



Recitation Class 09 for VG101

Date: 2012 / 11 / 26

Wang Qian

Mid-term 02

- Much easier than mid-term 01.
- Expected mean value: 60 - 70 ?
- Grader

No.	Grader
Problem 1	Zheng Yue
Problem 2	Wang Qian
Problem 3	Luo Jiaxi
Problem 4	Zhang Huajun

- If you still have problems or questions, please feel free to contact with us.

Explanation

- Key points

No.	Key Points
Problem 1	Basic knowledge and concepts
Problem 2	Euclid's Algorithm, function and comments
Problem 3	String and Algorithm Design
Problem 4	Array, Function, Pass by Reference and Sort

- Since this exam is easy, I will not explain in details. If you have problems with solving them, please feel free to contact the corresponding grader for help.

Explanation

- Problem 2

```
#include <stdio.h>
#include <stdlib.h>
int lcm(int a,int b);
int main()
{
    int a,b,x;
    scanf("%d%d",&a,&b);
    x = lcm(a,b);
    printf("%d\n",x);
    system("pause");
    return 0;
}
```

```
int lcm(int a,int b)
{
    int r,x = a,y = b;
    while (b > 0)
    {
        r = a % b;
        a = b;
        b = r;
    }
    return x / a * y;
}
```

Explanation

- Problem 3

```
#include <stdio.h>
#include <stdlib.h>
#define TOT 26
#define LEN 100
int main()
{
    int key,i;
    char tmp,text[LEN];
    char UPPER[TOT <<1],LOWER[TOT << 1];
    scanf("%d%c",&key,&tmp);
    for (i = 0; i < TOT; i++)
    {
        UPPER[i] = UPPER[i+TOT] = 'A' + i;
        LOWER[i] = LOWER[i+TOT] = 'a' + i;
    }
}
```

Explanation

- Problem 3 (cont.)

```
gets(text);
for (i = 0; text[i]; i++)
    if (text[i] >= 'A' && text[i] <= 'Z')
        text[i] = UPPER[text[i]-'A'+key];
    else if (text[i] >= 'a' && text[i] <= 'z')
        text[i] = LOWER[text[i]-'a'+key];

puts(text);
system("pause");
return 0;
}
```


Explanation

- Problem 4

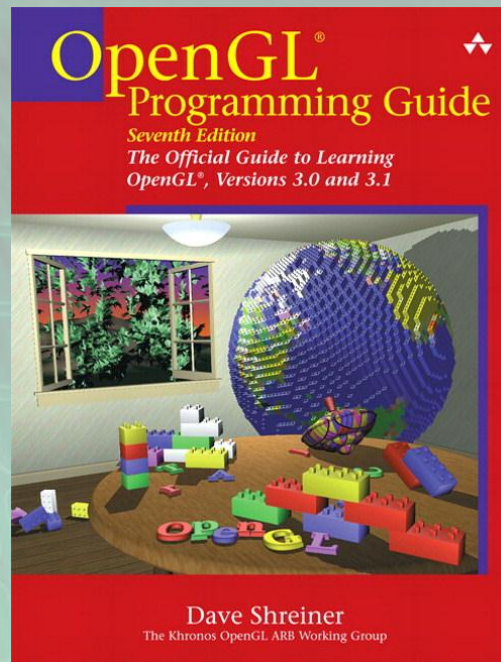
```
int getArray(int *a,int term)
{
    int i = 0,x;
    for (;i < MAX; i++)
    {
        scanf("%d",&x);
        if (x == term)
            return i;
        a[i] = x;
    }
    return i;
}
```

```
int sortArray(int *a,int n)
{
    int i,j,t;
    for (i = 0; i < n; i++)
        for (j = i; j < n; j++)
            if (a[j] < a[i])
            {
                t = a[i];
                a[i] = a[j];
                a[j] = t;
            }
    return 0;
}
```

```
int outArray(int *a, int n)
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ",a[i]);
    return 0;
}
```

OpenGL

- Only some very basic operations required in Vg101.
- If you really want to know more:
 - *OpenGL Programming Guide (7th ed.)* by David Shreiner



<http://www.informit.com/store/opengl-programming-guide-the-official-guide-to-learning-9780321552624>

OpenGL

- Tip: use the code in the lecture as a template.
- Only a few statements needs understanding.
- Settings about the window:
 - **void glutInitWindowPosition(int x, int y);**
 - Set the position of the window on the screen.
 - **void glutInitWindowSize(int width, int height);**
 - Set the size of the window.
 - **void glutCreateWindow(char *name);**
 - Set the title of the window.

