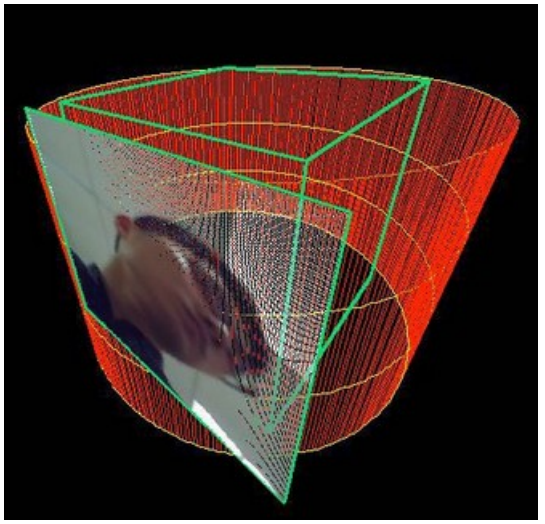
Introduction Advancement in AR and Computer Graphics

1

Pixel Graphics and Vector Graphics as the foundation

http://opencv.org/

www.opengl.org



2

https://xinreality.com/wiki/Input_Devices

IoT as a tool for better User Interface Experience, in 3D spaces

Accelerometer: senses the speed of the moving object in a particular direction. The accelerometer calculates the speed of the controller in the three axis

Intel realsense, Microsoft hololens etc , including gesture tracker

3

Smart phone becomes a popular deployment platform

4

CUDA GPU as core processing engine

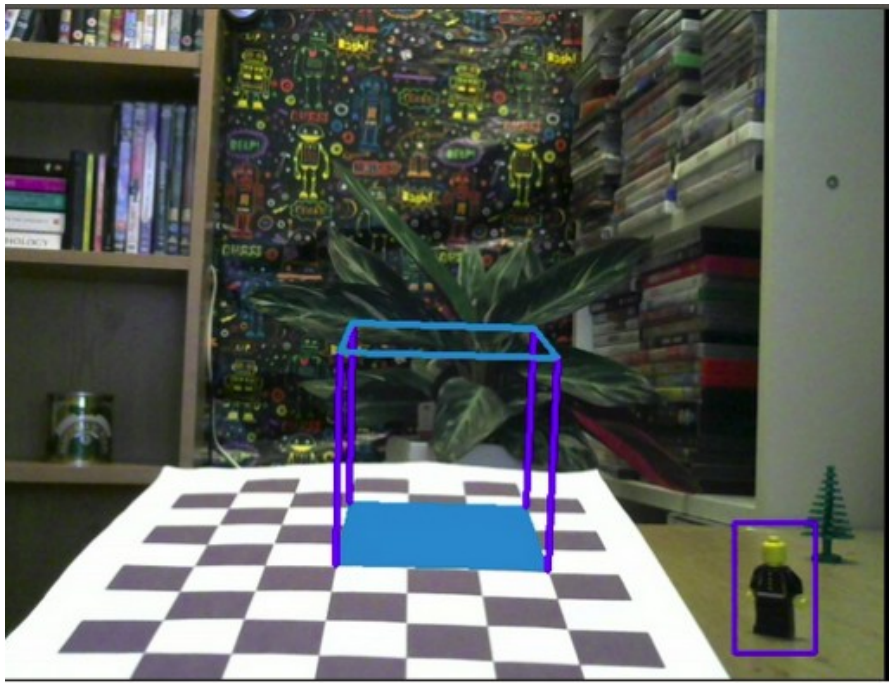
Embedded GPU Tx2 or GPU for x86 platform

Tx2 NVDA

https://www.marxentlabs.com/5-top-virtual-reality-augmented-reality-technology-trends-2017/

Build OpenCV with OpenGL

https://docs.opencv.org/3.0-beta/modules/core/doc/opengl_interop.html



https://www.google.com/search?q=opencv+plot+3d+points&tbm=isch&tbo=u&source=univ&sa=X&ved=2ahUKEwid7c3s8YHdAhWZGTQIHWY3DVYQsAR6BAgFEAE&biw=1301&bih=670#imgrc=ay8X4Woc_dbqBM:

sudo apt-get install libgtkglext1-dev

OpenGL interoperability

To enable OpenGL, configure OpenCV CMake with WITH_OPENGL=ON . Currently OpenGL is supported only with WIN32, GTK and Qt backends on Windows and Linux (MacOS and Android are not supported). For GTK backend gtkglext-1.0 library is required.

