OpenGL

CS418 Computer Graphics

John C. Hart

OpenGL

- Based on GL (graphics library) by Silicon Graphics Inc. (SGI)

Advantages:

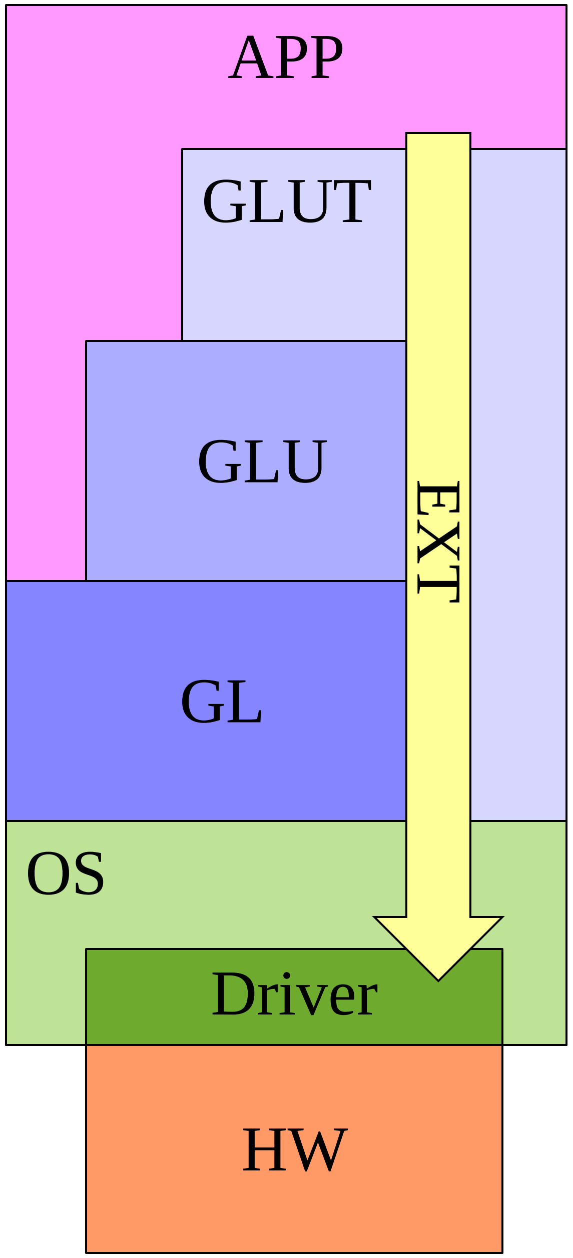
- Runs on everything, including cell phones (OpenGL/ES)

Alternatives:

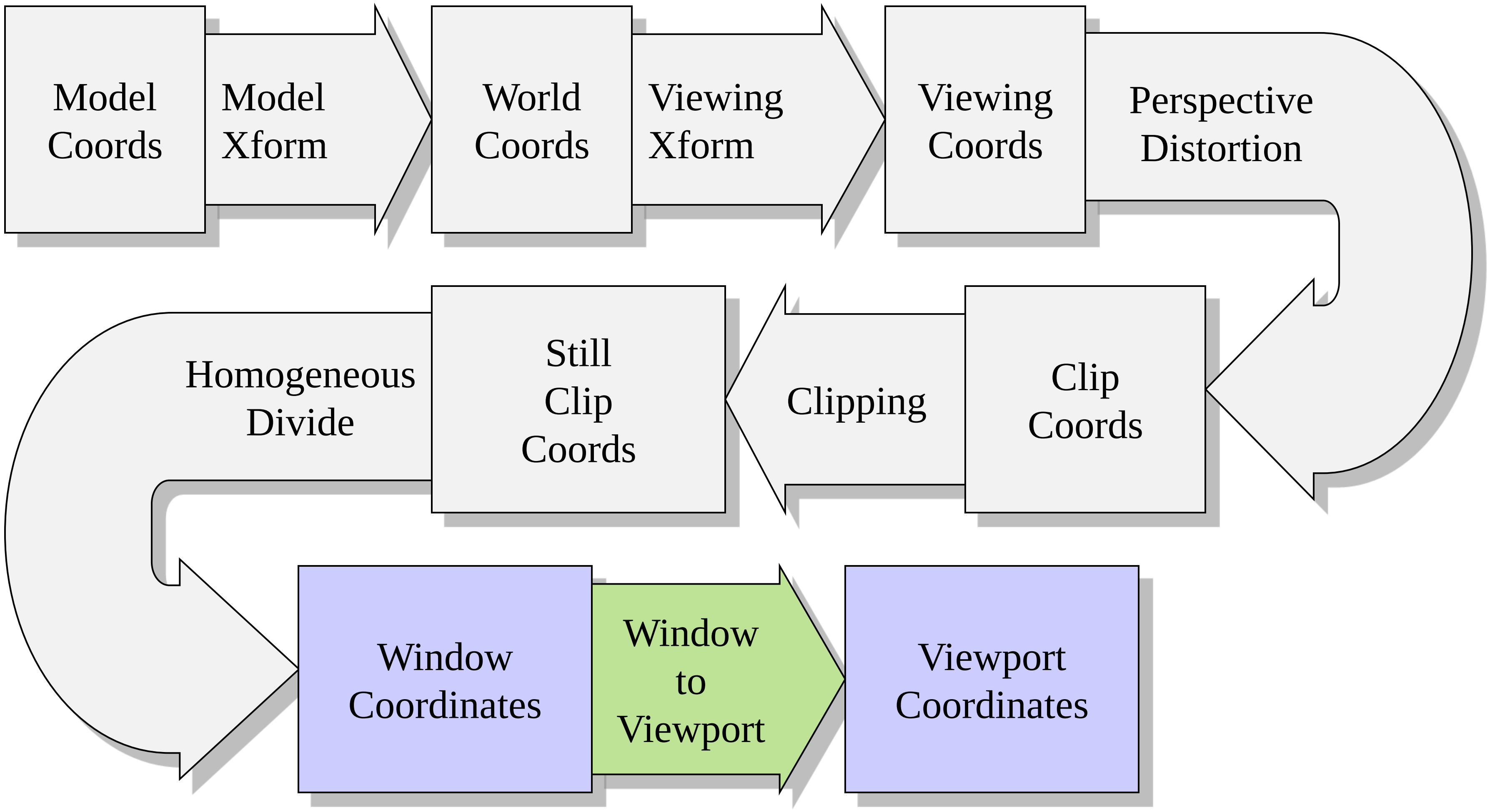
- Microsoft’s Direct3D – limited to MSWindows
- Sun’s Java3D – slower, implemented on top of OpenGL

Library Layers

- OpenGL = GL + GLU
  - Basic low-level GL routines implemented using OS graphics routines
  - Timesaving higher-level GLU routines implemented using GL routines
- GLUT opens and manages OpenGL windows and adds helper functions
- OpenGL Extensions provide direct device-dependent access to hardware