OpenGL

- Settings about the window color:
 - **void glClearColor(red, green, blue, 0.0);**
 - Parameter “alpha” is about blending. Set it as 0.0 now.
 - For the first three parameters, remember the following setting:
 - `glClearColor(0.0,0.0,0.0,0.0);` black
 - `glClearColor(1.0,0.0,0.0,0.0);` red
 - `glClearColor(0.0,1.0,0.0,0.0);` green
 - `glClearColor(1.0,1.0,0.0,0.0);` yellow
 - `glClearColor(0.0,0.0,1.0,0.0);` blue
 - `glClearColor(1.0,0.0,1.0,0.0);` magenta
 - `glClearColor(0.0,1.0,1.0,0.0);` cyan
 - `glClearColor(1.0,1.0,1.0,0.0);` white
 - **void glClear(GL_COLOR_BUFFER_BIT);**

OpenGL

- Settings about the object color:
 - **void glColor3f(red, green, blue);**
 - Set the color for the following objects.
 - In this course, we only use “3f”, means which the parameters should be “three float type parameters”.
 - The meaning of these parameters are the same as the previous slide.

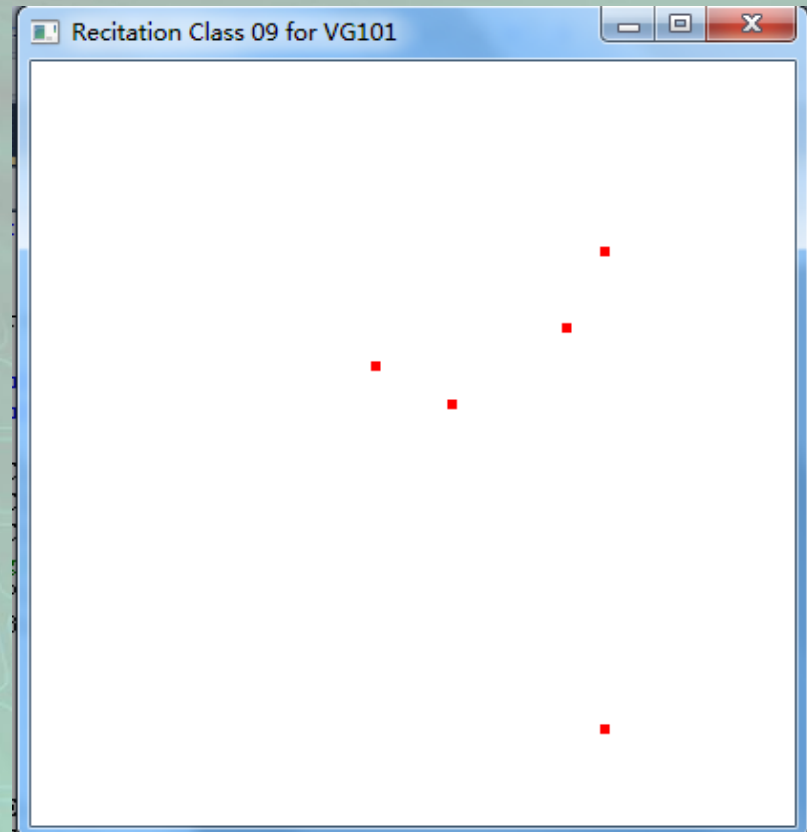
OpenGL

- Drawing objects:
 - First, we should know the following function:
 - **void glBegin(GLenum mode);**
 - **void glEnd(void);**
 - Between them, there are a sequence of vertices, presented by:
 - **void glVertex2f(float x, float y);**
 - The same, only “2f” is needed in this course.
 - Sometimes “4f” is also very convenient.
 - **void glVertex4f(float x, float y, float z, float w);**
 - Notice that the coordinates will all be divided by w.
 - I will give examples later.
- Let's discuss the input parameter of glBegin() in detail.

OpenGL

- GL_POINTS:
 - Draw separate points for each vertices.

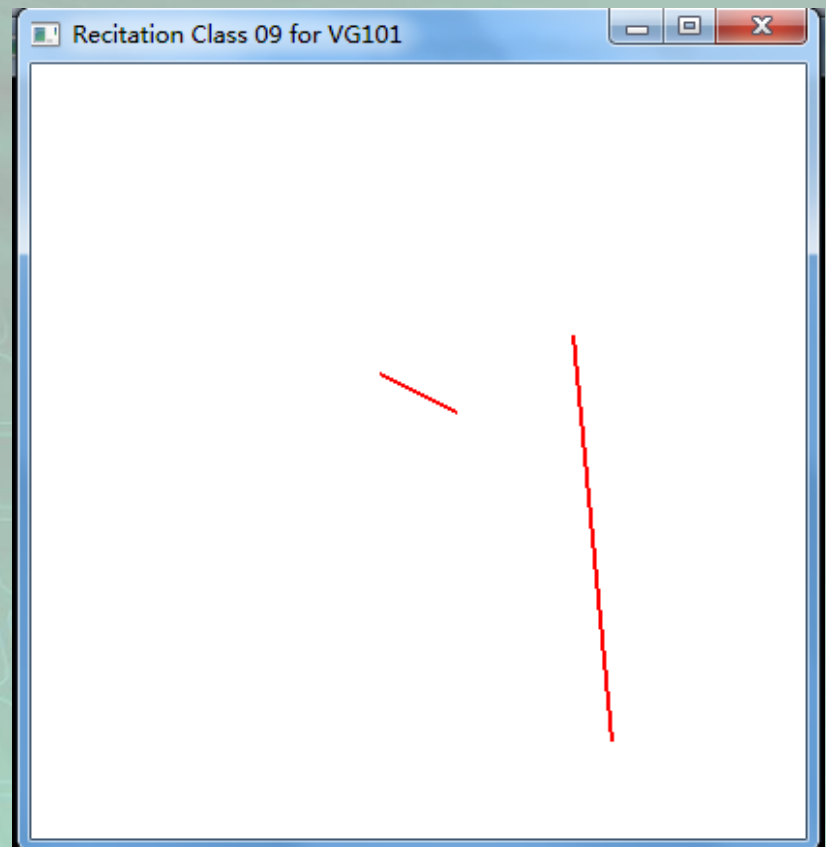
```
glPointSize(5.0);  
glBegin(GL_POINTS);  
    glVertex4f(1.0,1.0,0.0,10.0);  
    glVertex4f(-1.0,2.0,0.0,10.0);  
    glVertex4f(4.0,3.0,0.0,10.0);  
    glVertex4f(5.0,-7.5,0.0,10.0);  
    glVertex4f(5.0,5.0,0.0,10.0);  
glEnd();
```



