

1/

CMPE163 August 20 (Fri) Organizational Meeting

1) Harry Li E-mail:

hna.li@sjsu.edu

(650) 400-1116 Text

Office: M.W. 3:40-4:40 PM.

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. OpenCV. Homework:
Installation of OpenCV,
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.

Grading Policy:

- 30% Projects, Homework etc.
- 30% Midterm (ONE)
- 40% Final (Comprehensive)

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 Project)

a 2-3 person team;

b Proposal of A - Semester-long Project;

c Progress Report & Presentation During class show + tell

d Final Presentation (P.P.T. Demo)

3 projects.

Project to Build 3D Animated Graphics.

Virtual Camera + Video

"Game"-Like Environment

- { Robotics
- { Self-Driving.

August 26 (Th)

Topics 1° Software Development Tool

2° Vector Graphics
2D Vector Graphics.

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

Software Tool: First, Unity up By Friday

OpenGL Installation on your Machine

Example: Running Unity, "Karting" Game

Start the Unity.

Step 1. On the right hand UI. Interactive

Tutorial Panel (Window)

Select/Go through 2 Tutorial

First Tutorial — play the Karting Game

Step 2. UI Editor

a Scene View Window

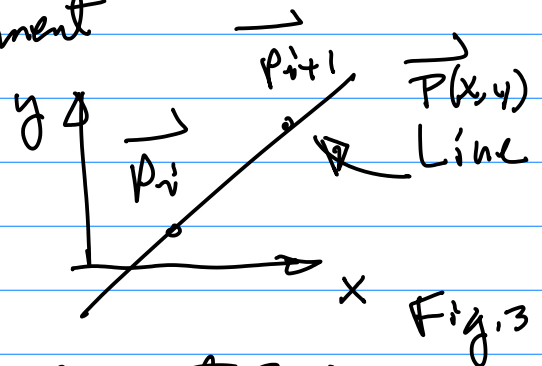
b Hierarchy Window
3D Graphics + Video

"Hierarchy": Everything Defined in this Window,

a PAN;
b Zoom In/Out, c Orbit Movement (Virtual Camera)

Use this platform to modify the "Karting" Game. Removal of Some/all
 3D Objects
 Re-Building 3D Scene.
 (3D World coordinate)

2D Vector Definition of a Line Segment



x-y Coordinate System

"Virtual" Display Coordinate System

Introduction to 2D Vector Graphics.

Dimensional Description

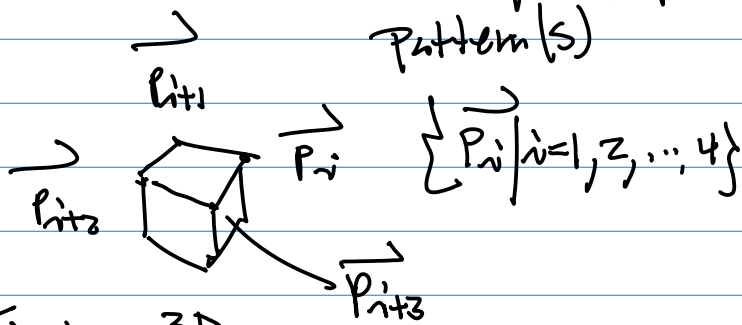


Fig.1. 3D

Vector \rightarrow Vertex \rightarrow Point

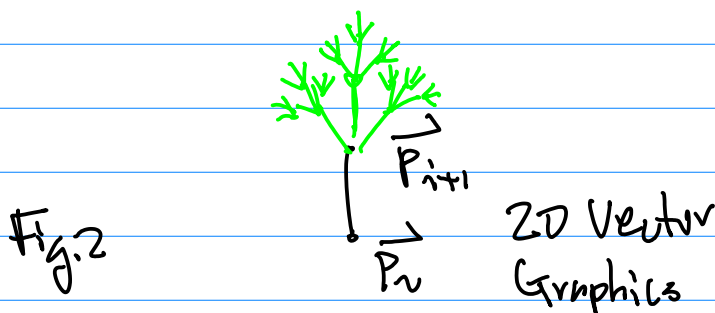


Fig.2

Primitive Graphics

2 pts to uniquely define a line
 \vec{P}_i, \vec{P}_{i+1}

Notation

\vec{P}_i Short Hand Notation

$\vec{P}_i(x_i, y_i)$, x_i - , y_i - comp.