Vertex Pipeline



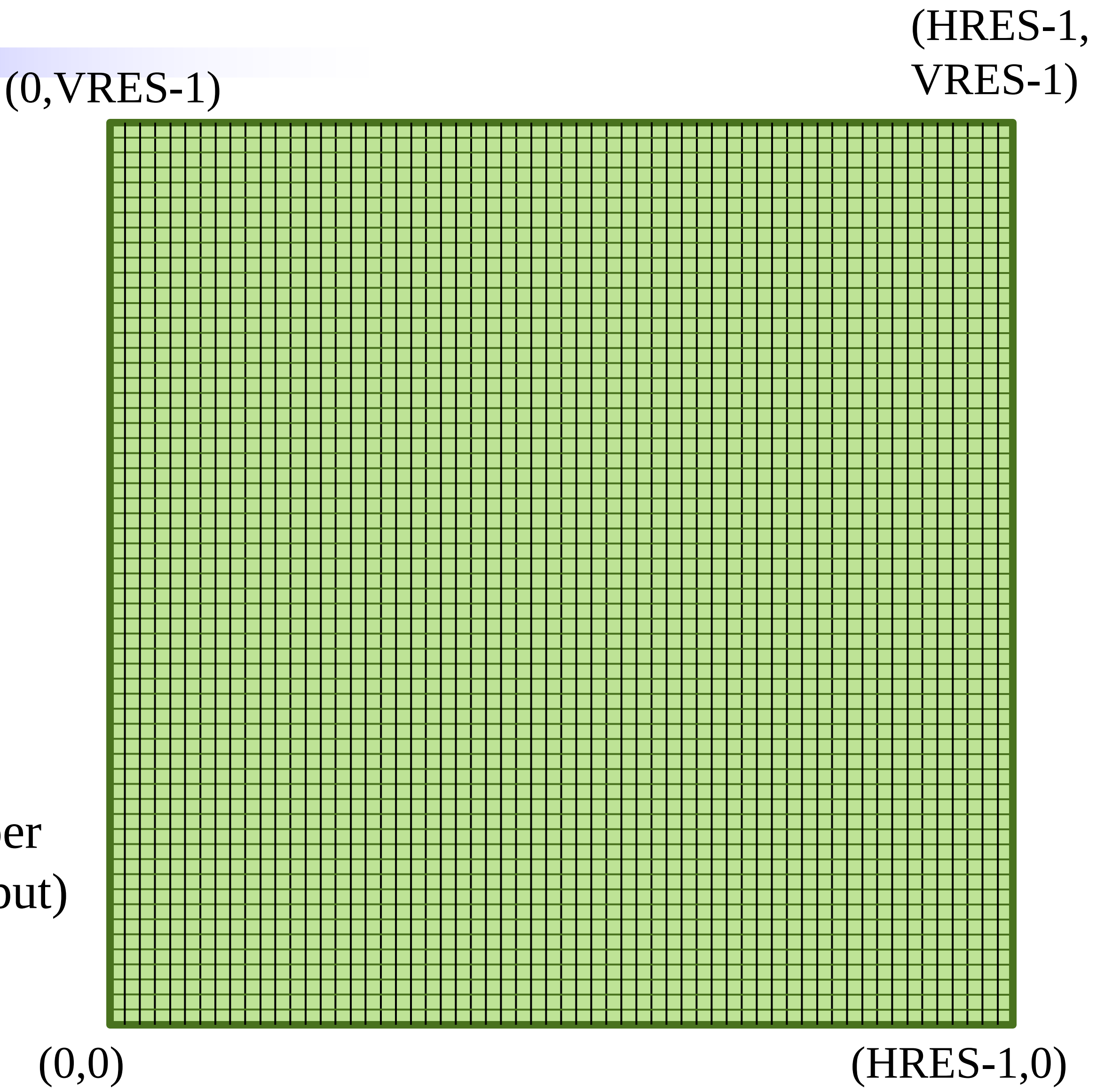
Viewport Coordinates

- Physical per-pixel integer coordinates
- Also called screen or device coordinates

glViewport(x,y,w,h)

- x,y – lower left pixel
- w – width
- h – height

- Sometimes (0,0) is in the upper left corner (e.g. for mouse input)



Window Coordinates

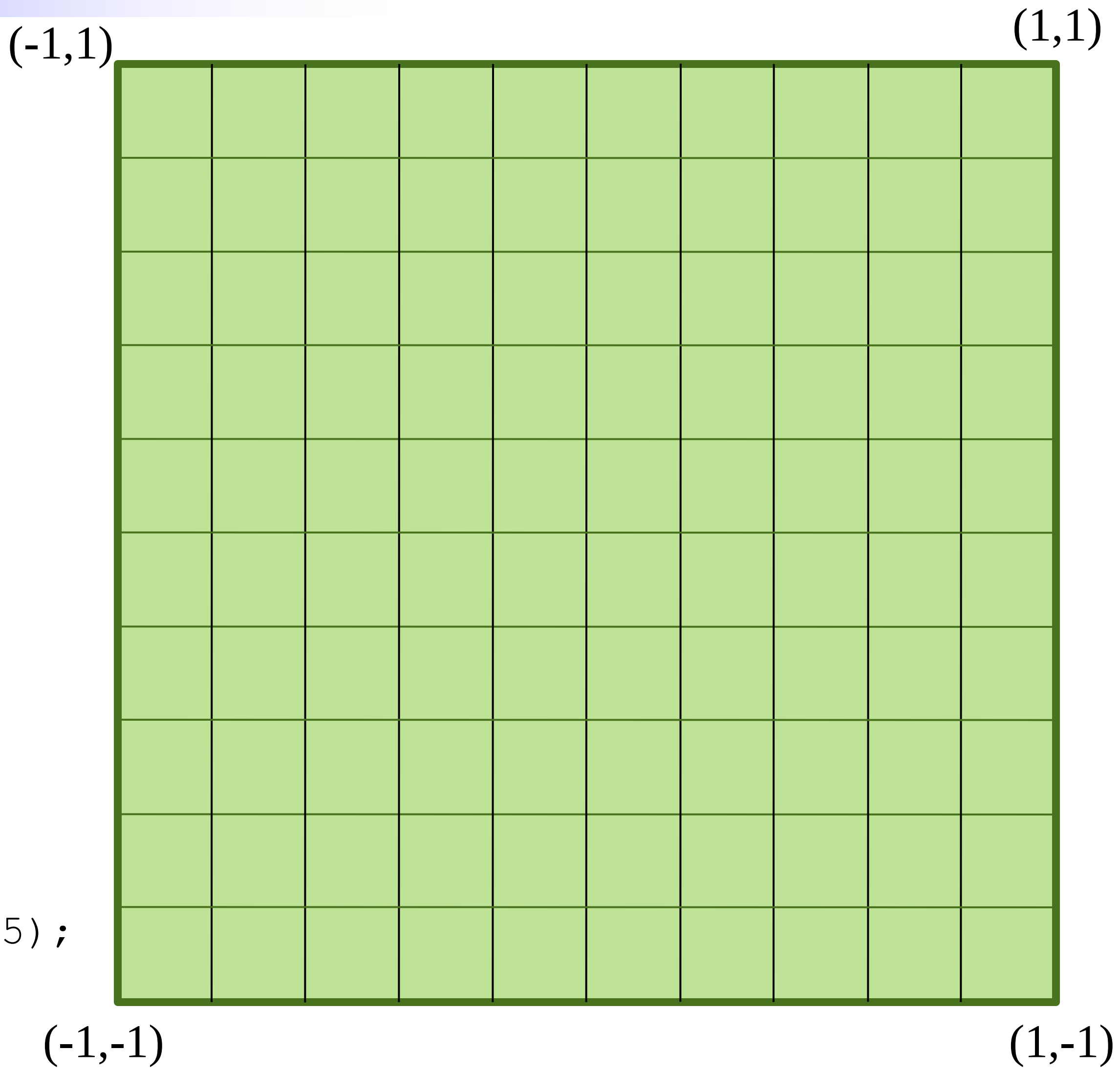
- Logical, mathematical floating-point coordinates

glOrtho(l,r,b,t,n,f)

- left, right, bottom, top
- near, far: limits depth
- gluOrtho2D(l,r,b,t) calls glOrtho(l,r,b,t,-1,1)