OpenGL

- **GL_LINES:**
 - Draw unconnected line segments.

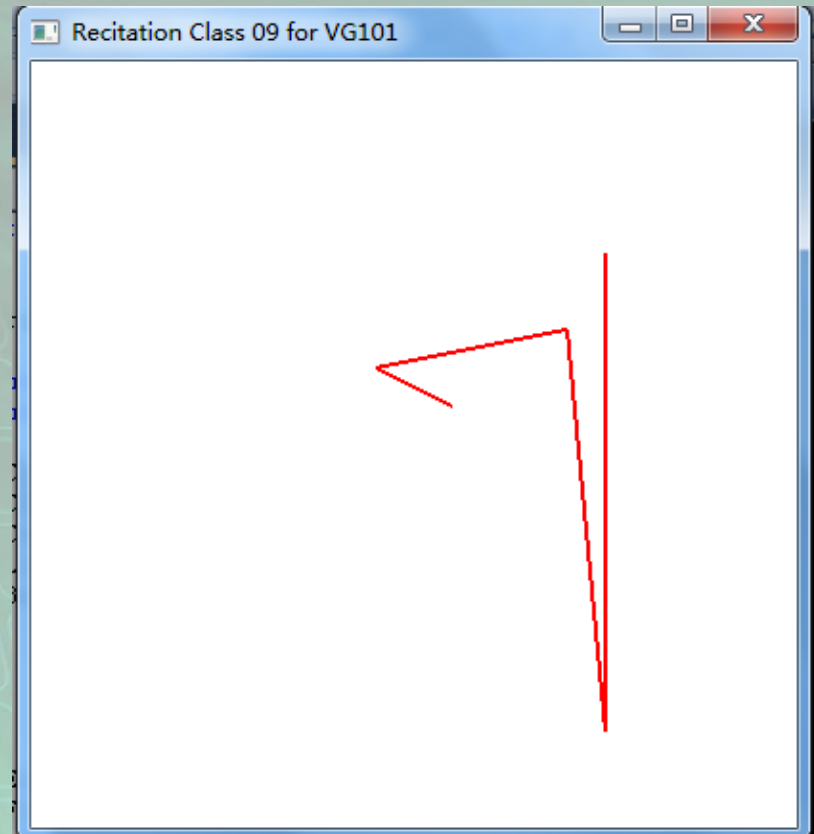
```
glLineWidth(2.0);  
glBegin(GL_LINES);  
    glVertex4f(1.0,1.0,0.0,10.0);  
    glVertex4f(-1.0,2.0,0.0,10.0);  
    glVertex4f(4.0,3.0,0.0,10.0);  
    glVertex4f(5.0,-7.5,0.0,10.0);  
    glVertex4f(5.0,5.0,0.0,10.0);  
glEnd();
```



OpenGL

- **GL_LINE_STRIP:**
 - Draw connected line segments with two endings.