$\vec{P}_i(x_i, y_i) = (x_i, y_i)$ for
 Coding in C/C++, Python, ...

To Define A line

(1) Direction of the Line

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

$\uparrow \quad \uparrow$
 Ending pt. Starting pt

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,

$$\begin{cases} x_d = x_{i+1} - x_i & (1-b) \\ y_d = y_{i+1} - y_i & (1-c) \end{cases}$$

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

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

if $\lambda = 1$

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

$$\begin{aligned} x_d[i] &= x[i+1] - x[i] ; // \text{for } x\text{-Comp of the directional vector} \\ y_d[i] &= y[i+1] - y[i] ; // \text{for } y\text{-Comp. of the directional vector.} \end{aligned} \dots (1-d), (1-e)$$

(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

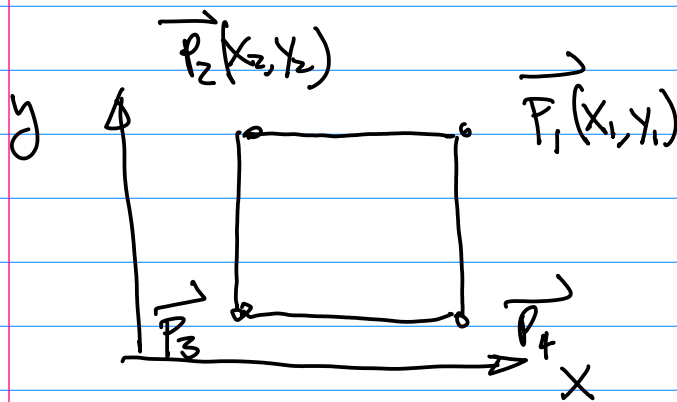
Let $\lambda = 0$, $\vec{P}(x, y) = \vec{P}_i(x_i, y_i)$ Starting pt.

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

Screen Saver a collection of 2D
Rotating Patterns. (Squares)

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

Define 2 vectors (pts)
Step 1. $\vec{P}_1(x_1, y_1)$, and $\vec{P}_{i+1}(x_{i+1}, y_{i+1})$



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

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.

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

From Coding Aspect:

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

From Sample/Example

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

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

Eqn (3a) becomes

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 $\vec{P}_3(x_3, y_3)$

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

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 (4-a), (4-b) can be done accordingly.