- To use per-pixel coordinates, call:

```
gluOrtho2D(-.5,-.5,w-.5,h-.5);
```



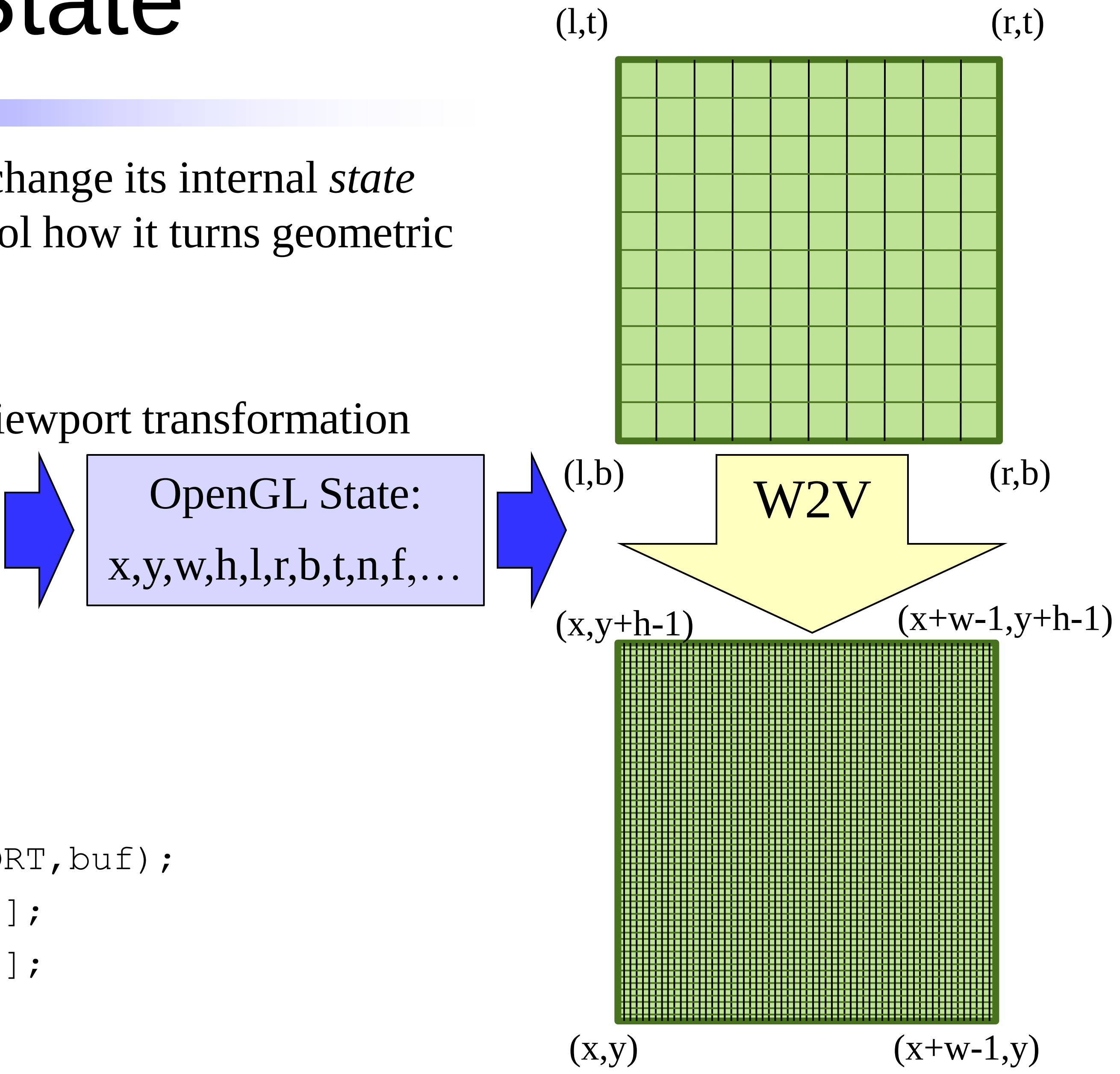
OpenGL State

- OpenGL commands change its internal *state* variables which control how it turns geometric models into pictures

- E.g. the window-to-viewport transformation

  glViewport(x,y,w,h)

  glOrtho(l,r,b,t,n,f)



- Can query this state:

```
GLfloat buf[4];
glGetFloatv(GL_VIEWPORT,buf);
x = buf[0]; y = buf[1];
w = buf[2]; h = buf[3];
```

OpenGL Commands

```
gl[u]Fubar[234][dfis][v]
```

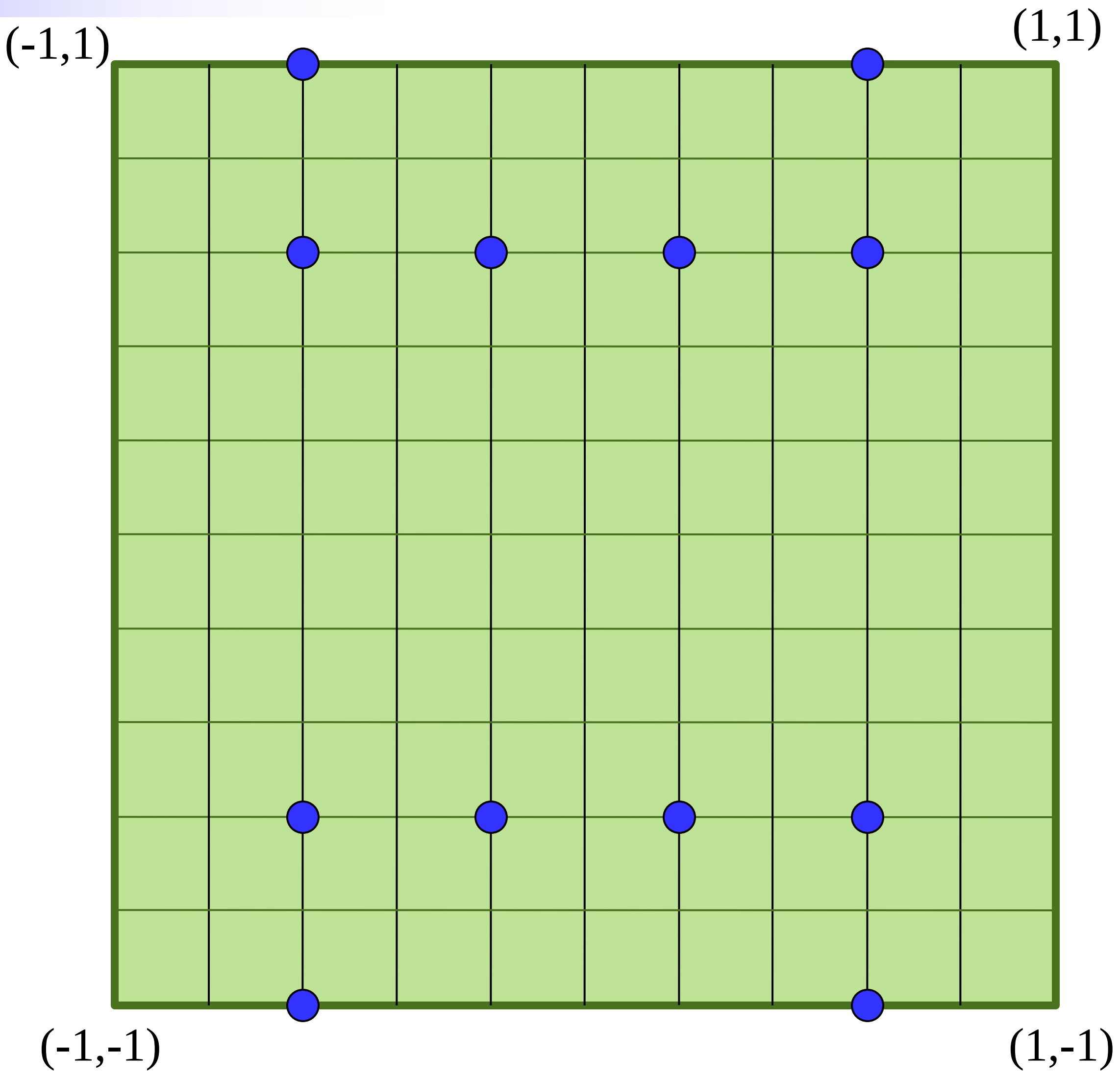
- **[u]**: GLU v. GL command
- **[234]**: dimension
- **[dfisb]**: data type
  - **d**: GLdouble (double)
  - **f**: GLfloat (float)
  - **i**: GLint (int)
  - **s**: string
- **[v]**: vector (reference v. value)

Examples:

```
glVertex3f(-0.5, 3.14159, 2);
glVertex2i(200, 350);

GLdouble point[4];
glVertex4dv(point);
```