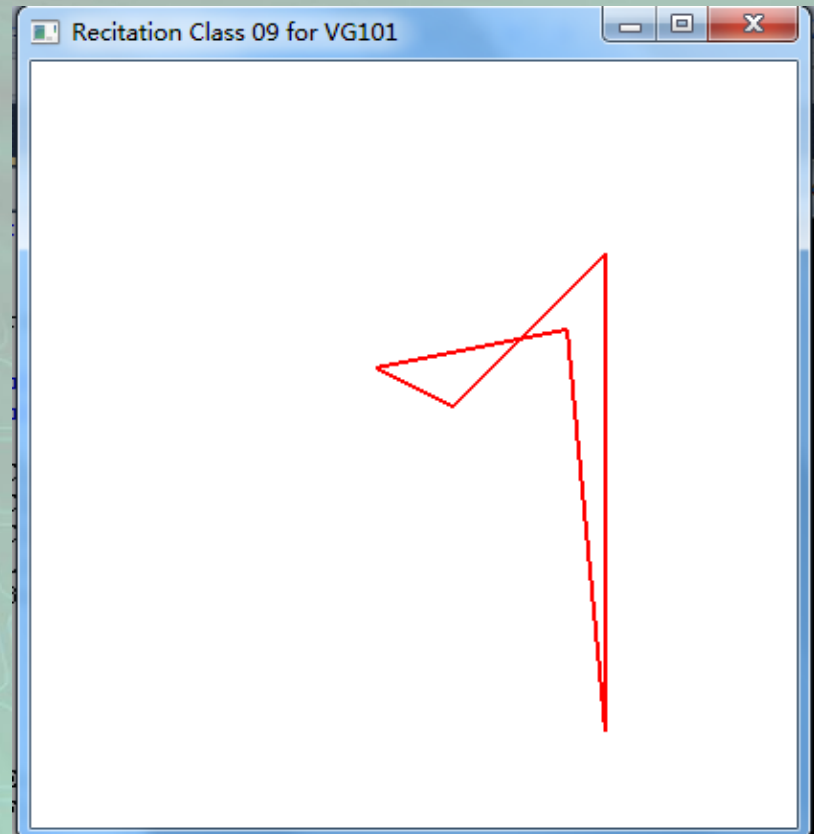
```
glLineWidth(2.0);  
glBegin(GL_LINE_STRIP);  
    glVertex4f(1.0,1.0,0.0,10.0);  
    glVertex4f(-1.0,2.0,0.0,10.0);  
    glVertex4f(4.0,3.0,0.0,10.0);  
    glVertex4f(5.0,-7.5,0.0,10.0);  
    glVertex4f(5.0,5.0,0.0,10.0);  
glEnd();
```



OpenGL

- **GL_LINE_LOOP:**
 - Draw connected line segments with no ending.

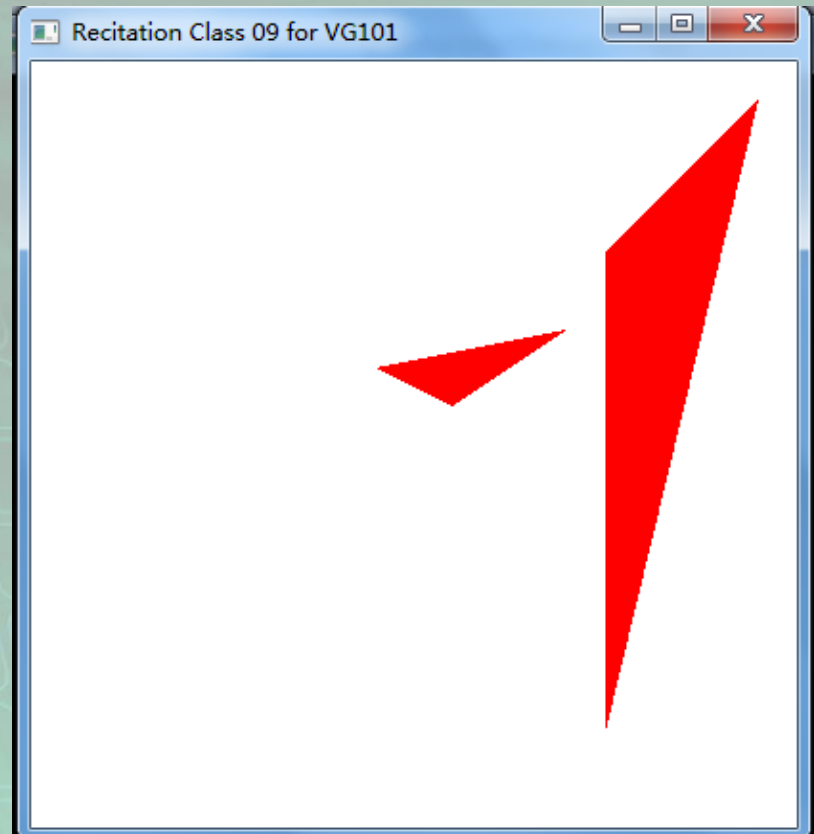
```
glLineWidth(2.0);  
glBegin(GL_LINE_LOOP);  
    glVertex4f(1.0,1.0,0.0,10.0);  
    glVertex4f(-1.0,2.0,0.0,10.0);  
    glVertex4f(4.0,3.0,0.0,10.0);  
    glVertex4f(5.0,-7.5,0.0,10.0);  
    glVertex4f(5.0,5.0,0.0,10.0);  
glEnd();
```



OpenGL

- GL_TRIANGLES:
 - Draw separate triangles.

