{i+1}
Let $\lambda = 0.8$, Substitute λ
into the above equation

$$\vec{P}'_{i+1} = (0, 0) + 0.8(0, 10)$$

$$\begin{aligned}
 &= (0 + 0.8 \times 0, 0 + 0.8 \times 10) \\
 &= (0, 8)
 \end{aligned}$$

$$\begin{cases} x'_{i+1} = x_{i+1} + \lambda (x_{i+1} - x_i) \dots (1a) \\ y'_{i+1} = y_{i+1} + \lambda (y_{i+1} - y_i) \dots (1b) \end{cases}$$

3. Create the Other 2 Side Branches (Left Branch, Right Branch)

Define a pt $\vec{P}_i(x_i, y_i)$
Rotate this pt. $\vec{P}'_i(x'_i, y'_i)$
By α

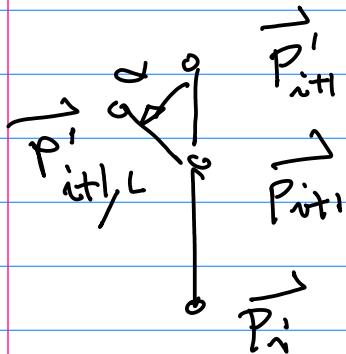


Fig 5.

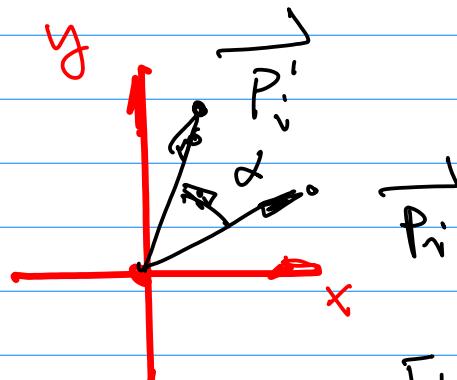


Fig b.

Left Branch: Rotating the main branch (at the same level) counter clockwise by α (Angle)

Denote the New Branch as
 $\vec{P}'_{i+1,L}$

Note: Need a Newer Math.
Formulation for this Rotation.

$$R_{3 \times 3} = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \dots (3)$$

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

Col. Vector

$$\vec{P}_i(x_i, y_i) = \begin{pmatrix} x_i \\ y_i \end{pmatrix}^T = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

$$\vec{P}_{i+1} = \begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix}$$

For Simplicity,

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

$$\vec{P}_{i+1}(x_{i+1}, y_{i+1}) = \begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix}$$

CMPE163

$A \cdot \vec{P}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$, Before the Rotation,

$\vec{P}_i \cdot P_i' = \begin{pmatrix} x_i' \\ y_i' \end{pmatrix}$, After the Rotation

?

Now, for the Branch
to the Right,
¹²

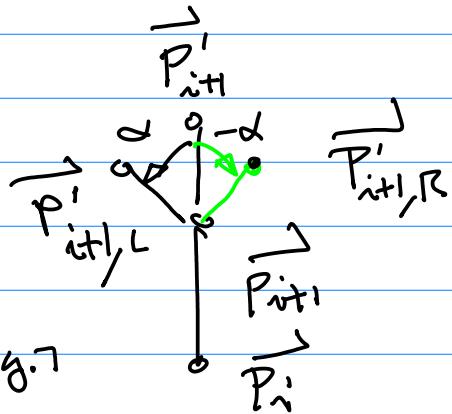


Fig. 7

After

$$\begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \dots (3b)$$

Before

$$\begin{pmatrix} x'_{i+1,R} \\ y'_{i+1,R} \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \quad \dots (5)$$

Therefore,

$$-\sin(-\alpha) = \sin\alpha$$

Rotation x)

Suppose $\alpha = 30^\circ$

$$\begin{cases} x'_i = x_i \cos\alpha - y_i \sin\alpha \dots (4a) \\ y'_i = x_i \sin\alpha + y_i \cos\alpha \dots (4b) \end{cases}$$

Note: 1° Positive α : α in Counter
Clockwise direction;

Negated α : Clockwise

2° The Rotation in (3b)

defines the Rotation

w.r.t the origin of the x-y Coordinate System.

$$\begin{cases} x'_{i+1,R} = \cos\alpha \cdot x_{i+1} + \sin\alpha \cdot y_{i+1} \\ y'_{i+1,R} = -\sin\alpha \cdot x_{i+1} + \cos\alpha \cdot y_{i+1} \end{cases} \dots (6a)$$

$$\begin{cases} x'_{i+1,R} = -\sin\alpha \cdot x_{i+1} + \cos\alpha \cdot y_{i+1} \\ y'_{i+1,R} = -\cos\alpha \cdot x_{i+1} - \sin\alpha \cdot y_{i+1} \end{cases} \dots (6b)$$

Team 1 { Christianha 1702,
Anh 3807,
Charles 042b
Ken 3381

Team 2 { Christoffer 7230
Patric 8001
Ammol 9654

Team 3. | Yong 423b
Zhonglin 6764

Semester Long - Project Proposal.

A Paragraph, 50 words, proposal.

Tech. Elements:

1. 3D Graphics
2. Interactive
3. Presentation mode — Script Features
4. Integrate Real/Live Video from
File(s) And/or from a Camera

Guidelines for Topics Selection

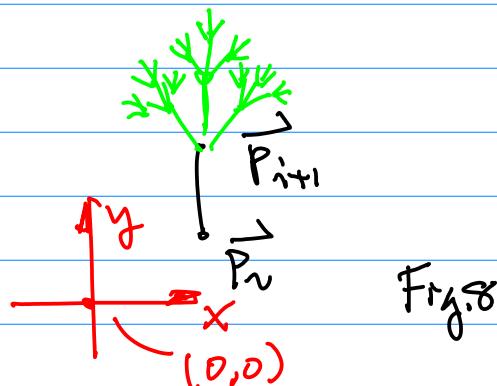
Enhance/Improve Productivity

Promote/Present Better Communication

By Sept. 23rd Submission to Email:
= zhua.li@sjtu.edu

Show + Tell

Note: For more general cases,



Sept 16 (Th)

Topics: 1° 2D Transforms

To Project Implementation
Trees/Forest

Note: Homework (Show+Tell)

Homework (Un-Official)

Rotating Squares Open GL

Reference: On git

github/finalli/OpenGL
Computer~ / F2018 / ~

2021F-b-... Homework.