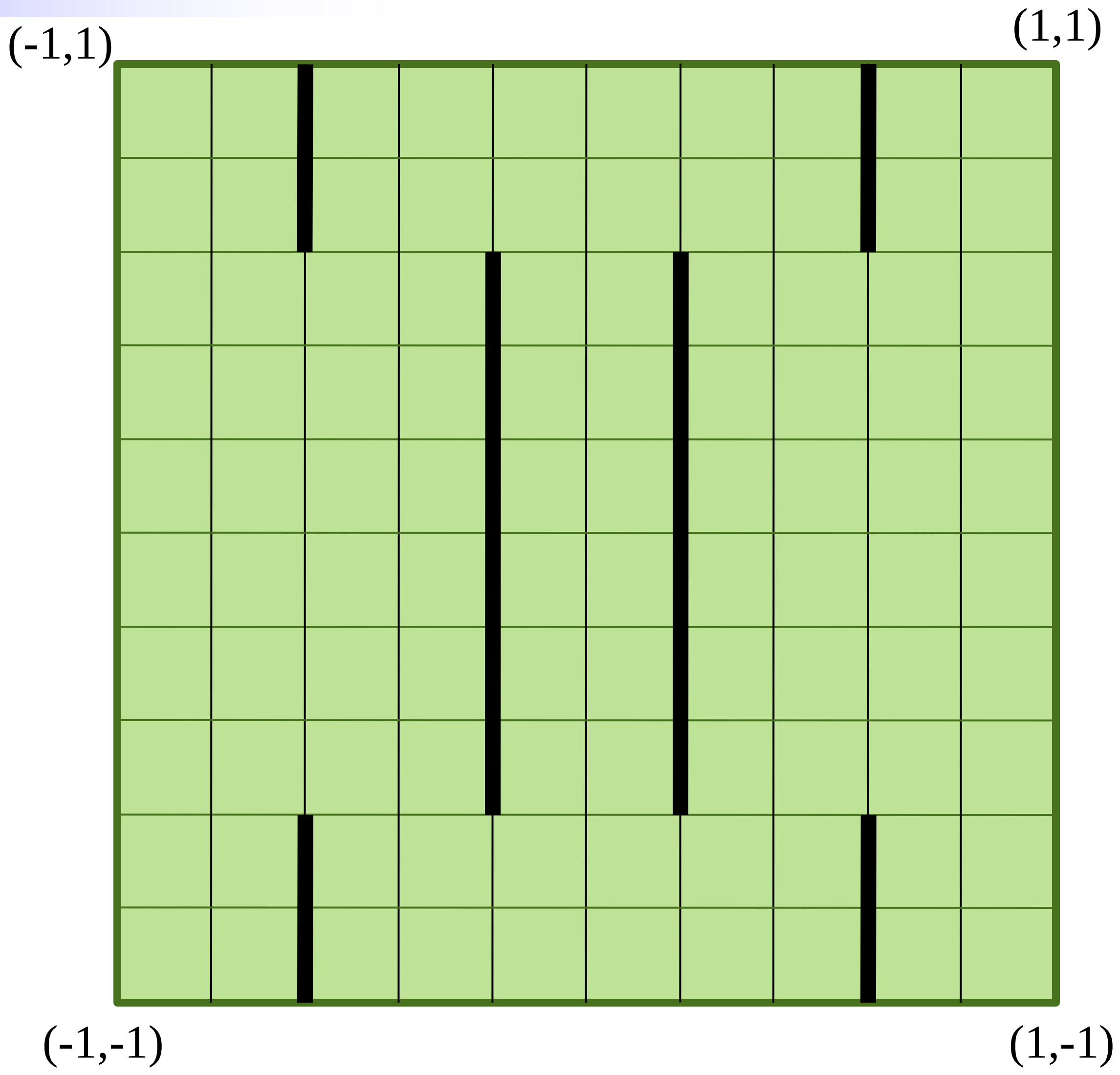
Points

```
glBegin(GL_POINTS);
  glVertex2f(-.6,1.);
  glVertex2f(-.6,.6);
  glVertex2f(-.2,.6);
  glVertex2f(-.2,-.6);
  glVertex2f(-.6,-.6);
  glVertex2f(-.6,-1.);
  glVertex2f(.6,-1.);
  glVertex2f(.6,-.6);
  glVertex2f(.2,-.6);
  glVertex2f(.2,.6);
  glVertex2f(.6,.6);
  glVertex2f(.6,1.);
glEnd();
```



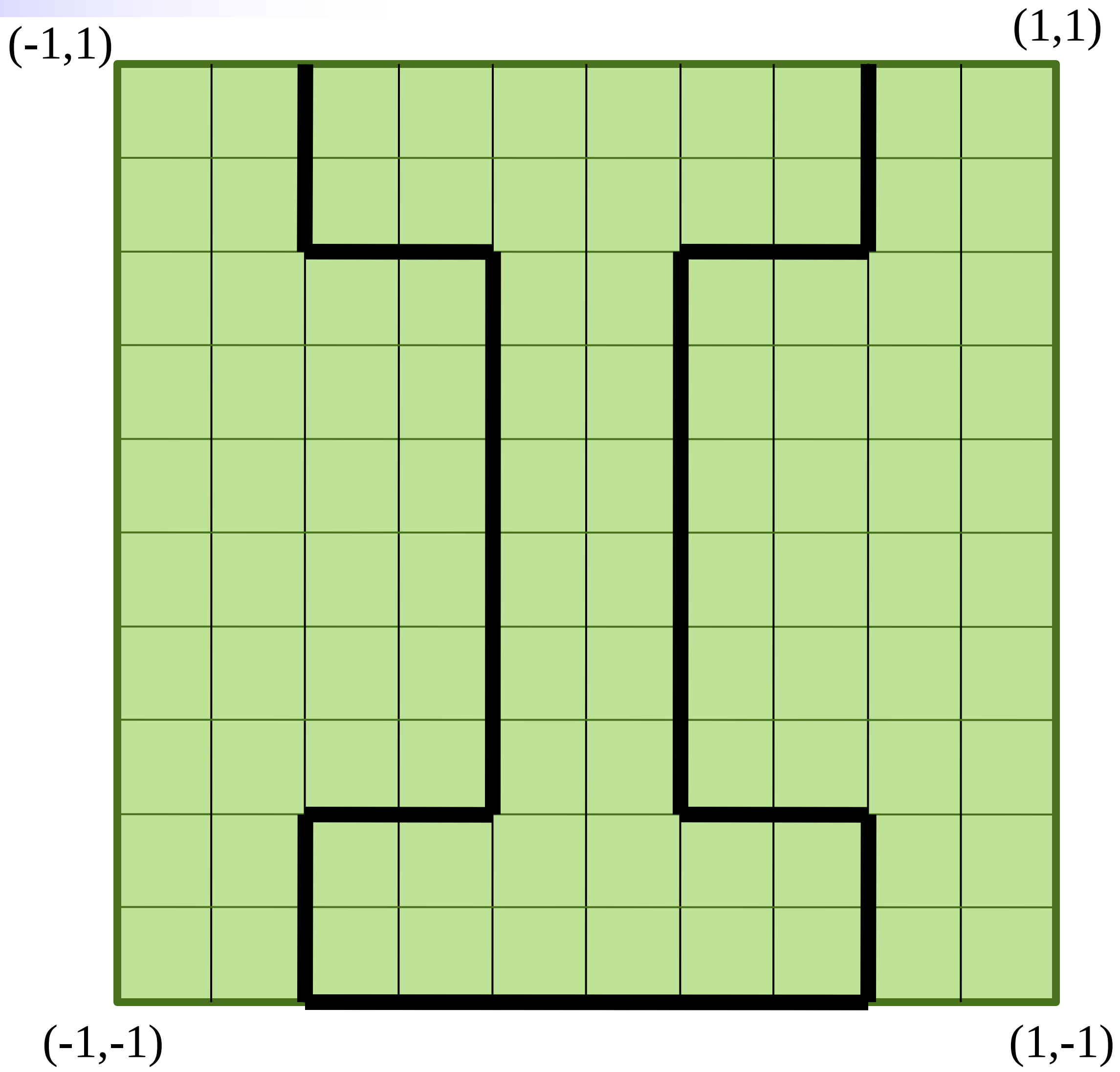
Lines

```
glBegin(GL_LINES);
  glVertex2f(-.6,1.);
  glVertex2f(-.6,.6);
  glVertex2f(-.2,.6);
  glVertex2f(-.2,-.6);
  glVertex2f(-.6,-.6);
  glVertex2f(-.6,-1.);
  glVertex2f(.6,-1.);
  glVertex2f(.6,-.6);
  glVertex2f(.2,-.6);
  glVertex2f(.2,.6);
  glVertex2f(.6,.6);
  glVertex2f(.6,1.);
glEnd();
```