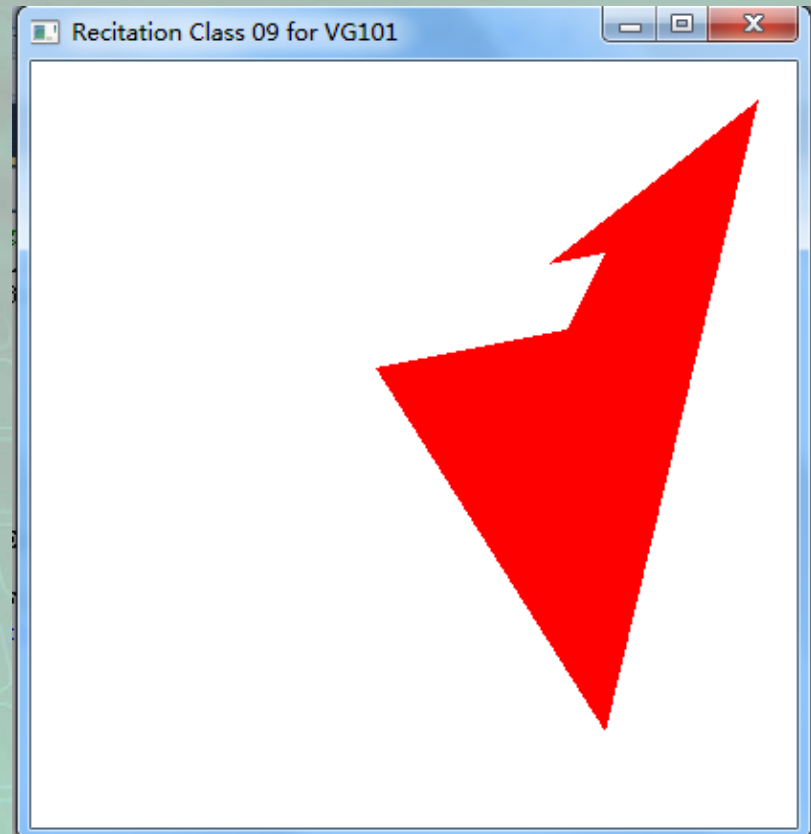
```
glBegin(GL_TRIANGLES);  
    glVertex4f(1.0,1.0,0.0,10.0);  
    glVertex4f(-1.0,2.0,0.0,10.0);  
    glVertex4f(4.0,3.0,0.0,10.0);  
    glVertex4f(5.0,-7.5,0.0,10.0);  
    glVertex4f(5.0,5.0,0.0,10.0);  
    glVertex4f(9.0,9.0,0.0,10.0);  
    glVertex4f(3.5,4.7,0.0,10.0);  
glEnd();
```



OpenGL

- **GL_TRIANGLE_STRIP:**
 - Draw connected triangles. (The order of vertices?)

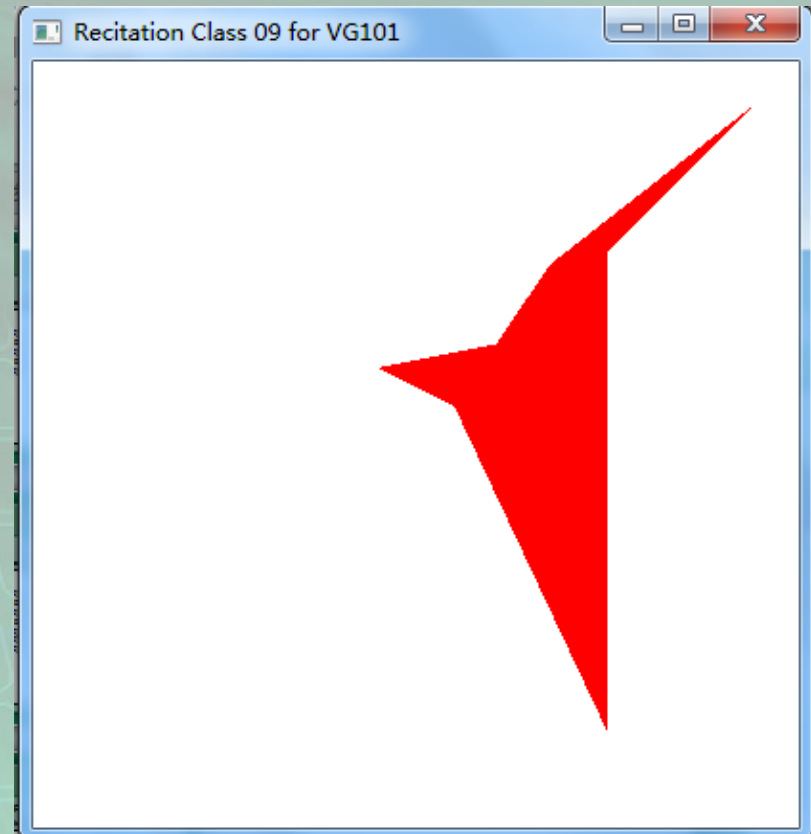
```
glBegin(GL_TRIANGLE_STRIP);  
    glVertex4f(1.0,1.0,0.0,10.0);  
    glVertex4f(-1.0,2.0,0.0,10.0);  
    glVertex4f(4.0,3.0,0.0,10.0);  
    glVertex4f(5.0,-7.5,0.0,10.0);  
    glVertex4f(5.0,5.0,0.0,10.0);  
    glVertex4f(9.0,9.0,0.0,10.0);  
    glVertex4f(3.5,4.7,0.0,10.0);  
glEnd();
```



OpenGL

- **GL_TRIANGLE_FAN:**
 - Draw triangle-fans with a common vertex. (The order of vertices?)

