MAKING LINE DRAWINGS

- Use GL_LINES as the argument to glBegin() and pass it the two endpoints as vertices.

```
glBegin(GL_LINES);
    glVertex2i(40, 100);
    glVertex2i(202, 96);
glEnd();
```

- If more than two vertices are specified between glBegin() and glEnd(), they are taken in pairs, and a separate line is drawn between each pair.

1

Making LINE Drawing

- Use GL_LINES as the argument to the glBegin(), and pass it the two endpoints as vertices.
- Thus to draw a line between (40, 100) and (202, 96), use the following:

```
glBegin(GL_LINES);
        glVertex2i(40, 100);
        glVertex2i(202, 96);
glEnd();
```

- This code might be encapsulated for convenience in the routine drawLineInt():

```
void drawLineInt(GLint x1, GLint y1, GLint x2, GLint y2) {
        glBegin(GL_LINES);
                glVertex2i(x1, y1);
                glVertex2i(x2, y2);
        glEnd();
}
```

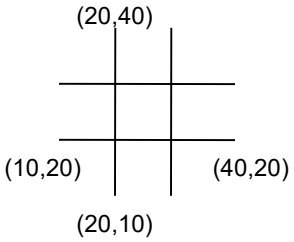
- An alternate routine drawLineFloat() could be implemented similarly.

2

Making Line Drawings

- If more than two vertices are specified between glBegin(GL_LINES) and glEnd(), they are taken in pairs, and a separate line is drawn between each pair.

```
glBegin(GL_LINES);
        glVertex2i(10, 20);
        glVertex2i(40, 20);
        glVertex2i(20, 10);
        glVertex2i(20, 40);
// more lines here
glEnd();
glFlush();
```



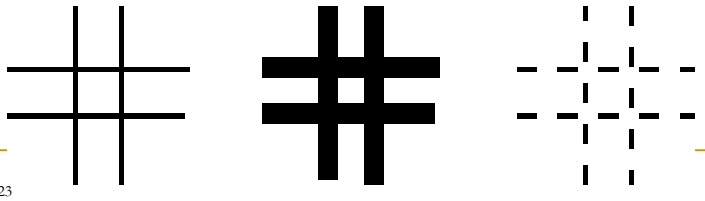
- OpenGL provides tools for setting the attributes of lines
  - A line's color is set using glColor3f();
  - A line's thickness is set by glLineWidth(4.0) to four pixels. The default thickness is 1.0.
  - Dotted, dashed and stippled lines can also be drawn. (Case Study..)

3

Line Attributes

- Color, thickness, stippling.
- glColor3f() sets color.
- glLineWidth(4.0) sets thickness. The default thickness is 1.0.

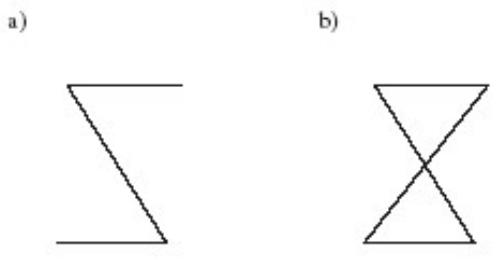
a). thin lines b). thick lines c). stippled lines



4

Setting Line Parameters

- Polylines and Polygons: lists of vertices.
- Polygons are closed (right); polylines need not be closed (left).

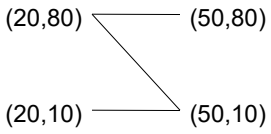


5

Drawing polylines and polygons

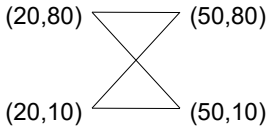
- A polyline is a collection of line segments joined end to end.
- In OpenGL, a polyline is called a "line strip" and is drawn as:

```
glBegin(GL_LINE_STRIP);
        glVertex2i(20, 10);
        glVertex2i(50, 10);
        glVertex2i(20, 80);
        glVertex2i(50, 80);
glEnd();
```



- Attribues such as color, thickness, and stippling can be applied to polylines in the same way as they are applied to single lines.
- If it is desired to connect the last point with the first point to make the polyline into a polygon, simply replace GL_LINE_STRIP with GL_LINE_LOOP.

```
glBegin(GL_LINE_LOOP);
        glVertex2i(20, 10);
        glVertex2i(50, 10);
        glVertex2i(20, 80);
        glVertex2i(50, 80);
glEnd();
```



- Such polygons cannot be filled with a color or pattern.

6

Drawing polylines and polygons

Example: Drawing line graphs

- A line graph is straight forward extension of the "dot plot" example.
- Suppose we have the following function to plot:

  $f(x) = 300 - 100 \cos(2\pi x/100) + 30 \cos(4\pi x/100) + 6 \cos(6\pi x/100)$

  as x varies in steps of 3 for 100 steps.
- As a blowup of this figure would show a sequence of connected line segments; in a normal sized picture, they blend to give an impression of a smoothly varying curve.
- We need to do two changes in the code of example 2.2.4.
  1. Calculate the scaling and shifting coefficients A, B, C and D appropriately for the above function.
  2. Instead of using GL_POINTS, use GL_LINE_STRIP.
- The rest of the code remains the same.