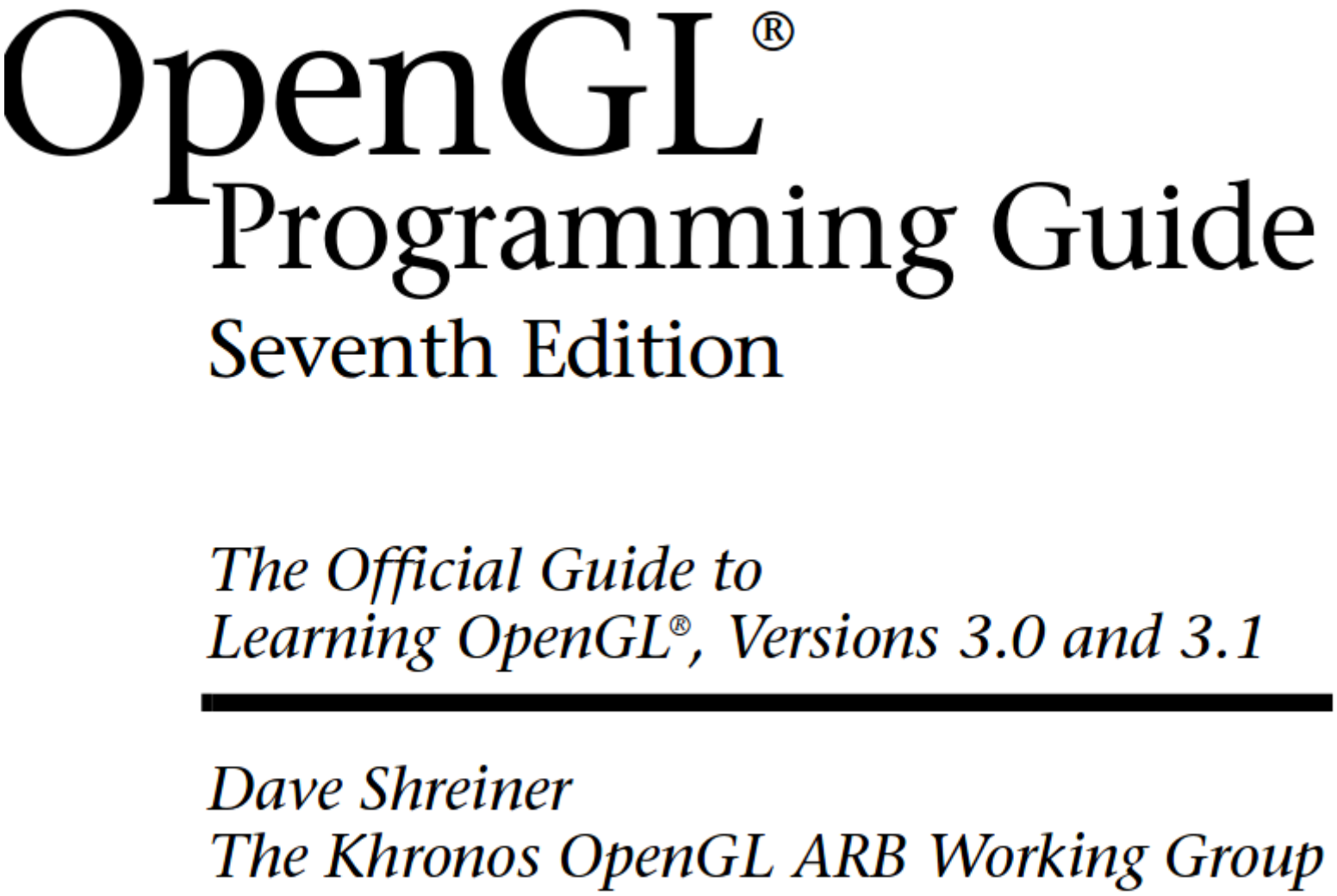
use of the cmake flag

cmake -DWITH_OPENGL=ON ..

To use OpenGL functionality you should first create OpenGL context (window or frame buffer). You can do this with namedWindow() function or with other OpenGL toolkit (GLUT, for example).