(2 pts.)

Sept. and

Homework: Implementation of
Rotating Squares Based on
Euler (2d), on pp.8.

Example: On Window.

Note:

1. Always start your
program with a header;

a. Program Name:

Coded by:

Date:

Version:

Status:

Note: (Environment, OS
Compilation & Build)

2. Implementation.

Vector Formulation w/o
Rotation Matrix.

Note: $\sin x$ or $\cos x$

Taylor Expansion (Taylor Series)

$$f(x) = f(x_0) + \frac{f'(x)}{1!}(x-x_0) + \frac{f''(x)}{2!}(x-x_0)^2 + \dots + \frac{f^{(n)}(x)}{n!}(x-x_0)^n$$

Look-up table.



3. Change Polygons to Lines

2D vertex is better for this implementation;

4. $\lambda = 0.8$ to begin with, then when finish the entire program execution, modify λ , $\lambda = 0.95$, observe the difference patterns.

+ try $x=0.05$

5. Add Delay.

Frame Rate (Refresh Rate for Display).

30 FPS (Frames per Second)

$$f = 30 \text{ Hz}$$

$$T = \frac{1}{f} = 33.3 \text{ ms}$$

6. Keep one color at time for a set of all rotating squares

7. Create multiple sets of rotating squares, by defining "Anchor point" for each set. Change 2D vertex (x_i, y_i)

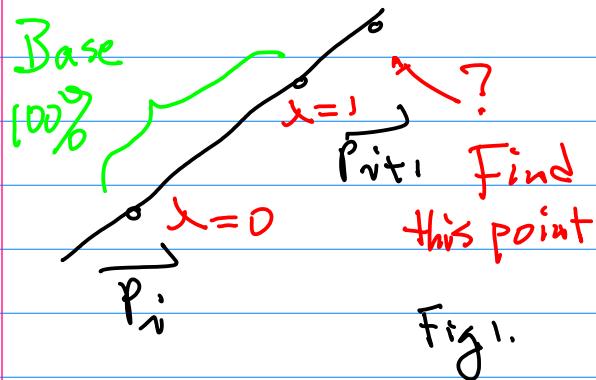
+ to

2D vertex $(x_i + x_0, y_i + y_0)$
where my anchor point is (x_0, y_0)

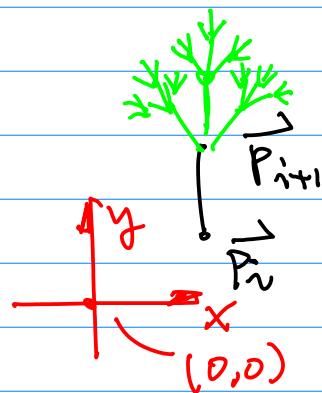
Later, you can randomize the anchor point.

8. Modify the Line width

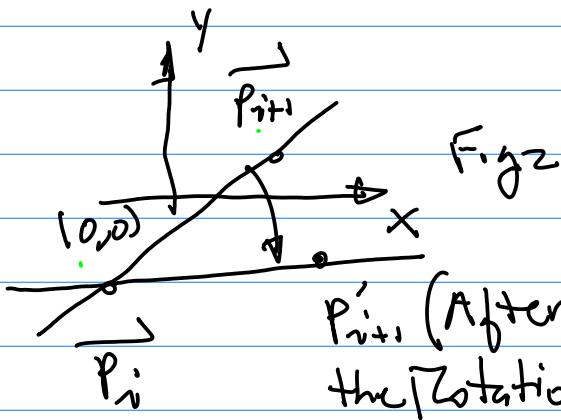
Note for the Homework, 1.3



Now, Consider Question 2



To Rotate the Tree Branches, we want to make sure the rotation is performed w.r.t the origin.



Translate \vec{P}_{i+1} to the origin,

\vec{P}'_{i+1} (After Translation matrix T to move \vec{P}_{i+1} to the origin) ... (1)

Pre-processing : to move \vec{P}_i to the origin $(0,0)$

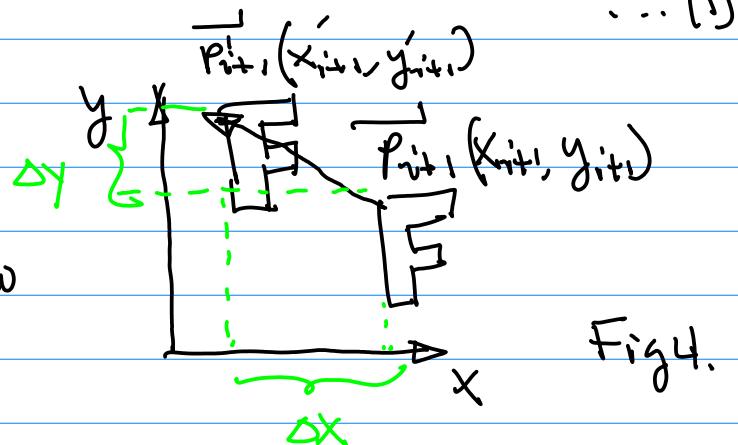
$$x'_{i+1} = x_{i+1} + \Delta x \quad (\text{After}) \quad (\text{Before})$$

then, Rotation;

Post-Processing.

Sept. 23rd.