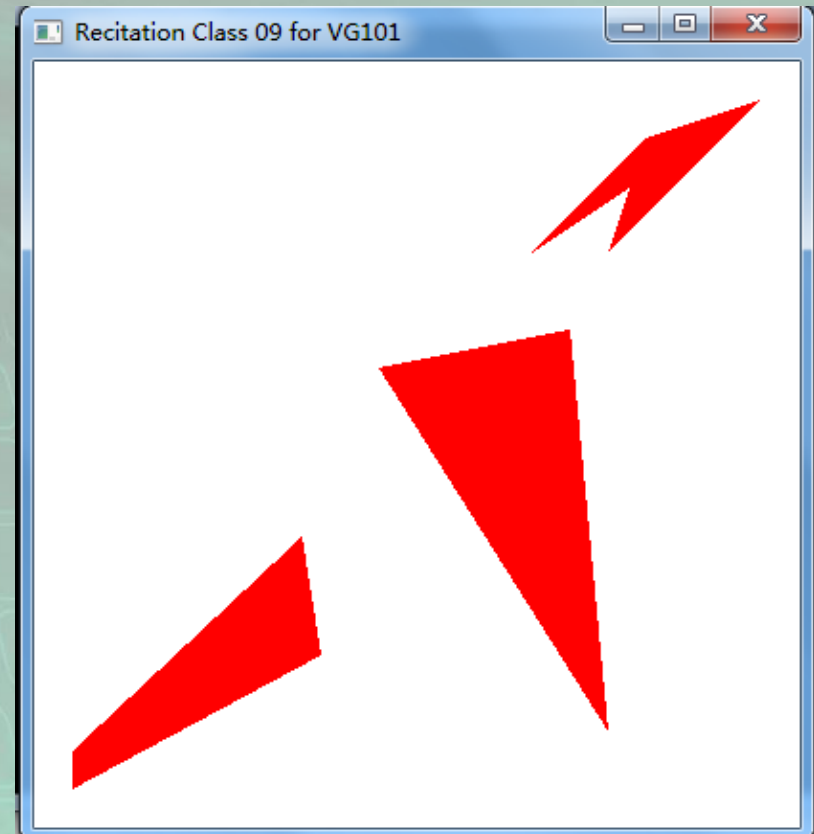
```
glBegin(GL_TRIANGLE_FAN);  
    glVertex4f(1.0,1.0,0.0,10.0);  
    glVertex4f(-1.0,2.0,0.0,10.0);  
    glVertex4f(4.0,3.0,0.0,10.0);  
    glVertex4f(5.0,-7.5,0.0,10.0);  
    glVertex4f(5.0,5.0,0.0,10.0);  
    glVertex4f(9.0,9.0,0.0,10.0);  
    glVertex4f(3.5,4.7,0.0,10.0);  
glEnd();
```



OpenGL

- GL_QUADS:
 - Separate quadrilateral. (valid only for convex quadrilateral)

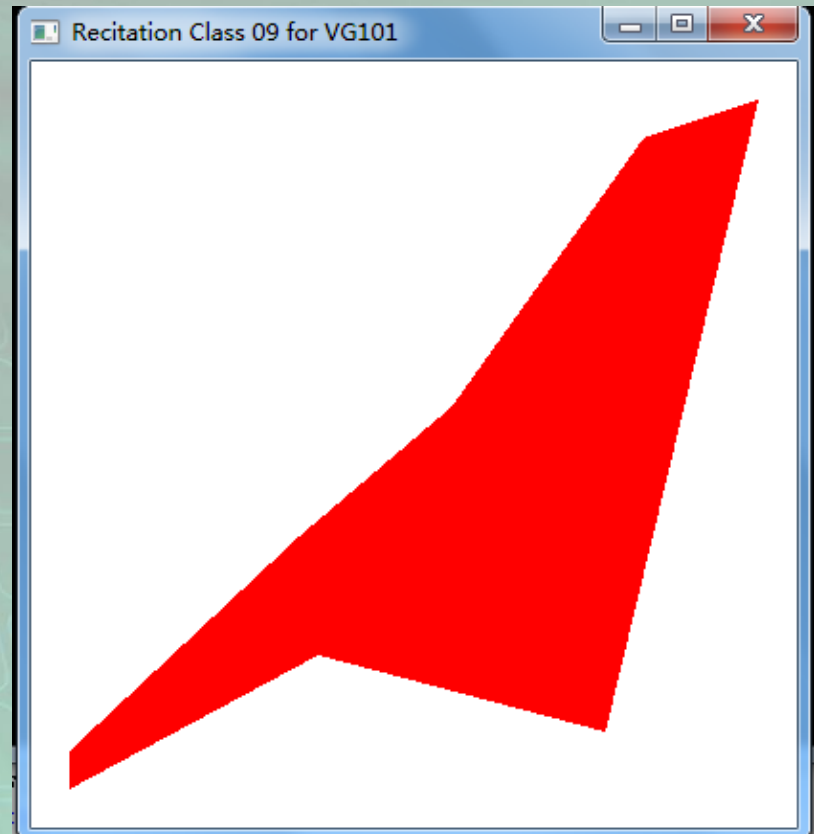
```
glBegin(GL_QUADS);  
    glVertex4f(-9.0,-9.0,0.0,10.0);  
    glVertex4f(-9.0,-8.0,0.0,10.0);  
    glVertex4f(-3.0,-2.4,0.0,10.0);  
    glVertex4f(-2.5,-5.5,0.0,10.0);  
    glVertex4f(1.0,1.0,0.0,10.0);  
    glVertex4f(-1.0,2.0,0.0,10.0);  
    glVertex4f(4.0,3.0,0.0,10.0);  
    glVertex4f(5.0,-7.5,0.0,10.0);  
    glVertex4f(5.0,5.0,0.0,10.0);  
    glVertex4f(9.0,9.0,0.0,10.0);  
    glVertex4f(3.0,5.0,0.0,10.0);  
    glVertex4f(6.0,8.0,0.0,10.0);  
glEnd();
```