OpenGL Program

https://sgar91.files.wordpress.com/2010/12/opengl-programming-guide-7e.pdf



First OpenGL Program

OpenGL comes with Ubuntu preinstalled. However, you will have to install glut.h as follows

sudo apt-cache search glut

After you have installed package freeglut3-dev you can confirm it contains the needed GL files by issuing

dpkg -L freeglut3-dev

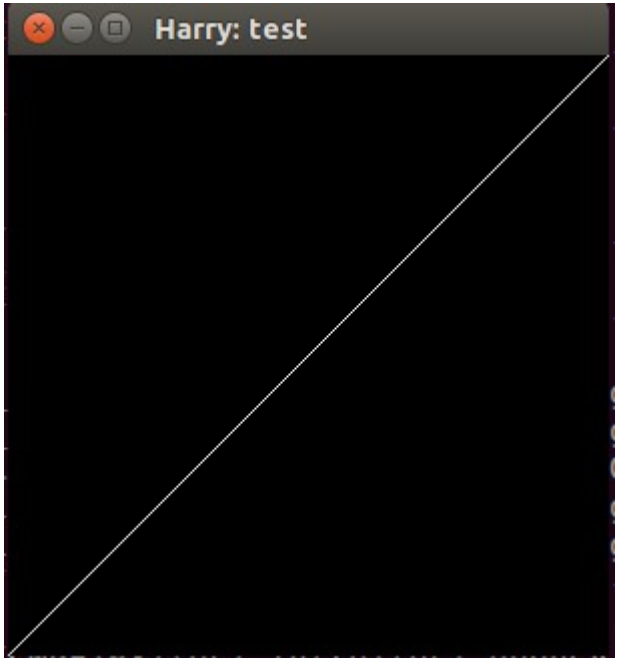
which will list contents of the package which include :

/usr/include/GL
/usr/include/GL/freeglut.h
/usr/include/GL/freeglut_ext.h
/usr/include/GL/freeglut_std.h
/usr/include/GL/glut.h
/usr/lib/x86_64-linux-gnu/libglut.a
...
/usr/lib/x86_64-linux-gnu/libglut.so

OpenGL For A Line Segment

```
/*****************************************************
 * Program: line.c   Coded by: Harry Li              *
 * Version: x1.0;    status: tested;                 *
 * Compile and build:                                *
 * gcc main.cpp -o main.o -lGL -lGLU -lglut          *
 * Date: Jun 5, 2014                                 *
 * Purpose: Graphics Demo.                           *
 *****************************************************/
#include<GL/glut.h>
#include<stdio.h>
void mydisplay()
{ //the window coordinates (-1.0, 1.0)
float p1x=1.0f,p1y=1.0f;
float p2x=-1.0f,p2y=-1.0f;
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();
glBegin(GL_LINES);
glVertex2f(p1x,p1y);
glVertex2f(p2x,p2y);
glEnd();
glFlush();
usleep(50);
}
```

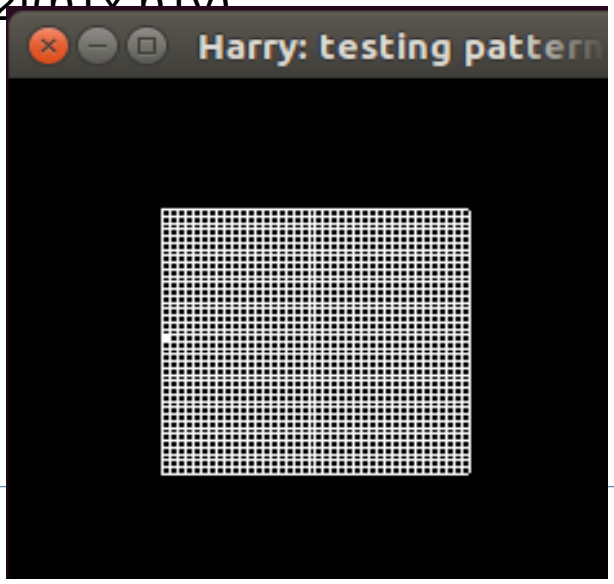
```
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutCreateWindow("Harry: test");
glutDisplayFunc(mydisplay);
glutMainLoop();
}
```