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



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

Example: OpenGL CPP code
1° 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. glBegin( );
   |
   |
   | glEnd( );
   |
   |
   | glClear( );
```

GL_POLYGON keyword.

Vertex (pt) ↓

Homework: GL_LINES

Modify the Sample code to draw a line with

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

$\overrightarrow{P_{1+1}(x_{1+1}, y_{1+1})} = \overrightarrow{P_2(x_2, y_2)} = (60, 100)$

Sept 2nd (Th)

Topics : 1° 2D Screen Saver Implementation;

Ref: github/finalidi/opengl/

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

Sept. 9th (Th) (1pt)

Compile your program, run it.
E-mail your Screen Capture, or 5 seconds Video Clips.
Submission in e-mail, Before Sept 4: 12pm.
(No point)

Create Rotating Squares for a Screen Saver.

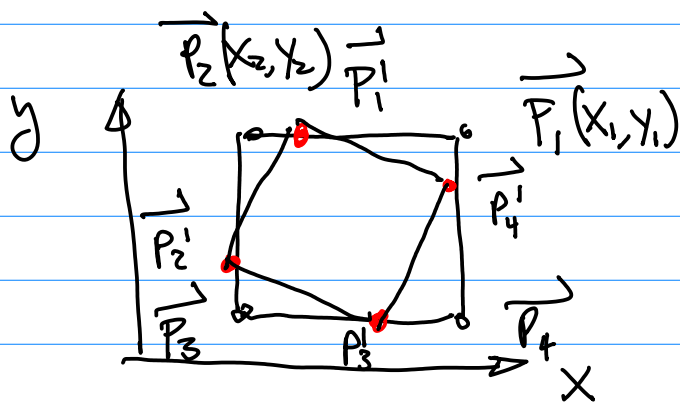


Fig. 1. Note: $\vec{P}_1, \vec{P}_2, \dots, \vec{P}_4$ are defined in A Counter Clock Wise direction (Later in 3D Graphics we will do Hidden Line/Surface Removal)

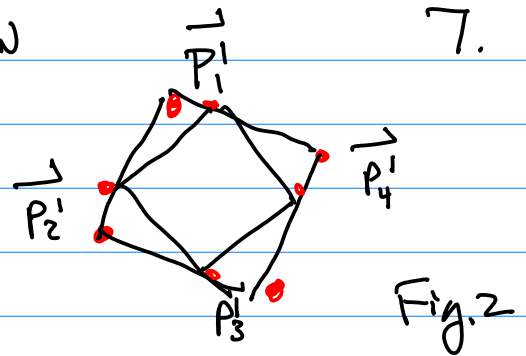
From Eqn (2-a),

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

Let $\lambda = 0.8$ tp. 5

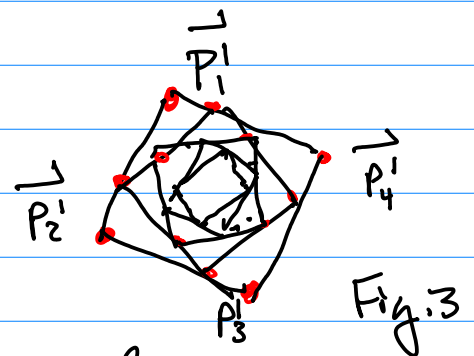
$$\vec{P}(x, y) = \vec{P}_1(x_1, y_1) + \lambda (\vec{P}_2(x_2, y_2) - \vec{P}_1(x_1, y_1)) \dots (3a)$$

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



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:



To generalize this process, we introduce a superscript j as follows,
From Eqn (2a)

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

becomes

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

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

For x-comp.

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

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

C/C++ Code

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

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

Sample code Example: [github/hualili/opencv ... /1_line.c](https://github.com/hualili/opencv/blob/master/ComputerGraphics_AR/F2018/1_line.c)

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

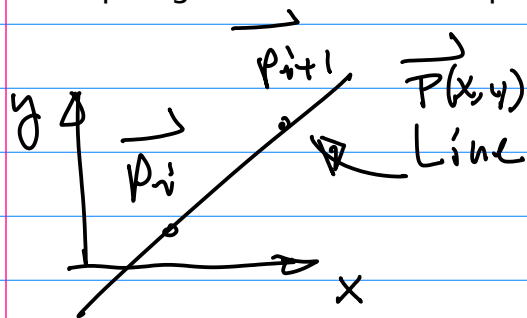


Fig.4

① Header

```

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>

```

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;

```

$$\vec{P}_1(x_1, y_1), \vec{P}_2(x_2, y_2), \vec{P}_3(x_3, y_3) = (1, 1), \vec{P}_2 = (-1, -1)$$


```
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();
```

$\text{glVertex2f}(x, y) \rightarrow \vec{P}_i(x_i, y_i) = (x_i, y_i)$

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

Example: Creating a Tree By 2D
vector graphics.

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, Bull.
Patrick, Anh

Homework: Implementation of
Rotating Squares Based on
Eqn (2b), on pp 8.

Due A week from Today.

Show+Tell

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

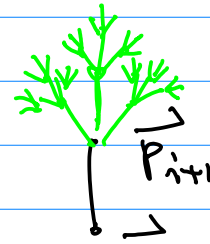


Fig. 1

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 $\text{rand}()$;
 - 3° Python Version OpenGL
for the Implementation is
encouraged.

Description of the Algorithm:

1. You Design the Length of A Tree
Trunk. By 2D Vectors. \vec{P}_i, \vec{P}_{i+1}
 \vec{P}_i : Starting pt; \vec{P}_{i+1} Ending pt.