OpenGL

- GL_QUAD_STRIP:
 - Connected quadrilateral. (The order of vertices?)

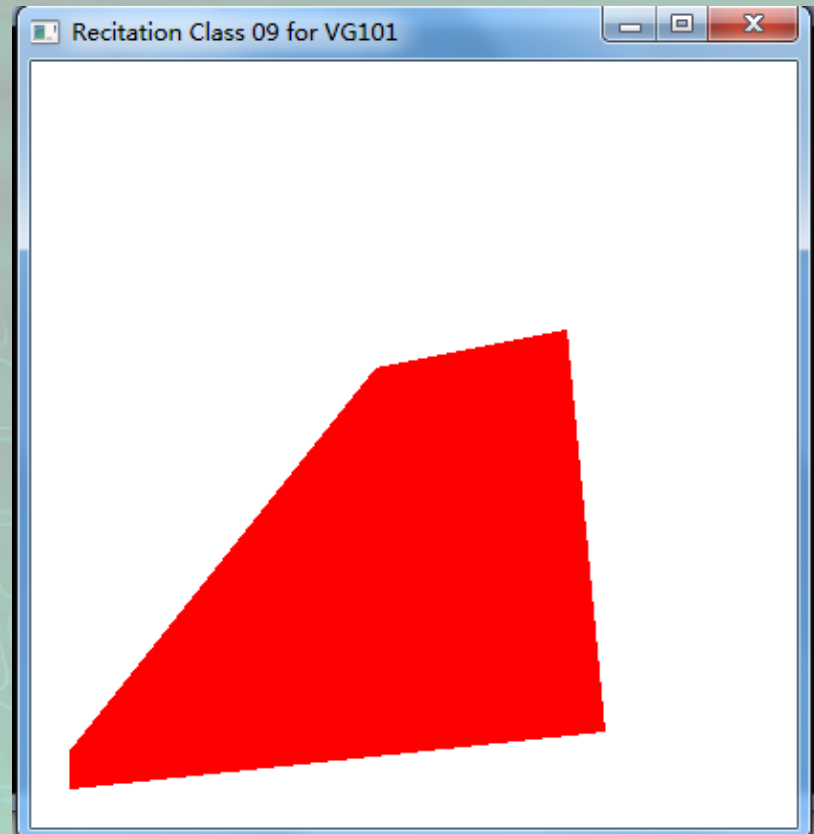
```
glBegin(GL_QUAD_STRIP);  
    glVertex4f(-9.0,-8.0,0.0,10.0);  
    glVertex4f(-9.0,-9.0,0.0,10.0);  
    glVertex4f(-3.0,-2.4,0.0,10.0);  
    glVertex4f(-2.5,-5.5,0.0,10.0);  
    glVertex4f(1.0,1.0,0.0,10.0);  
    glVertex4f(5.0,-7.5,0.0,10.0);  
    glVertex4f(6.0,8.0,0.0,10.0);  
    glVertex4f(9.0,9.0,0.0,10.0);  
glEnd();
```



OpenGL

- GL_QUAD_STRIP:
 - Only a polygon. (valid only for convex polygon)

```
glBegin(GL_POLYGON);  
    glVertex4f(-9.0,-9.0,0.0,10.0);  
    glVertex4f(-9.0,-8.0,0.0,10.0);  
    glVertex4f(-1.0,2.0,0.0,10.0);  
    glVertex4f(4.0,3.0,0.0,10.0);  
    glVertex4f(5.0,-7.5,0.0,10.0);  
glEnd();
```