OpenGL Sample Code (1)

```
/*****************************************************
 * Program: line.c   Coded by: Harry Li              *
 * Version: x1.0;    status: tested;                 *
 * gcc main.cpp -o main.o -lGL -lGLU -lglut          *
 * Date: Jun 5, 2014                                 *
 * Purpose: 2D grids demo                            *
 *****************************************************/
#include<GL/glut.h>
#include<stdio.h>
void mydisplay()
{ //define virtex for the square
float p1x=1.0f,p1y=1.0f;
float p2x=1.0f,p2y=0.0f;
float p3x=0.0f,p3y=0.0f;
float p4x=0.0f,p4y=1.0f;
int r, i;
float grid, float_r;
//define the x-y axis
float px1x=0.0f,px1y=0.5f, px2x=1.0f, px2y=0.5f;
float px3x=0.5f,px3y=0.0f, px4x=0.5f, px4y=1.0f;
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();
glTranslatef(-0.5f,-0.5f,0.0f);
```

```
glFlush();
Sleep(1);
//-------draw square--------------------//
glBegin(GL_LINES);
//draw square
glVertex2f(p1x,p1y);
glVertex2f(p2x,p2y);
glVertex2f(p2x,p2y);
glVertex2f(p3x,p3y);
glVertex2f(p3x,p3y);
glVertex2f(p4x,p4y);
glVertex2f(p4x,p4y);
glVertex2f(p1x,p1y);
glEnd();
```



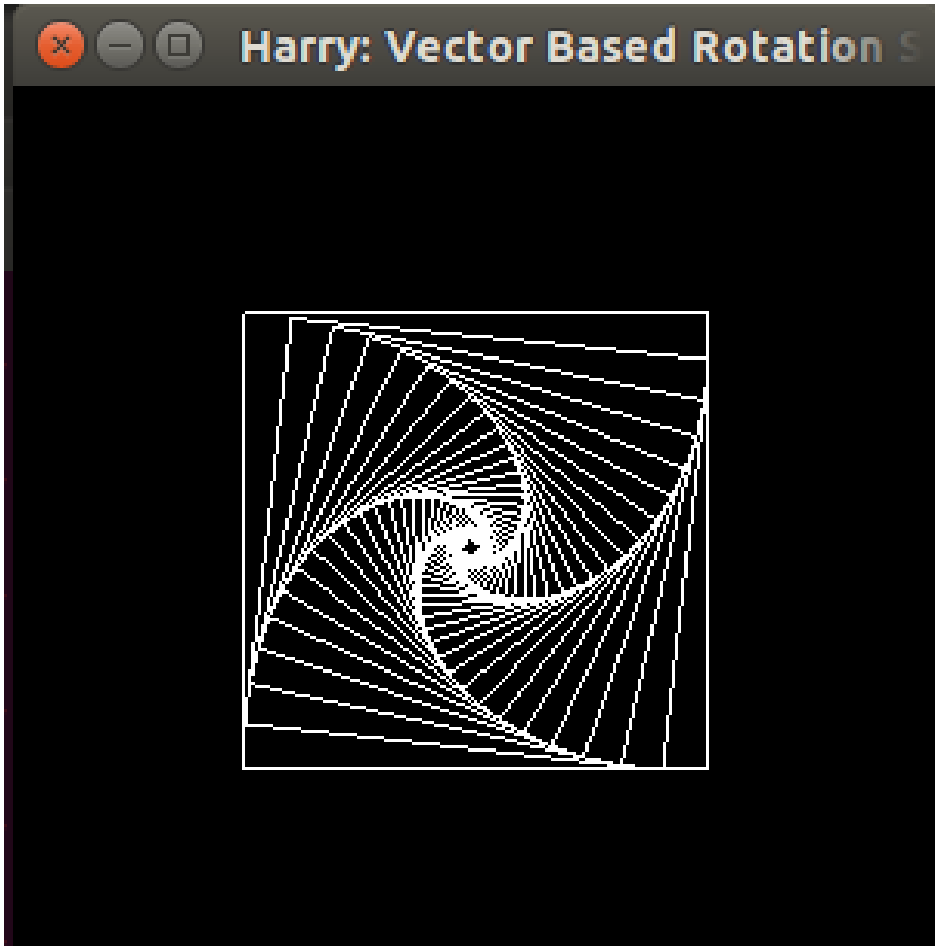
OpenGL Sample Code (2)

```
//-------draw x-y axis-------------------//
glBegin(GL_LINES);
glVertex2f(px1x,px1y);
glVertex2f(px2x,px2y);
glVertex2f(px3x,px3y);
glVertex2f(px4x,px4y);
glEnd();
//-------draw grids-------------------//
float_r = 20.0; r =  float_r;
grid = 1.0/(2*float_r);

glBegin(GL_LINES);
for (i=0; i < r; i++) {
glVertex2f(px1x,px1y+i*grid);  //x, upper half
glVertex2f(px2x,px2y+i*grid);
glVertex2f(px1x,px1y-i*grid);   //x, lower half
glVertex2f(px2x,px2y-i*grid);
glVertex2f(px3x+i*grid,px3y);  //y, right half
glVertex2f(px4x+i*grid,px4y);
glVertex2f(px3x-i*grid,px3y);   //y, left half
glVertex2f(px4x-i*grid,px4y);
}
glEnd();
```

```
//-------draw grid polygon----------------//
glBegin(GL_POLYGON);
glVertex2f(px1x, px1y);
//px1y = px1y + grid;
glVertex2f(px1x, px1y);
glVertex2f(px1x, px1y+grid);
glVertex2f(px1x+grid, px1y+grid);
glVertex2f(px1x+grid, px1y);
glEnd();
glFlush();
usleep(50);
}

int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutCreateWindow("Harry: testing
pattern: square");
glutDisplayFunc(mydisplay);
glutMainLoop();
}
```