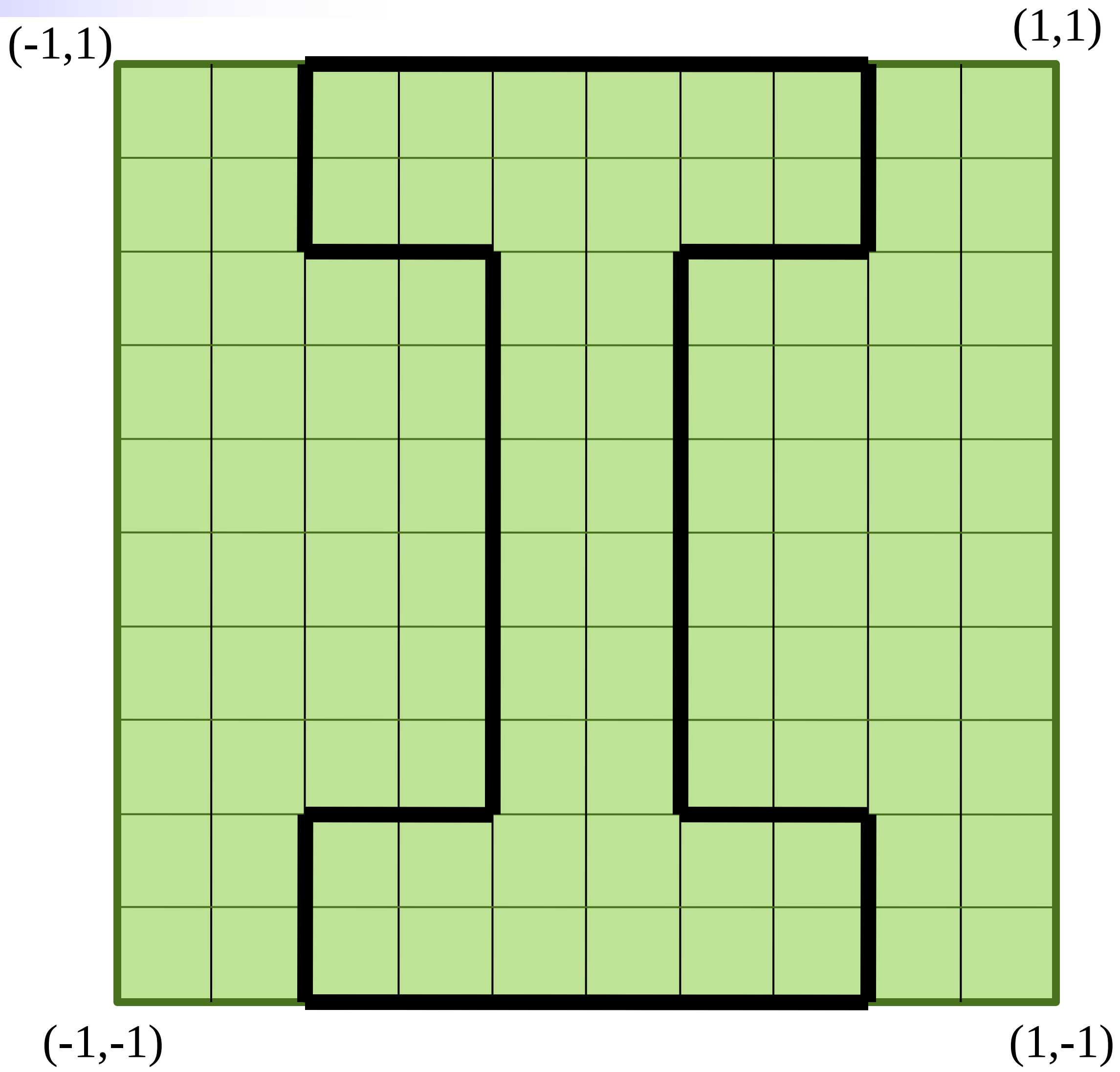
Line Strip

```
glBegin(GL_LINE_STRIP);
  glVertex2f(-.6,1.);
  glVertex2f(-.6,.6);
  glVertex2f(-.2,.6);
  glVertex2f(-.2,-.6);
  glVertex2f(-.6,-.6);
  glVertex2f(-.6,-1.);
  glVertex2f(.6,-1.);
  glVertex2f(.6,-.6);
  glVertex2f(.2,-.6);
  glVertex2f(.2,.6);
  glVertex2f(.6,.6);
  glVertex2f(.6,1.);
glEnd();
```



Line Loop

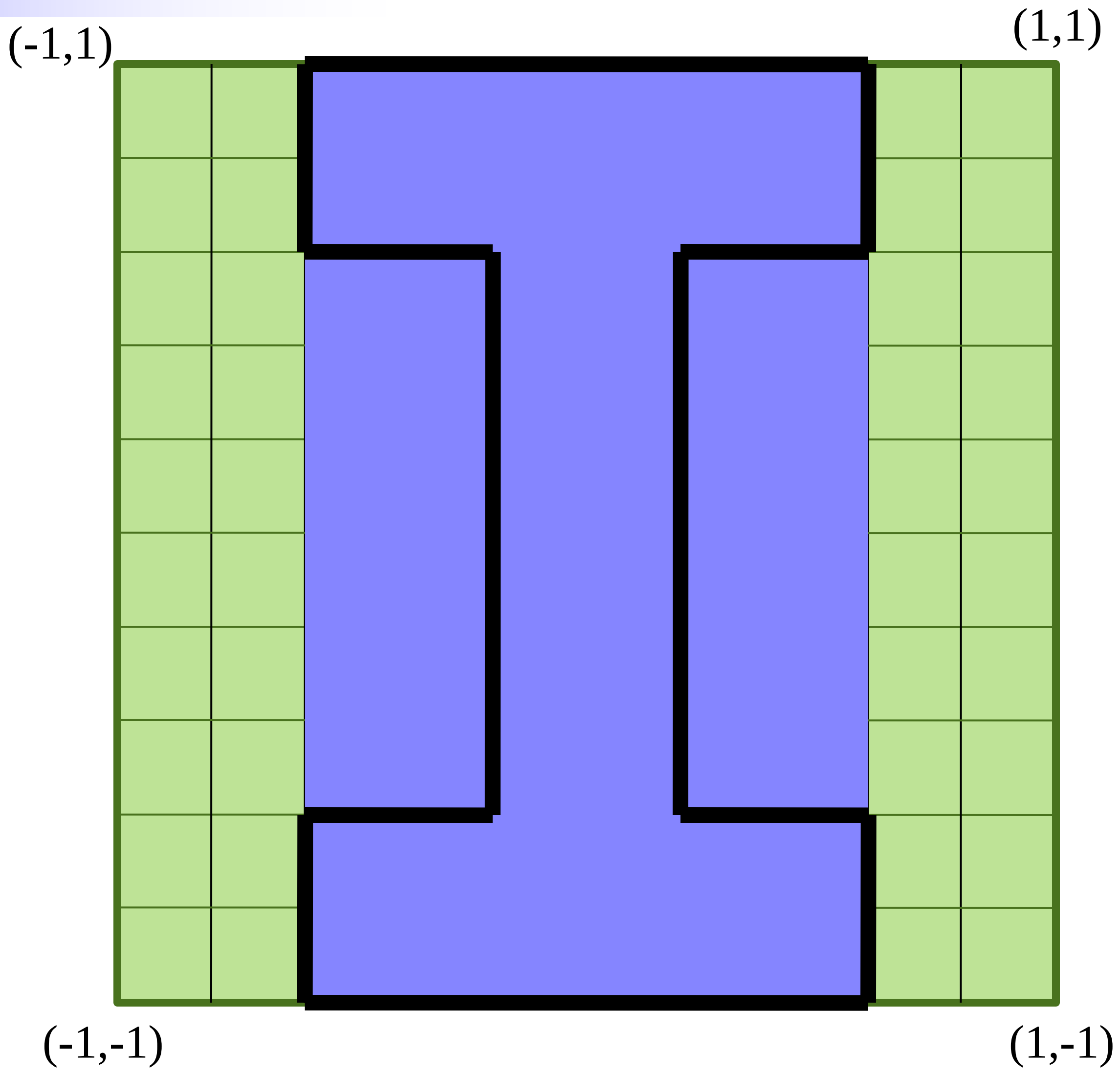
```
glBegin(GL_LINE_LOOP);
  glVertex2f(-.6,1.);
  glVertex2f(-.6,.6);
  glVertex2f(-.2,.6);
  glVertex2f(-.2,-.6);
  glVertex2f(-.6,-.6);
  glVertex2f(-.6,-1.);
  glVertex2f(.6,-1.);
  glVertex2f(.6,-.6);
  glVertex2f(.2,-.6);
  glVertex2f(.2,.6);
  glVertex2f(.6,.6);
  glVertex2f(.6,1.);
glEnd();
```