Composition:
Topic: Trees & 2D Transformation
Homework (Due A week)



Composition of 2D Transforms

$$y'_{i+1} = y_{i+1} + \Delta y \quad \dots (1b)$$

$$\begin{pmatrix} x_{i+1}' \\ y_{i+1}' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta X \\ 0 & 1 & \Delta Y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \dots (z)$$

Before)

After

Post Processing to undo the
Pre-processing.

$$T^{-1} = \begin{pmatrix} 1 & 0 & -\Delta X \\ 0 & 1 & -\Delta Y \\ 0 & 0 & 1 \end{pmatrix} \dots (4)$$

$$\Delta X = -10$$

$$\Delta Y = -10$$



Substitute the given condition,
we have

$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 10 \\ 10 \\ 1 \end{pmatrix}$$

verified.

$$T^{-1} = \begin{pmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{pmatrix}$$

Then, do this for $\vec{P}_2(10, 20)$, find \vec{P}'_2
then, Rotation

$$R = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\alpha = 30^\circ$$

Integrate this into the
Composition matrices, we
have

$$T^{-1} R T = \quad \text{1st applied to the input vector}$$

$$\begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 10 \\ 20 \\ 1 \end{pmatrix}$$

... (3)

$$\begin{pmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{pmatrix}$$

... (5)

Then Post Processing,



$$\begin{pmatrix} \cos\alpha & -\sin\alpha & \Delta x \cos\alpha - \Delta y \sin\alpha \\ \sin\alpha & \cos\alpha & \Delta x \sin\alpha + \Delta y \cos\alpha \\ 0 & 0 & 1 \end{pmatrix}$$

Step 2.

$$T^{-1} RT =$$

$$T^{-1} \begin{pmatrix} \cos\alpha & -\sin\alpha & \Delta x \cos\alpha - \Delta y \sin\alpha \\ \sin\alpha & \cos\alpha & \Delta x \sin\alpha + \Delta y \cos\alpha \\ 0 & 0 & 1 \end{pmatrix}$$

Sept. 30 (Thu)

Today's Topics

1° Unity. 2° 3D Computer

Graphics.

Note:

1. New Homework Due A week from

Today

Continue the discussion of Eqn(5).

Step 1. To Take care of $R T$

$$= \begin{pmatrix} 1 & 0 & -\Delta x \\ 0 & 1 & -\Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\alpha & -\sin\alpha & \Delta x \cos\alpha - \Delta y \sin\alpha \\ \sin\alpha & \cos\alpha & \Delta x \sin\alpha + \Delta y \cos\alpha \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\alpha & -\sin\alpha & \Delta x \cos\alpha - \Delta y \sin\alpha - \Delta x \\ \sin\alpha & \cos\alpha & \Delta x \cos\alpha + \Delta y \sin\alpha - \Delta y \\ 0 & 0 & 1 \end{pmatrix}$$

... (b)

Hence,

$$\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} \cos\alpha \cdot 1 & -\sin\alpha \cdot 1 & \Delta x \cos\alpha - \Delta y \sin\alpha \\ \sin\alpha \cdot 1 & \cos\alpha \cdot 1 & \Delta x \sin\alpha + \Delta y \cos\alpha \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & \Delta x \cos\alpha - \Delta y \sin\alpha - \Delta x \\ \sin\alpha & \cos\alpha & \Delta x \cos\alpha + \Delta y \sin\alpha - \Delta y \\ 0 & 0 & 1 \end{pmatrix}$$

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

Therefore,

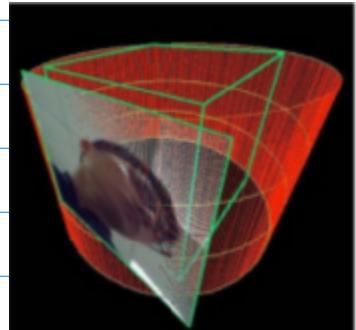
$$\begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha - \sin\alpha \cos\delta & \cos\alpha - \delta \sin\alpha & -\delta x_i \\ \sin\alpha \cos\delta & \cos\alpha + \sin\alpha \cos\delta & -\delta y_i \\ 0 & 0 & \sin\delta & \cos\delta \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

↓
Before

After

C/C++ Coding, we need

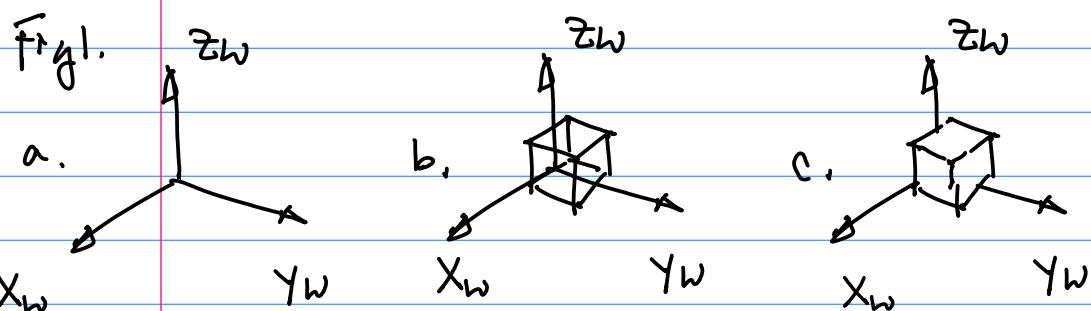
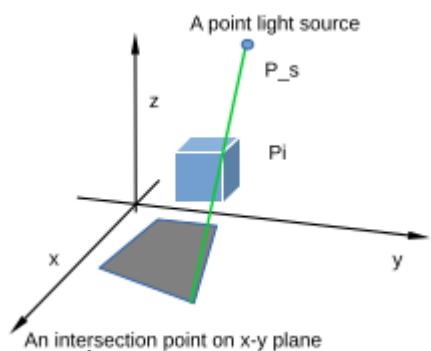
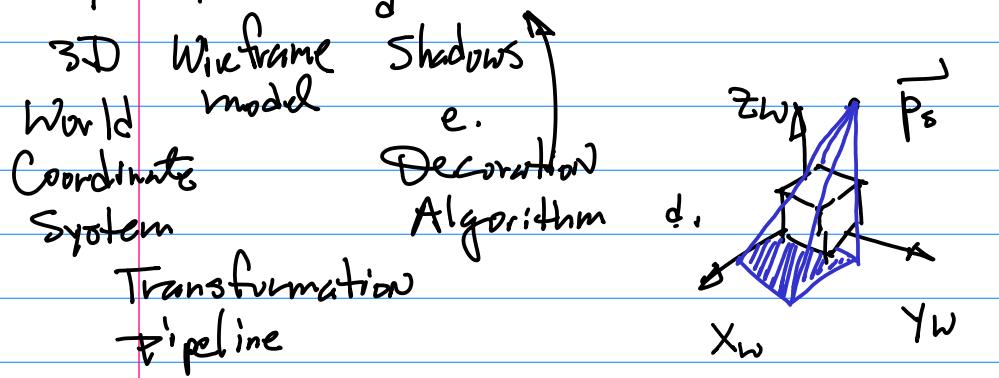
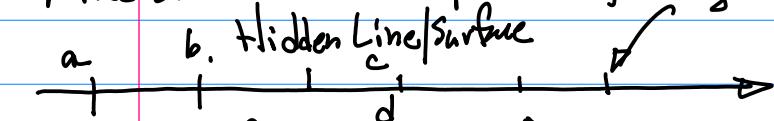
e. Decoration
Texture Mapping