Vector Graphics 2D Rotating Square (1)

```
/*****************************************************
 * Program: LinePattern.c                            *
 * Purpose: Rotation square based vectors            *
 * Compile and build:                                *
 *         gcc -lGU -lGLU -lglut                     *
*****************************************************/
#include<GL/glut.h>
#include<stdio.h>
#define MAX_LEVEL 35
#define LAMBDA 0.1
void mydisplay()
{
float p1x=1.0f,p1y=1.0f,p2x=1.0f,p2y=0.0f;
float p3x=0.0f,p3y=0.0f,p4x=0.0f,p4y=1.0f;
int i=0;
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();
glTranslatef(-0.5f,-0.5f,0.0f);
glFlush();
sleep(1);
```

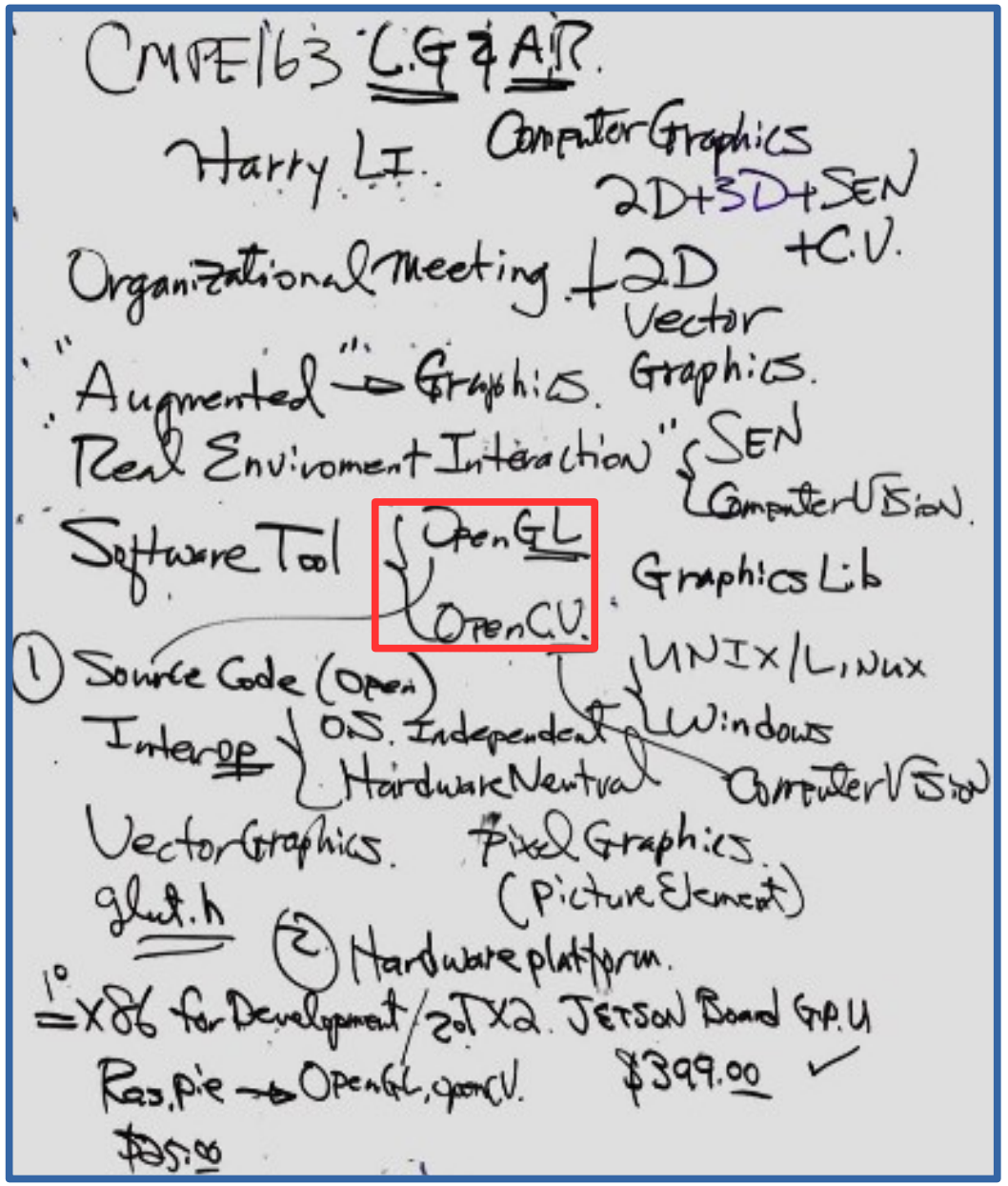