Polygon

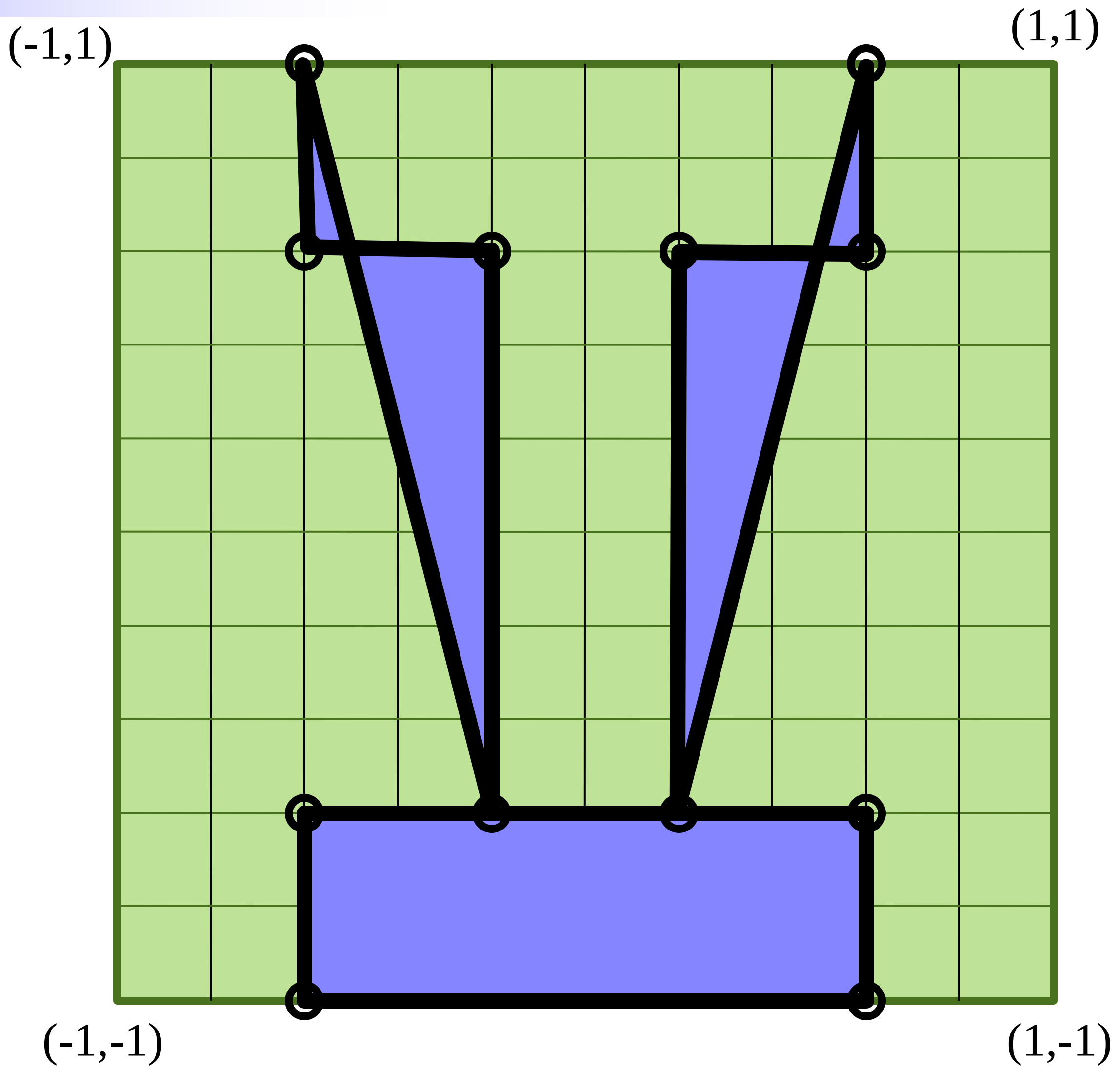
OpenGL only supports **convex polygons**
(and really only triangles)

```
glBegin(GL_POLYGON);
  glVertex2f(-.6,1.);
  glVertex2f(-.6,.6);
  glVertex2f(-.2,.6);
  glVertex2f(-.2,-.6);
  glVertex2f(-.6,-.6);
  glVertex2f(-.6,-1.);
  glVertex2f(.6,-1.);
  glVertex2f(.6,-.6);
  glVertex2f(.2,-.6);
  glVertex2f(.2,.6);
  glVertex2f(.6,.6);
  glVertex2f(.6,1.);
glEnd();
```



Quads

```
glBegin(GL_QUADS);
  glVertex2f(-.6,1.);
  glVertex2f(-.6,.6);
  glVertex2f(-.2,.6);
  glVertex2f(-.2,-.6);
  glVertex2f(-.6,-.6);
  glVertex2f(-.6,-1.);
  glVertex2f(.6,-1.);
  glVertex2f(.6,-.6);
  glVertex2f(.2,-.6);
  glVertex2f(.2,.6);
  glVertex2f(.6,.6);
  glVertex2f(.6,1.);
glEnd();
```