$$\begin{cases} x'_i = \cos\alpha \cdot x_i - \sin\alpha \cdot y_i + \delta x \cos\delta - \delta y \sin\alpha - \delta x \\ y'_i = \sin\alpha \cdot x_i + \cos\alpha \cdot y_i + \delta x \sin\alpha + \delta y \cos\alpha - \delta y \end{cases}$$

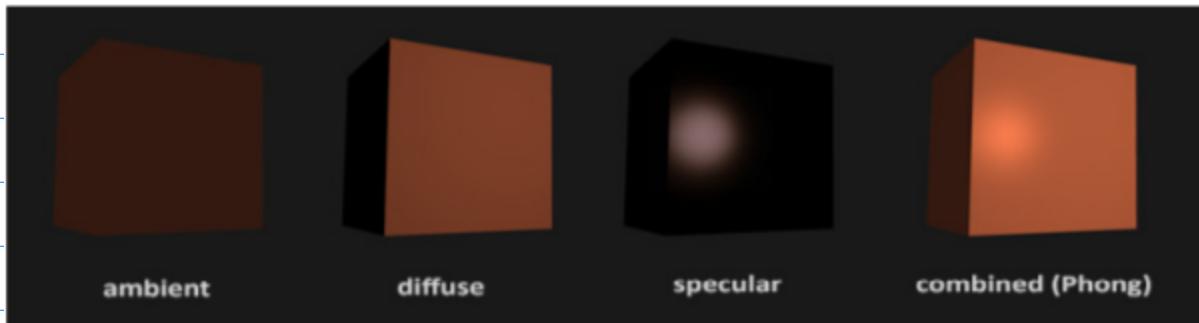
$$\begin{cases} x'_i = \cos\alpha \cdot x_i - \sin\alpha \cdot y_i + \delta x \cos\delta - \delta y \sin\alpha - \delta x \dots (7a) \\ y'_i = \sin\alpha \cdot x_i + \cos\alpha \cdot y_i + \delta x \sin\alpha + \delta y \cos\alpha - \delta y \dots (7b) \end{cases}$$

Three Dimensional Graphics. f. Lighting models.



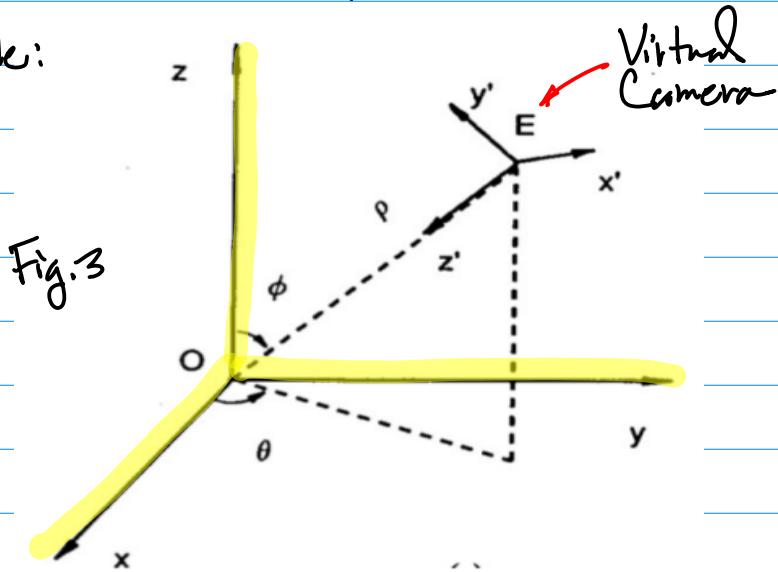
from H.Li's Book
manuscript.

f. Lighting models



World Coordinate System.

Example:



3D Transformation Pipelines

Step 1. Transform of a 3D Object from World Coordinate System to A Viewer Coordinate System.

Step 2. Project the Object from the Viewer Transform to 2D Display Screen
(Perspective Projection)

1° World Coordinate System (Yellow Highlight)

Subscript "w" —> World, $x_w - y_w - z_w$

2° Right-Hand System.