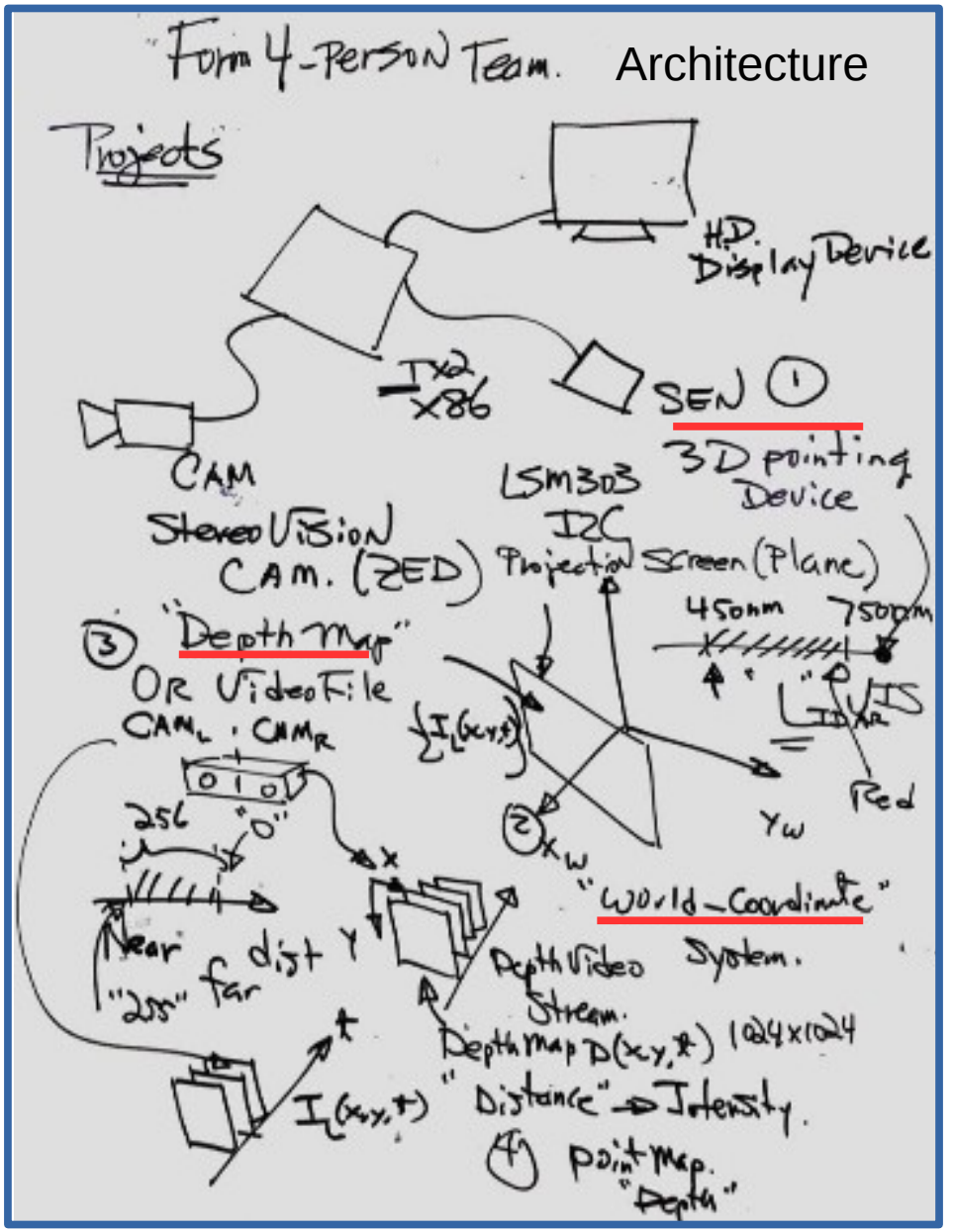
Vector Graphics 2D Rotating Square (2)

```
for(i=0;i<MAX_LEVEL;i++)
{ glBegin(GL_LINES);
 glVertex2f(p1x,p1y);
glVertex2f(p2x,p2y);
glVertex2f(p2x,p2y);
glVertex2f(p3x,p3y);
glVertex2f(p3x,p3y);
glVertex2f(p4x,p4y);
glVertex2f(p4x,p4y);
glVertex2f(p1x,p1y);

p1x=p1x + (LAMBDA * (p2x-p1x));
p1y=p1y + (LAMBDA * (p2y-p1y));
p2x=p2x + (LAMBDA * (p3x-p2x));
p2y=p2y + (LAMBDA * (p3y-p2y));
p3x=p3x + (LAMBDA * (p4x-p3x));
p3y=p3y + (LAMBDA * (p4y-p3y));
p4x=p4x + (LAMBDA * (p1x-p4x));
p4y=p4y + (LAMBDA * (p1y-p4y));
glEnd();
glFlush();
usleep(50000);
}
}
```

```
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutCreateWindow("Harry: Vector Square");
glutDisplayFunc(mydisplay);
glutMainLoop();
}
```