Drawing polylines and polygons

Example: Drawing polylines stored in a file (1/2)

- Most interesting pictures made up of polylines contain a rather large number of line segments.
- It is not hard to write a routine that draws the polylines stored in a file.
- A typical file format could be:

```
21          Number of polylines in the file
4           number of points in the first polyline
169 118     first point of the first polyline
174 120     second point of the first polyline
179 124
178 126
5           number of points in the second polyline
298 86      first point of the second polyline
304 92
310 104
314 114
314 119
29
32 435
10 439
```

Drawing polylines and polygons

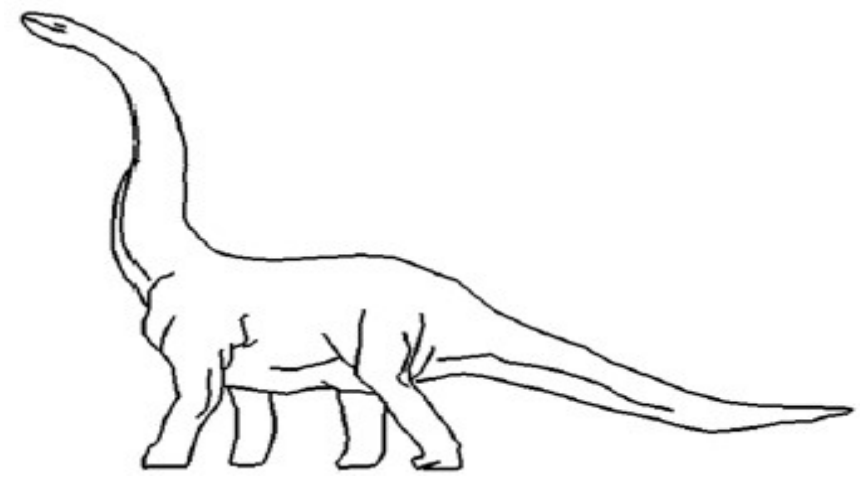
Example: Drawing polylines stored in a file 2/2)

- A simple routine (without any error checks) could be written as follows, to read from such a data file.

```
#include <fstream.h>
Void drawPolyLineFile(char *fileName) {
    fstream inStream;
    inStream.open(fileName, ios::in);                    // open the file
    if (inStream.fail() ) return;
    glClear(GL_COLOR_BUFFER_BIT);                        // clear the screen
    GLint numpolys, numLines, x, y;
    inStream >> numpolys;                                // read the number of polylines
    for (int j=0; j < numpolys;, j++) {
        inStream >> numLines;                            // read the number of lines in a polyline
        glBegin(GL_LINE_STRIP);
        for (int i=0; i<numLines; i++) {
            inStream >> x >> y;                          // read the next x, y pair
            glVertex2i(x, y);
        }
        glEnd();
    }
    glFlush();
    inStream.close();
}
```


9

Drawing polylines and polygons



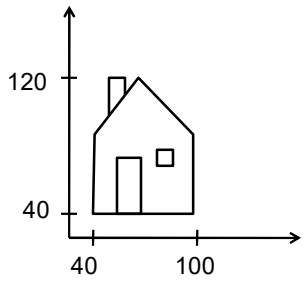
10

Drawing polylines and polygons

Example: Parametrizing Figures (1/3)

- The figure below shows a simple house consisting of a few polylines.
- It can be drawn using the code shown below.

```
void hardwiredHouse(void) {
 glBegin(GL_LINE_STRIP);
    glVertex2i( 40, 40);  // draw the shell of the house
    glVertex2i( 40, 90);
    glVertex2i( 70, 120);
    glVertex2i( 100, 90);
    glVertex2i( 100, 40);
 glEnd();
 glBegin(GL_LINE_STRIP);
    glVertex2i( 50, 100);  // draw the chimney
    glVertex2i( 50, 120);
    glVertex2i( 60, 120);
    glVertex2i( 60, 110);
 glEnd();
..... // draw the door
..... // draw the window
}
```



This is not a very flexible approach.
Only one house, with fixed size and location can be drawn with this approach.

11

Drawing polylines and polygons

Example: Parametrizing Figures (2/3)

- More flexibility can be achieved if we parameterize the figure and pass these parameters to the routine.

```
void parameterizedHouse(GLintPoint peak, GLintPoint widthm GLintPoint height)
// the top of the house is at the peak; the size of house is given by
// the height and width of the house
{
 glBegin(GL_LINE_LOOP);
    glVertex2i( peak.x, peak.y);  // draw the shell of the house
    glVertex2i( peak.x + width/2, peak.y – 3 * height / 3);
    glVertex2i( peak.x + width/2, peak.y – height);
    glVertex2i( peak.x – width/2, peak.y – height);
    glVertex2i( peak.x – width/2, peak.y – 3 * height / 3);
 glEnd();
.....  // draw the chimney in the same fashion
..... // draw the door
..... // draw the window
}
```


12

Assignment

Parametrizing Figures (3/3)

- We can draw families of objects, by passing different parameter values.
- The routine can be used to draw a village of houses, as shown below.