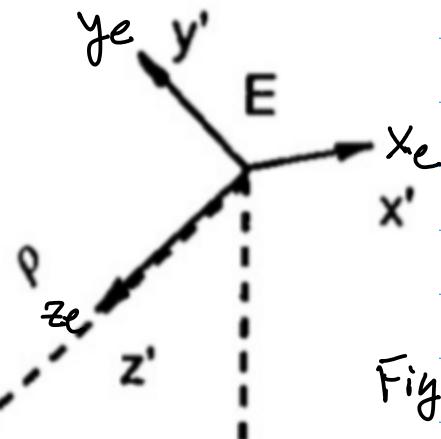
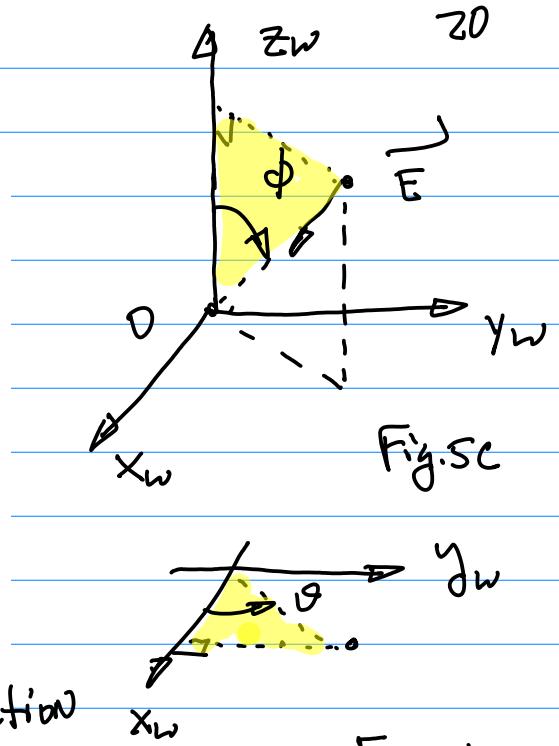
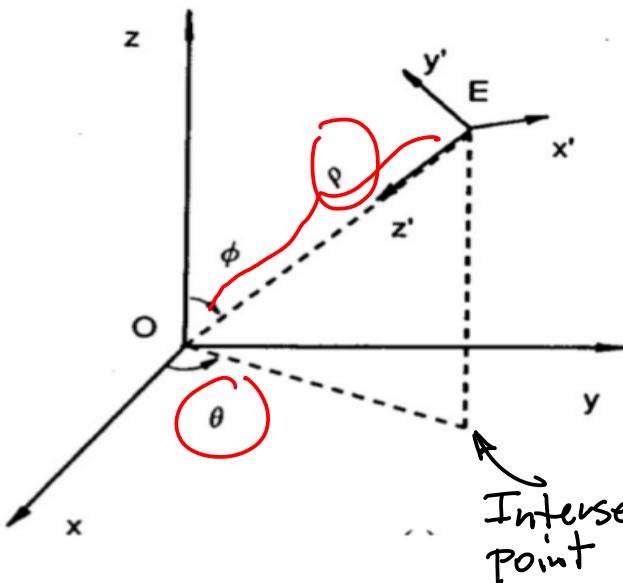
3° Define Everything in a World Coordinate System, including a Virtual Camera located at $E(x_e, y_e, z_e)$ 4° Viewer Coordinate System, a left hand system, $x_e - y_e - z_e'$, as in Fig 4.

Fig. 4

CmpE163

Example:

Figs



Figs

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

World-to-Viewer
Transform

θ (Theta): Angle on x_w - y_w plane

... (1)

Formed by 2 vectors, 1st Vector

x_w -axis,

Project \vec{E} onto x_w - y_w plane,

Link the Intersection pt on the

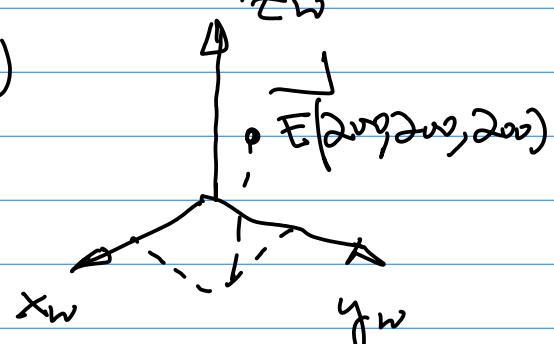
x_w - y_w plane to the Origin $(0,0,0)$

to form the 2nd vector.

ϕ (phi): Angle Between z_w and $\vec{E}O$ vector

ρ (rho): Distance, from \vec{E} to the
Origin of the World Coordinate
System.

Given the $\vec{E}(200, 200, 200)$,
find World
To Viewer Transform
By Eqn(1).



Sol. From Eqn(1)

We need to find $\theta, \cos\theta, \sin\theta$; $\phi, \sin\phi, \cos\phi$; ρ .
2. find New point $P_i' (x'_i, y'_i, z'_i)$ After the Transformation.

Oct. 7 (Th)

Tupils: 1^o 3D Transformation Pipeline

2^o Homework 5 Key's
Presentation of Homework 5.

Homework 6 (1 pt) One week
Submission online (CANVAS)

Example:

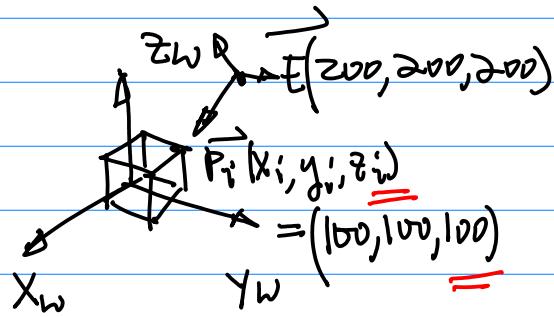


Fig 1.

$\vec{P}_i (x_i, y_i, z_i)$ Before the Transform
(in the world)

$\vec{P}_i' (x'_i, y'_i, z'_i)$ After the Transform.

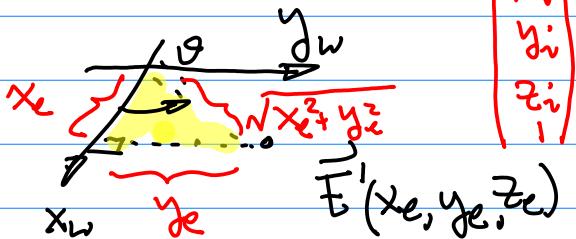
Find: 1. World-To-Viewer

Transformation matrix;

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

First, find $\sin\theta$

Fig. 2



$E'(x_e, y_e, z_e)$ is formed as an intersection point between a perpendicular line passing through $E(x_e, y_e, z_e)$ and X_w-Y_w plane