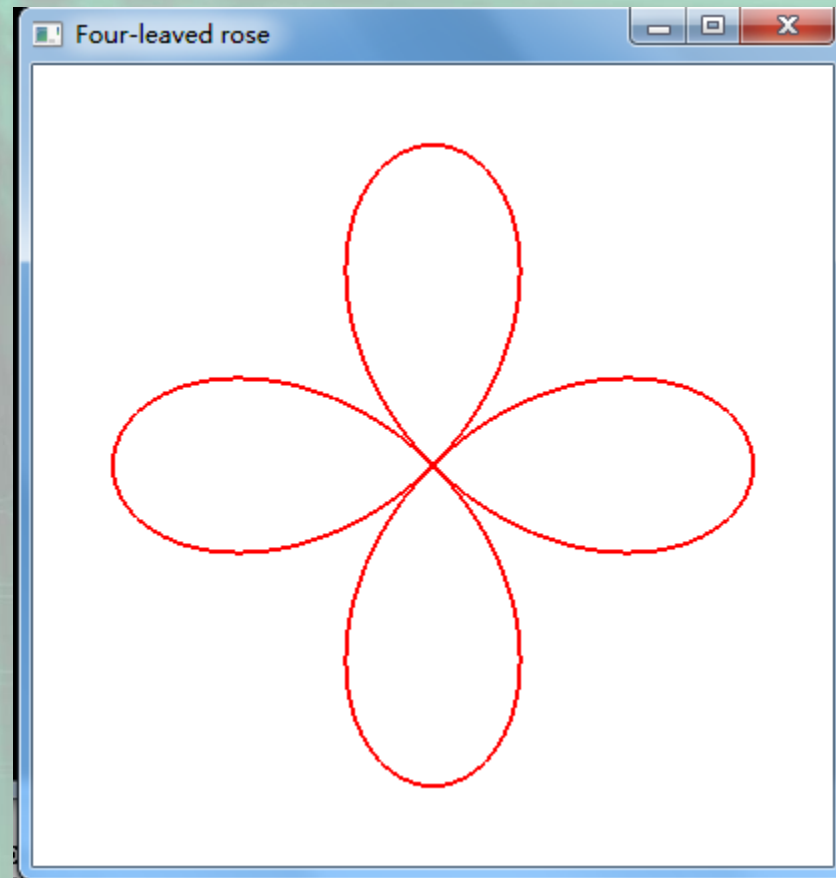
OpenGL

- Summary (assume there are six points):

Value	Order
GL_POINTS	v0; v1; v2; v3; v4; v5; v6
GL_LINES	v0v1; v2v3; v3v4; v4v5
GL_LINE_STRIP	v0v1v2v3v4v5v6
GL_LINE_LOOP	v0v1v2v3v4v5v6v0
GL_TRIANGLES	v0v1v2; v3v4v5
GL_TRIANGLE_STRIP	v0v1v2; v1v2v3; v2v3v4 ...
GL_TRIANGLE_FAN	v0v1v2; v0v2v3; v0v3v4 ...
GL_QUADS	v0v1v2v3
GL_QUAD_STRIP	v0v1v3v2; v2v3v5v4
GL_POLYGON	v0v1v2v3v4v5v6v0

Quiz

- Sketch the four leaved rose



Quiz

- Never forget the idea of using parametric equations.

```
#include <GL\glut.h>
#include <math.h>
#define R 0.8
#define RAD_PER_DGR (3.14159265358 / 180)
void DrawIt();
int main(int argc, char *argv[]) {
    glutInit(&argc, argv);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
    glutCreateWindow("Four-leaved rose");
    glutDisplayFunc(DrawIt);
    glutMainLoop();
    return 0;
}
```

Quiz

```
void DrawIt() {  
    float x,y,r;  
    int i;  
    glClearColor(1.0,1.0,1.0,0.0);  
    glClear(GL_COLOR_BUFFER_BIT);  
    glColor3f(1.0,0.0,0.0);  
    glLineWidth(2.0);  
    glBegin(GL_LINE_LOOP);  
    for (i = 0; i < 360; i++) {  
        r = R * cos(2 * i * RAD_PER_DGR);  
        x = r * cos(i * RAD_PER_DGR);  
        y = r * sin(i * RAD_PER_DGR);  
        glVertex2f(x,y);  
    }  
    glEnd();  
    glFlush();  
}
```


Quiz

- What about showing the whole procedure of generation?

```
.....  
void DrawIt();  
void Tick(int);  
int main(int argc, char *argv[]) {  
    glutInit(&argc, argv);  
    glutInitWindowPosition(100, 100);  
    glutInitWindowSize(400, 400);  
    glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);  
    glutCreateWindow("Four-leaved rose");  
    glutDisplayFunc(DrawIt);  
    glutTimerFunc(10, Tick, 0.0);  
    glutMainLoop();  
    return 0;  
}
```

Quiz

```
void Tick(int n) {
    glutTimerFunc(10, Tick, 0.0);
    glutPostRedisplay();
}
void DrawIt() {
    .....
    glLineWidth(2.0);
    glBegin(GL_LINE_STRIP);
    for (i = 0; i < t; i++) {
        r = R * cos(2 * i * RAD_PER_DGR);
        x = r * cos(i * RAD_PER_DGR);
        y = r * sin(i * RAD_PER_DGR);
        glVertex2f(x, y);
    }
    t++;
    .....
}
```