8-23-2018 Introduction System Architecture





8-23-2018 Line Equation In Vector Form

CMPE163 C.G & AR.

One Midterm 30pts 30%

Final Exam. 40pts, 40%. Rm267A

Office Hours: M.W. 4:30–5:30pm.

hua.li@sjsu.edu & hualili@yahoo.com

(650) 400-1116

TextBooks: On-Line check URL Listings.

Conduct of the Class: 1° 10 min Breaks. ON Each hr.

2° Last hour for Hands-on Work.

Track your work (No. of Lines of Coding) you have done for this class.

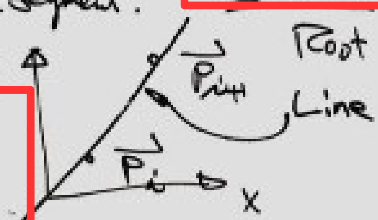
* Introduction to 2D Computer Graphics.

Example: Draw a line segment. glVertex2f

line 17 & 20

$\vec{P_i} \to \vec{P_i}(x_i, y_i) \to$

$\vec{P_i}(x_i, y_i) = (x_i, y_i)$



4/.

$$\vec{P} = \vec{P_i} + \lambda \vec{d} \quad \cdots (1)$$

λ: Scalar.

directional vector

$$\vec{d} = \vec{P_{i+1}} - \vec{P_i} \quad \cdots (2)$$

$$\vec{d}(x_d, y_d) = \vec{P_{i+1}}(x_{i+1}, y_{i+1}) - \vec{P_i}(x_i, y_i)$$

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

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

OR

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

$$\begin{cases} x_d = x_{i+1} - x_i \\ y_d = y_{i+1} - y_i \end{cases} \Rightarrow$$

C/C++

```
x_d = x[i+1] - x[i];
y_d = y[i+1] - y[i];
```

3 Equivalent Notations

In C/C++ for directional vector