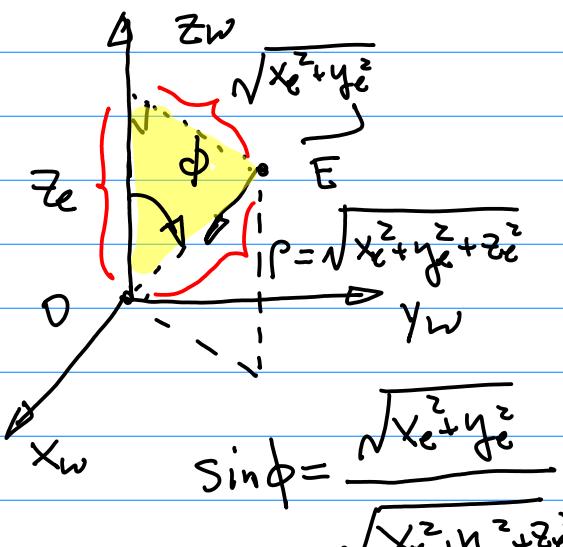
$$\text{Hence, } \sin\theta = \frac{y_e}{\sqrt{x_e^2 + y_e^2}} = \frac{200}{200\sqrt{2}} = \sqrt{2}/2$$

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

Similarly, find

$\sin\phi$

Draw a line passing E' perpendicular to Z_w Axis to form a Triangle



$$= \frac{2\sqrt{2}\sqrt{2}}{2\sqrt{2}\sqrt{3}} = \sqrt{2}/3 = \sqrt{6}/3$$

$$\cos \phi = \frac{-z_e}{\sqrt{x_e^2 + y_e^2 + z_e^2}} = \frac{-700}{200\sqrt{3}}$$

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

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

$$= \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ -\frac{\sqrt{6}}{2} \cdot \frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{2} \cdot \frac{\sqrt{2}}{2} & \sqrt{6}/3 & 0 \\ -\frac{\sqrt{6}}{2} \cdot \frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{2} \cdot \frac{\sqrt{2}}{2} & -\frac{\sqrt{3}}{3} & 200\sqrt{3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ -\frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{6} & \sqrt{6}/3 & 0 \\ -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & 200\sqrt{3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then, the point \vec{P}_i after the transformation is \vec{P}'_i , given below

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ -\frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{6} & \sqrt{6}/3 & 0 \\ -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & 200\sqrt{3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}$$

From Fig 1
Part 1

where $\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = \begin{bmatrix} 100 \\ 100 \\ 100 \\ 1 \end{bmatrix}$

$$x'_i = -\frac{\sqrt{2}}{2} \cdot \frac{100}{200} + \frac{\sqrt{2}}{2} \cdot \frac{100}{200} = 0$$

$$y'_i = -\frac{\sqrt{6}}{6} \cdot \frac{100}{200} - \frac{\sqrt{6}}{6} \cdot \frac{100}{200} + \frac{\sqrt{6}}{3} \cdot \frac{100}{200}$$

$$= -\frac{\sqrt{6}}{3} \cdot \frac{100}{200} + \frac{\sqrt{6}}{3} \cdot \frac{100}{200} = 0$$

$$z'_i = -\frac{\sqrt{3}}{3} \cdot \frac{100}{200} - \frac{\sqrt{3}}{3} \cdot \frac{100}{200} - \frac{\sqrt{3}}{3} \cdot \frac{100}{200} + 200\sqrt{3}$$

$$= -\frac{\sqrt{3}}{3} \cdot \frac{100}{200} + 200\sqrt{3} = 100\sqrt{3}$$

Check physical meaning!

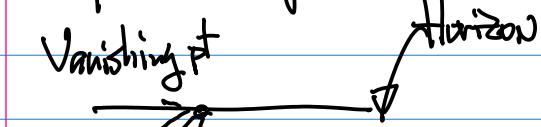
C/C++ Implementation. From Eqn(1)

$$\left. \begin{aligned} x'_i &= -\sin \theta \cdot x_i + \cos \theta \cdot y_i \\ y'_i &= -\cos \phi \cos \theta \cdot x_i - \cos \phi \sin \theta \cdot y_i + \sin \phi \cdot z_i \\ z'_i &= -\sin \phi \cos \theta \cdot x_i - \sin \phi \sin \theta \cdot y_i \\ &\quad - \cos \phi \cdot z_i + \rho \end{aligned} \right\} \dots (2)$$

163

2nd Step of Transformation Pipeline:

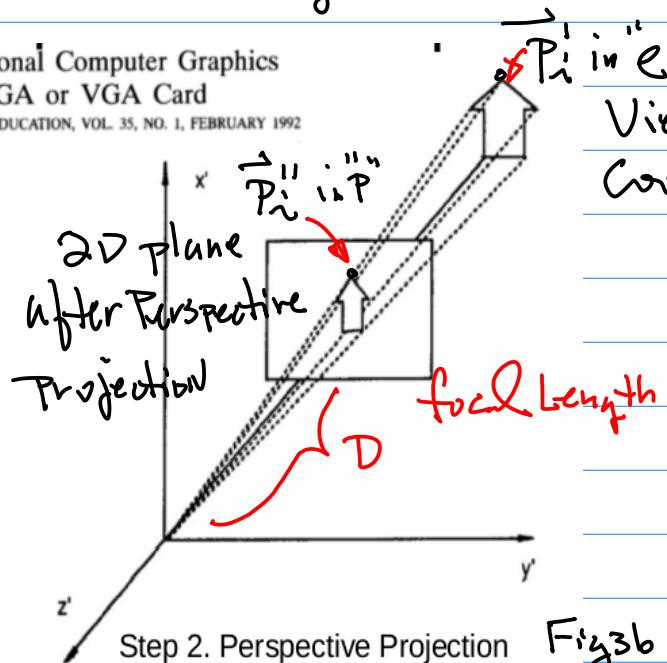
Perspective Projection



Depth Perception

Fig 3a

onal Computer Graphics
GA or VGA Card
EDUCATION, VOL. 35, NO. 1, FEBRUARY 1992



Step 2. Perspective Projection

Fig 3b

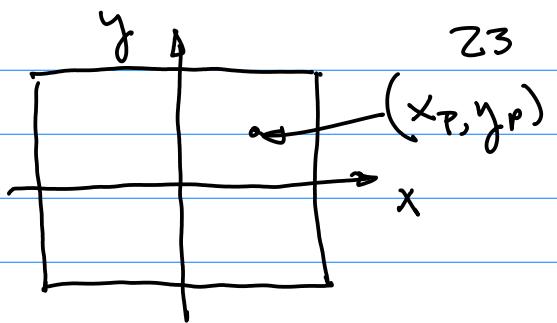
$$x_p = x_e \left(\frac{D}{z_e} \right)$$