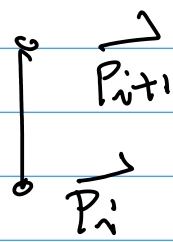


Fig. 2

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

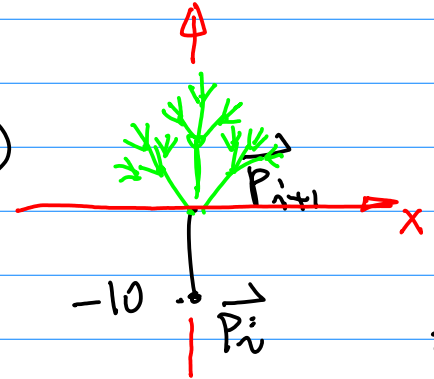


Fig 4

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 (*), we have

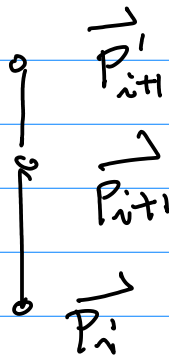


Fig. 3

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

$$= (0, 0) + \lambda ((0, 0) - (0, -10))$$

$$= (0, 0) + \lambda (0, 10) \quad \dots (2)$$

To find New Ending pt \vec{P}'_{i+1}

Let $\lambda = 0.8$, substitute λ into the above equation

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

$$= (0 + 0.8 \times 0, 0 + 0.8 \times 10)$$

$$= (0, 8)$$

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

Starting Pt.

Direction Vector of the previous level

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

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

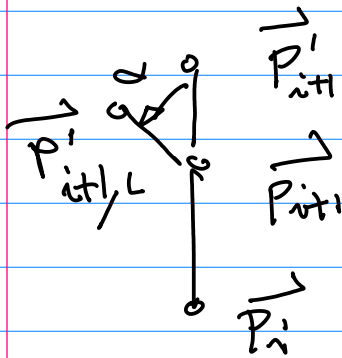


Fig 5.

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.

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

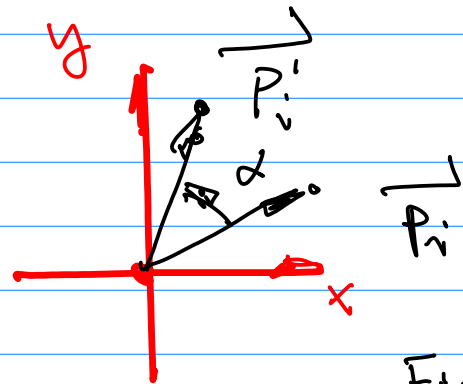


Fig 6.

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

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

Col. Vector

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

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

For Simplicity,

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

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

A Pt $\vec{P}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$, Before the Rotation,
 Pt. $\vec{P}'_i = \begin{pmatrix} x'_i \\ y'_i \end{pmatrix}$, After the Rotation

Now, for the Branch¹²
 to the Right,

After the Rotation = Before the Rotation,
 Hence, we need Rotation matrix
 Eqn (3),

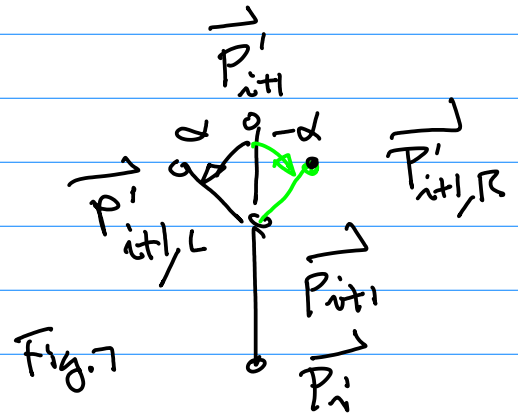


Fig. 7

After Before

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

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

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

Rotation

Suppose $\alpha = 30^\circ$

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

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

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

Team 1 { Christiana 1702,
 Anh 3807,
 Charles 0426
 Ken 3381

2° The Rotation in (3b)

Team 2 { Christopher 7230
 Patric 8001
 Anmol 9654

defines the Rotation

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

Team 3. $\left\{ \begin{array}{l} \text{Yong 4236} \\ \text{Zhonglin 6764} \end{array} \right.$

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 Productive

Promote/Present Better Communication

By Sept. 23rd Submission to e-mail;
= hua.li@sjtu.edu

Show + Tell

Note: For more general cases,

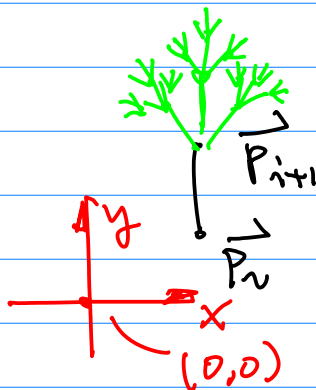


Fig. 8