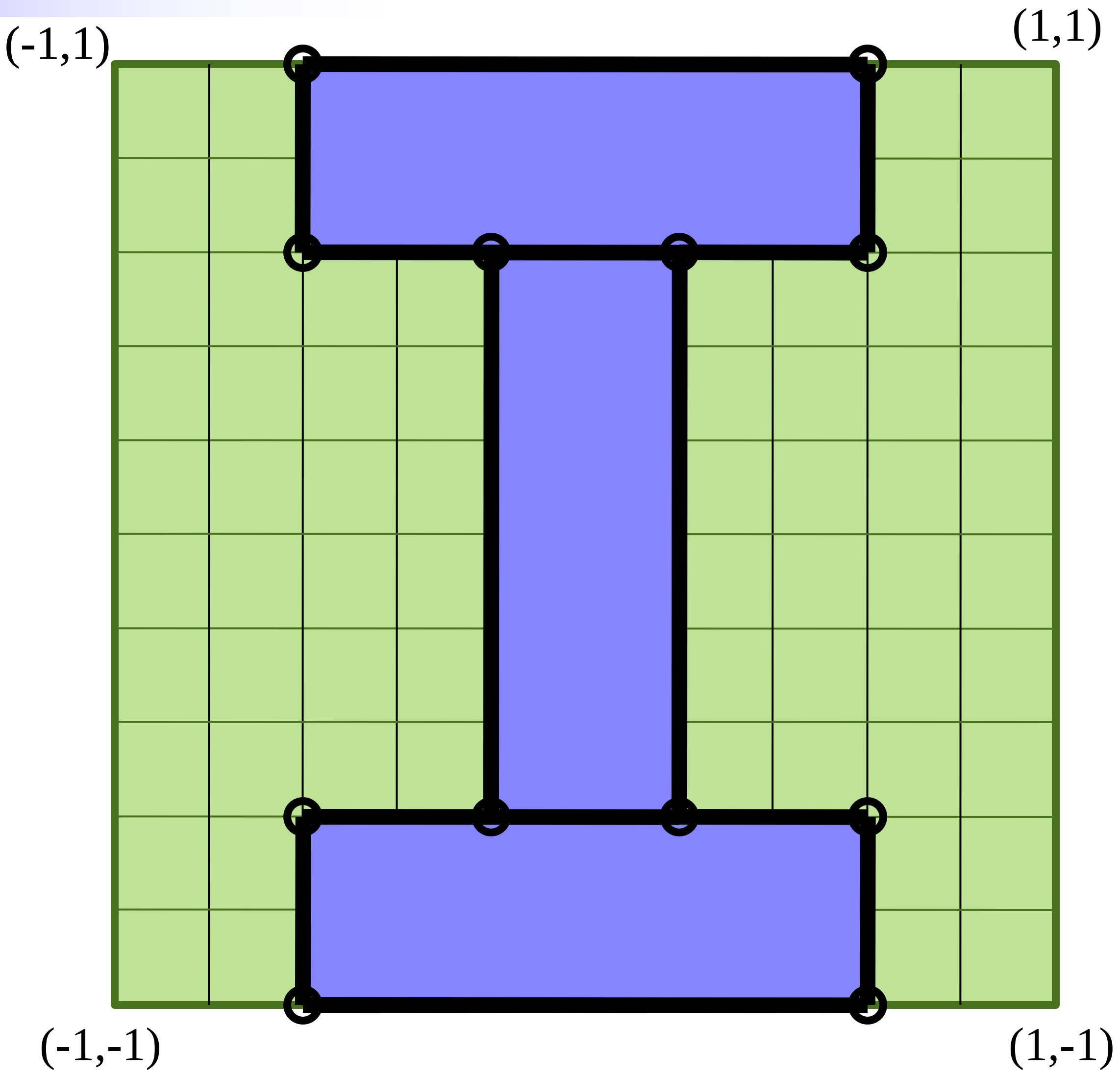
Quads

```
glBegin(GL_QUADS);
  glVertex2f(-.6,1.);
  glVertex2f(-.6,.6);
  glVertex2f(-.2,.6);
  glVertex2f(-.2,-.6);
  glVertex2f(-.6,-.6);
  glVertex2f(-.6,-1.);
  glVertex2f(.6,-1.);
  glVertex2f(.6,-.6);
  glVertex2f(.2,-.6);
  glVertex2f(.2,.6);
  glVertex2f(.6,.6);
  glVertex2f(.6,1.);
glEnd();
```

Quads

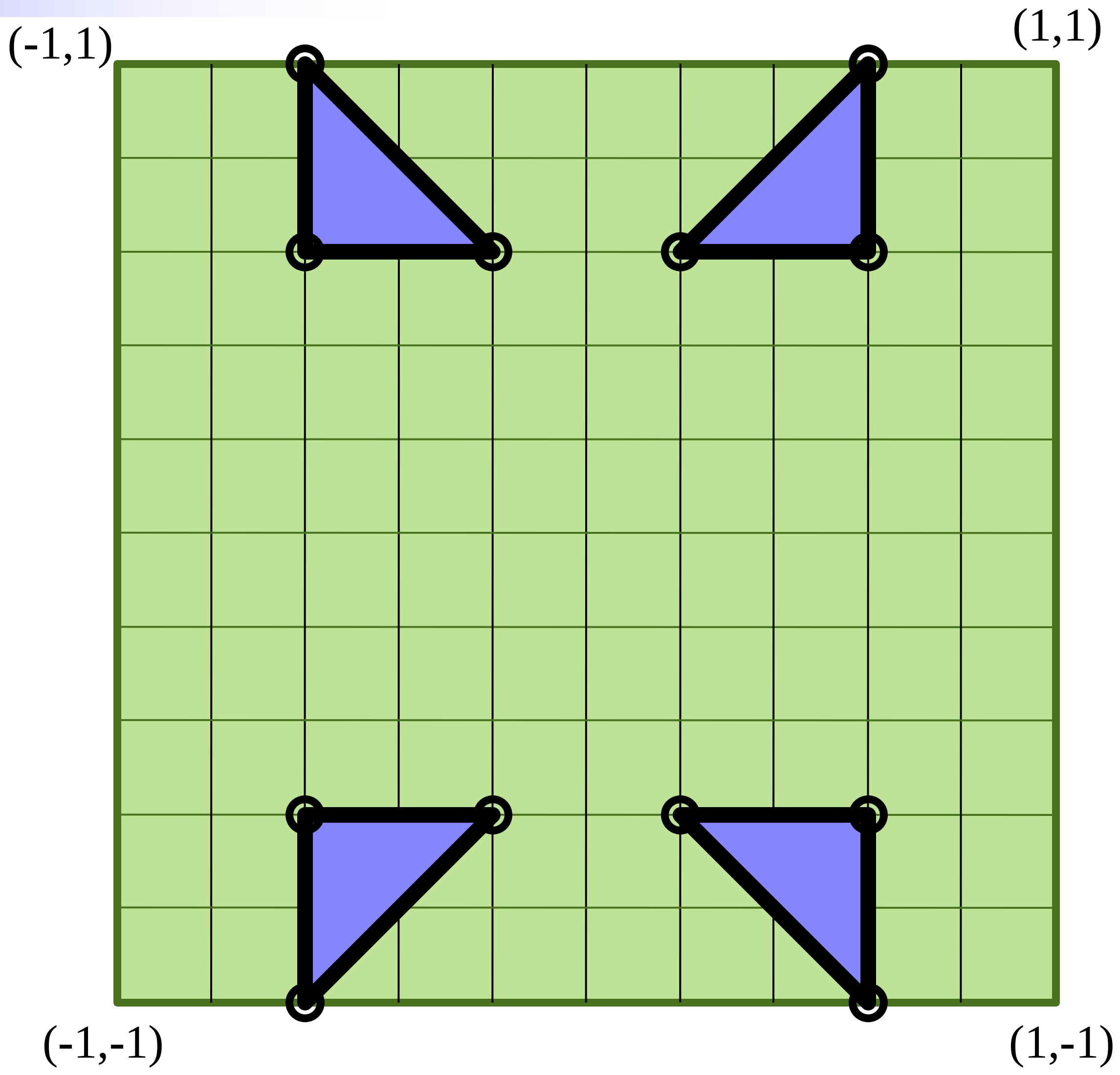
Lines should **never** pass through a vertex.

```
glBegin(GL_QUADS);
  glVertex2f(-.6,1.);
  glVertex2f(-.6,.6);
  glVertex2f(.6,.6);
  glVertex2f(.6,1.);
  glVertex2f(-.6,-.6);
  glVertex2f(-.6,-1.);
  glVertex2f(.6,-1.);
  glVertex2f(.6,-.6);
  glVertex2f(-.2,.6);
  glVertex2f(-.2,-.6);
  glVertex2f(.2,-.6);
  glVertex2f(.2,.6);
glEnd();
```