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

... (3)

where Subscript "e" for Viewer Coordinate System, "p" for Perspective Projection

23



Example: Given a point \vec{P}_i' (after World to Viewer Transform), Compute Perspective projection

Coordinates S_D

(1) Assume $D=20$ from Eqn (3).

$$x_i'' = x_i' \frac{D}{z_i'} \Big|_{D=20} = \frac{20}{z_i'} \cdot x_i'$$

$$= \frac{20}{100\sqrt{3}} x_i' \Big|_{x_i'=0} = 0$$

$$y_i'' = y_i' \frac{D}{z_i'} = \frac{20}{z_i'} y_i'$$

$$= \frac{20}{100\sqrt{3}} \cdot 0 = 0$$

C/C++ Implementation:

$$\begin{cases} x_i'' = x_i' D / z_i' \\ y_i'' = y_i' D / z_i' \end{cases}$$

Oct.14 (Th)

Note: 1^o Homework Submission

Migrate All Submissions to CANVAS.

(1) Put a list of all the homeworks publish it on CANVAS;

(2) Submit All your homework in One zip. to CANVAS

One week from Today.

Oct 21st.

Midterm Exam: In 2 weeks

Oct. 28th.

Today's Topics :

1^o Examples of (World-To-Viewer, Perspective Projection) Transformation Pipelines.2^o Shadows

Example: Step1 for Homework3.

3. Implement world coordinate system and draw it on the screen.

Submission requirements:

- (1) source code;
- (2) compiled binary;
- (3) 5 second video clips;
- (4) put all in a zip file with the following naming convention:
FirstName-LastName-nameOfProject-4digitsID-YY-MM-DD.zip
for example:
Harry-Li-3Dworld-1116-2021-9-30.zip.

Design Data set, e.g. a collection of Vectors, $\vec{P}_0, \vec{P}_1, \dots, \vec{P}_i$ to define the

World Coordinate System

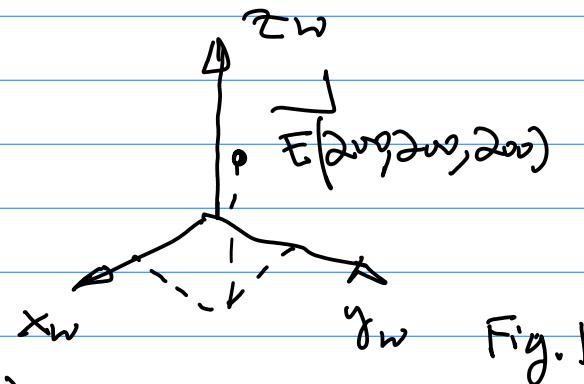


Fig. 1

 $\vec{P}_i(x_i, y_i)$ to cover all axis

For 2 pts for the x-axis

$$\begin{aligned} \text{Origin } \vec{P}_0(x_0, y_0, z_0) &= (0, 0, 0) \\ \vec{P}_1(x_1, y_1, z_1) &= (50, 0, 0) \end{aligned}$$

C/C++ for x-axis: Group \vec{P}_0, \vec{P}_1 togetherfor y_w -axis, 2 pts,

$$\vec{P}_0(x_0, y_0, z_0) = (0, 0, 0)$$

and

$$\vec{P}_2(x_2, y_2, z_2) = (0, 50, 0)$$

for z_w -axis, 2 pts

$$\vec{P}_0(x_0, y_0, z_0) = (0, 0, 0)$$

and

$$\vec{P}_3(x_3, y_3, z_3) = (0, 0, 50)$$

Step 2. Design/Define a Virtual Camera Location

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

Step 3. First step of the transformation pipeline,
e.g. World To Viewer
Transform From Eqn(1) pp22.

$$T_{w2v} = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ -\frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{6} & \frac{\sqrt{6}}{3} & 0 \\ -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & 2\sqrt{6}\sqrt{3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

Now, for all the vectors,
 $\vec{P}_0, \vec{P}_1, \dots, \vec{P}_3$

for \vec{P}_0 , we have

Find $\sin\theta, \cos\theta, \sin\phi, \cos\phi, \rho$
From Eqn(1), we can derive 3 Equations as follows

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ -\frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{6} & \frac{\sqrt{6}}{3} & 0 \\ -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & 2\sqrt{6}\sqrt{3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$x'_0 = -\sin\theta x_0 + \cos\theta y_0$$

$$\left. \begin{array}{l} y'_0 = -\cos\phi\cos\theta x_0 - \cos\phi\sin\theta y_0 + \sin\phi z_0 \\ z'_0 = -\sin\phi\cos\theta x_0 - \sin\phi\sin\theta y_0 \\ \quad - \cos\phi z_0 + \rho \end{array} \right. \dots (2)$$

$$\rho = \sqrt{x_0^2 + y_0^2 + z_0^2}$$

$$\therefore \vec{P}_0 = (0, 0, 0)$$

for \vec{P}_1 ,

$$\begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ -\frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{6} & \frac{\sqrt{6}}{3} & 0 \\ -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & 2\sqrt{6}\sqrt{3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 50 \\ 0 \\ 0 \\ 1 \end{bmatrix} =$$

