Lecture 3 Generate a simple shape using OpenGL

Yong-Jin Liu
What is this course about?

- Generate an element shape
  - point, line, region
  - OpenGL command
- Window based programming
  - Message-driven Mechanism
  - call back function
- Use GLUT advanced library
  - Window, menu, mouse response
Raster graphics
Hardware display system

- Monitor
  - CRT (Cathode Ray Tube)
  - LCD (Liquid Crystal Display)
  - LED (Light Emitting Diode)
Scan conversion of line
Solution

A line passing \( P_0(x_0, y_0), P_1(x_1, y_1) \)

\[
L: \quad y = kx + b
\]

\[
k = \frac{y_1 - y_0}{x_1 - x_0}
\]

\[
\text{for (x}=x_1; \ x<=x_2; \ x++){\quad y = kx+b; \quad \text{draw_pixel}(x, \ \text{round}(y), \ \text{line_color});}
\]
- **DDA**
  - Digital differential analyser
  - \( \frac{dy}{dx} = k \)
  - \((x_1, y_1), (x_2, y_2)\)
  - \(y = kx + b \Rightarrow \Delta y = k\Delta x \Rightarrow \Delta y = k \quad (\Delta x = 1)\)

```java
for (x=x_1; x<=x_2; x++) {
    y+=k;
    y+=k;
    draw_pixel(x, round(y), line_color);
}
```
Scan conversion of line $\Rightarrow$
Closed polygon region

![Diagram of scan conversion of line and closed polygon region](image)
Region filling
4-connected region and 8-connected region

four direction eight direction 4-connected region 8-connected region

● 表示内点 ○ 表示边界点
Recursive filling algorithm of 4-connected region represented by boundary:

```c
void BoundaryFill4(int x, int y, int boundarycolor, int newcolor)
{
    int color;
    if(color!=newcolor && color!=boundarycolor)
    {
        drawpixel(x,y,newcolor);
        BoundaryFill4 (x,y+1, boundarycolor,newcolor);
        BoundaryFill4 (x,y-1, boundarycolor,newcolor);
        BoundaryFill4 (x-1,y, boundarycolor,newcolor);
        BoundaryFill4 (x+1,y, boundarycolor,newcolor);
    }
}
```
Scan conversion algorithm of polygon region
Scan line algorithm

- Basic idea:
  - According scan line order, calculate the intersection region of scan line and polygon, then display these region using the required color, then the work is completed.

- The Scan line algorithm can be divided into four steps:
  - intersection
  - sort
  - pair
  - filling
OpenGL:
Draw point, line, face

```c
glBegin(parameter);
...
...
glEnd();
```

parameter: GL_POINTS, GL_LINES,
GL_POLYGON, GL_TRIANGLES
Draw points

```c
glBegin(GL_POINTS);
    glVertex3f(-0.5, -0.5, 0.0);
glEnd();

glBegin(GL_POINTS);
    glVertex3f(0.0, 0.5, 0.0);
glEnd();

.......
```

Low efficiency!
Draw point

```c
glBegin(GL_POINTS);
    glVertex3f(-0.5, -0.5, 0.0);
    glVertex3f(0.5, 0.0, 0.0);
    glVertex3f(0.0, 0.5, 0.0);
    ... ...
    glEnd();
```
Draw line

Low efficiency!
Draw many lines one time

glBegin(GL_LINES);
    glVertex3f(-0.5, -0.5, 0.0);
    glVertex3f(0.5, 0.0, 0.0);
    glVertex3f(0.0, 0.5, 0.0);
    glVertex3f(0.0, 0.0, 0.5);
    ... ...

    glEnd();
OpenGL programming guide
6\textsuperscript{th} edition
Draw polygon

```c
glBegin(GL_POLYGON);
    glVertex3f(-0.5, -0.5, 0.0);
    glVertex3f(0.5, 0.0, 0.0);
    glVertex3f(0.0, 0.5, 0.0);
    glVertex3f(0.0, 0.0, 0.5);
    ... ...

gLEnd();
```

Must be convex polygon
Region filling

```c
glPolygonMode(parameter1, parameter2);
```

- **parameter2**: GL_LINE, GL_FILL
- **parameter1**: GL_FRONT, GL_BACK
Draw polygon

```c
glPolygonMode(GL_FRONT, GL_FILL);

// Draw the polygon

glBegin(GL_POLYGON);
    glVertex3f(coordinate of vertex1);
    glVertex3f(coordinate of vertex2);
    glVertex3f(coordinate of vertex3);
    glVertex3f(coordinate of vertex4);
    ... ...

glEnd();
```
Draw triangle

```cpp
glPolygonMode(GL_FRONT, GL_FILL);
glBegin(GL_TRIANGLES);
   glVertex3f(coordinate of vertex1);
   glVertex3f(coordinate of vertex2);
   glVertex3f(coordinate of vertex3);
   ...
   ...
   ...
   ...

 glEnd();
```
* Modeling using 3D mesh

Mesh representation of shape, retrieving of 3D model
3D shape acquisition

- Using mathematical models
  Sphere, torus, cylinder, etc
- Using hardware digitalization
  Range scanner
  Laser scanner
From video to 3D geometry space
From video to 3D geometry space
From video to 3D geometry space

CAS scanner
3D shape acquisition

- Using mathematical models
  Sphere, torus, cylinder, etc

- Using hardware digitalization
  Range scanner
  Laser scanner
Draw triangle

glPolygonMode(GL_FRONT, GL_FILL);

 glBegin(GL_TRIANGLES);
    glVertex3f(coordinate of vertex1);
    glVertex3f(coordinate of vertex2);
    glVertex3f(coordinate of vertex3);
    ... ...
    ... ...

 glEnd();
The first OpenGL program

Event and message driven mechanism in Windows operating system

Program run,

Enter waiting status,

“do nothing”
Course grading

- Totally \( \geq 5 \) course projects
- Select 3~5 to finish
- Graphics programming practices
  
  Discuss with TA to find an appropriate content to be implemented

  Submit workable code and demo

  or submit a technical report
Selective course project 1

- 2D nesting problem
- Problem identification
Layout algorithm

1. Given $N$ polygons, arrange them in the fabric of given width, the length of fabric is not limited.

2. Arrange these polygons from one end of fabric according to the most compact way, but cannot overlap, requiring the length $H$ of fabric is minimum.

3. While arranging polygons, there are two ways: (1) changing the orientation, (2) direction is random. Now we just consider the first approach.
Selective course project 1

- 2D nesting problem
- Problem identification
- Simplified version: Tetris?

Submit report to me before 6-May
Selective course project

• Writing a program (OpenGL, DirectX) to display a 3D mesh model

• Submit source code to TA before 6-May