To write the Above Equation as

in C Code,

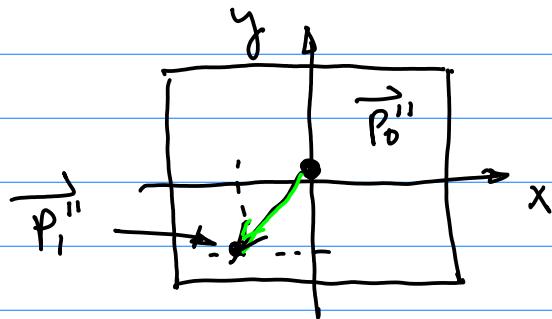
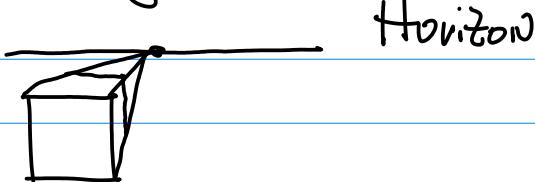
$$x_prim[i] = -\sin(\theta) * x[i] + \cos(\theta) * y[i];$$

$$\begin{bmatrix} -\frac{\sqrt{2}}{2} \cdot 50 \\ -\frac{\sqrt{6}}{6} \cdot 50 \\ -\frac{\sqrt{3}}{3} \cdot 50 + 2\sqrt{6}\sqrt{3} \end{bmatrix} = \begin{bmatrix} -25\sqrt{2} \\ -\frac{25\sqrt{6}}{3} \\ -\frac{\sqrt{3}}{3} \cdot 50 + 2\sqrt{6}\sqrt{3} \end{bmatrix} \dots (3)$$

You can Compute for \vec{P}_3 (y_w -axis)
 \vec{P}_3 (z_w -axis)

Background on Perspective Projection

Vanishing Point



Next Step: Perspective Projection

Step 4: Perspective Projection

$$\left\{ \begin{array}{l} x_i'' = \left(\frac{D}{Z_i'} \right) x_i' \quad \dots (4a) \end{array} \right.$$

$$\left\{ \begin{array}{l} y_i'' = \left(\frac{D}{Z_i'} \right) y_i' \quad \dots (4b) \end{array} \right.$$

Since $\theta = 20^\circ$.For $i=0$, \vec{P}_0

$$\left\{ \begin{array}{l} x_0'' = \frac{D}{Z_0'} \cdot x_0' = 0 \quad \dots (5a) \end{array} \right.$$

$$\left\{ \begin{array}{l} y_0'' = \frac{D}{Z_0'} \cdot y_0' = 0 \quad \dots (5b) \end{array} \right.$$

Therefore $\vec{P}_0'' = (0, 0)$ For $i=1$, (x -axis)

$$x_1'' = \frac{D}{Z_1'} \cdot x_1'$$

$$= \frac{D}{(\sqrt{3.50} + 20\sqrt{3})} (-25\sqrt{2}) \quad \dots (ba)$$

Similarly, for \vec{P}_1, \vec{P}_2

plot x_w -axis's on 2D Display Screen. Plot the result in Green.

Sample code for Transformation
Pixelizing
 $\vec{E}(x_e, y_e, z_e)$

```
28 float Xe = 100.0f;
29 float Ye = 100.0f;
30 float Ze = 100.0f;
```

https://github.com/hualili/opencv/blob/master/ComputerGraphics_AR/F2018/10-world-20190925.cpp

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

```
32 float Rho = sqrt(pow(Xe, 2) + pow(Ye, 2) + pow(Ze, 2));
33 float D_focal = 20.0f;
```

$$y_1'' = \frac{D}{Z_1'} \cdot y_1' = \frac{20}{\sqrt{\frac{2}{3}50 + 20\sqrt{3}}} \cdot (-25\sqrt{2})$$

Define $\vec{P}_i(x_i, y_i, z_i)$ struct in $x_w y_w z_w$ (World)

```
35 typedef struct {
36     float X[UpperBD];
37     float Y[UpperBD];
38     float Z[UpperBD];
39 } pworld;
```

```

41  typedef struct {
42      float X[UpperBD];
43      float Y[UpperBD];
44      float Z[UpperBD];
45  } pviewer;

```

$x-y-z$
ViewerCoordinate

```

47  typedef struct{
48      float X[UpperBD];
49      float Y[UpperBD];
50  } pperspective;

```

PointStruct
for Perspective Projection
2D points

Defining $x_w-y_w-z_w$ World Coordinate Below,

```

74      //define the x-y-z world coordinate
75      world.X[0] = 0.0;    world.Y[0] = 0.0;    world.Z[0] = 0.0;    // origin
76      world.X[1] = 50.0;   world.Y[1] = 0.0;   world.Z[1] = 0.0;    // x-axis
77      world.X[2] = 0.0;    world.Y[2] = 50.0;  world.Z[2] = 0.0;    // y-axis
78      world.X[3] = 0.0;    world.Y[3] = 0.0;  world.Z[3] = 50.0;   // z-axis

```

World To Viewer Transform : (3 Equation, one for Each x'_i, y'_i, z'_i)

```

184      for(int i = 0; i <= UpperBD; i++)
185      {
186          viewer.X[i] = -sPheta * world.X[i] + cPheta * world.Y[i];
187          viewer.Y[i] = -cPheta * cPhi * world.X[i]
188              - cPhi * sPheta * world.Y[i]
189              + sPhi * world.Z[i];
190          viewer.Z[i] = -sPhi * cPheta * world.X[i]
191              - sPhi * cPheta * world.Y[i]
192              -cPheta * world.Z[i] + Rho;
193      }

```

$$\begin{cases} x'_i = -\sin\theta x_i + \cos\theta \cdot y_i \\ y'_i = -\cos\phi \cos\theta x_i - \cos\phi \sin\theta y_i + \sin\phi z_i \\ z'_i = -\sin\phi \cos\theta x_i - \sin\phi \sin\theta y_i \\ \quad - \cos\phi z_i + p \end{cases}$$

For Perspective Projection, we have

```

195     for(int i = 0; i <= UpperBD; i++)
196     {
197         perspective.X[i] = D_focal * viewer.X[i] / viewer.Z[i] ;
198         perspective.Y[i] = D_focal * viewer.Y[i] / viewer.Z[i] ;

```

$$\begin{cases} x''_i = \left(\frac{D}{Z'_i} \right) x'_i \quad \dots (4a) \\ y''_i = \left(\frac{D}{Z'_i} \right) y'_i \quad \dots (4b) \end{cases}$$