Triangles

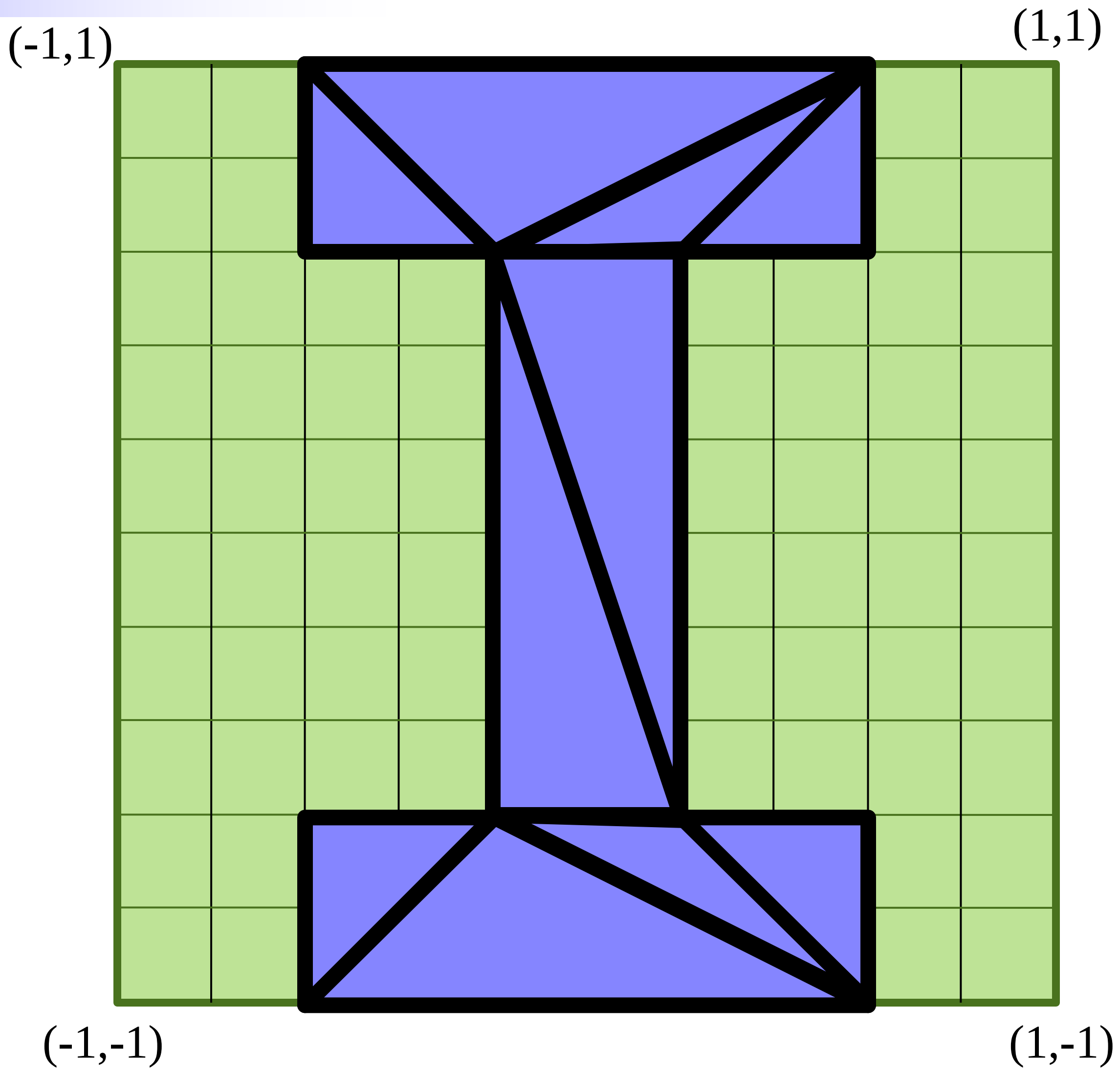
```
glBegin(GL_TRIANGLES);
  glVertex2f(-.6,1.);
  glVertex2f(-.6,.6);
  glVertex2f(-.2,.6);

  glVertex2f(-.2,-.6);
  glVertex2f(-.6,-.6);
  glVertex2f(-.6,-1.);

  glVertex2f(.6,-1.);
  glVertex2f(.6,-.6);
  glVertex2f(.2,-.6);

  glVertex2f(.2,.6);
  glVertex2f(.6,.6);
  glVertex2f(.6,1.);
glEnd();
```



Triangles

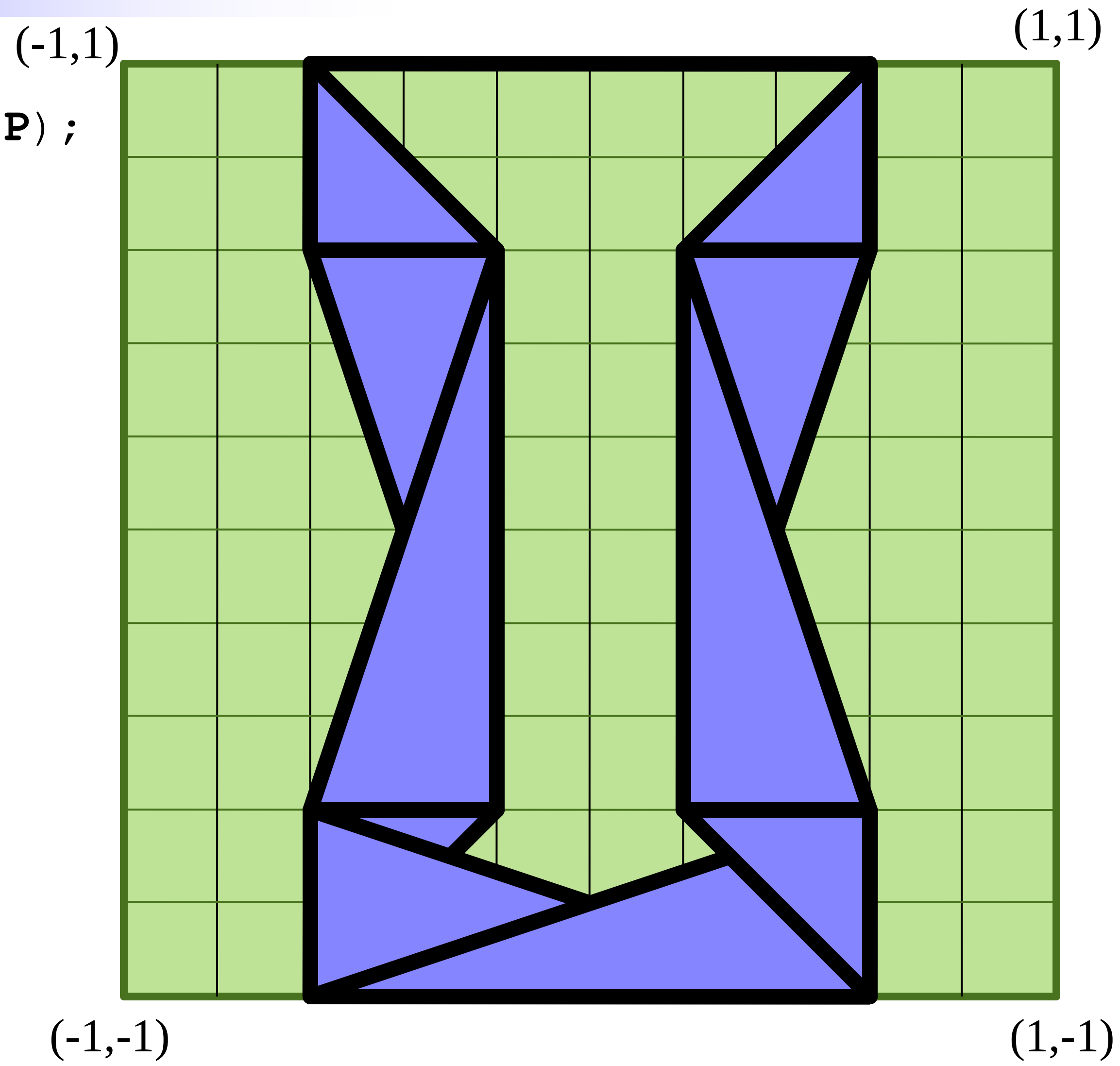
Lines should **never** pass through a vertex.

```
glBegin(GL_TRIANGLES);
  glVertex2f(-.6,1.);
  glVertex2f(-.6,.6);
  glVertex2f(-.2,.6);

  glVertex2f(-.6,1.);
  glVertex2f(-.2,.6);
  glVertex2f(.6,1.);

  glVertex2f(-.2,.6);
  glVertex2f(.2,.6);
  glVertex2f(.6,1.);

  glVertex2f(.2,.6);
  glVertex2f(.6,.6);
  glVertex2f(.6,1.);
  …
glEnd();
```



Triangle Strip

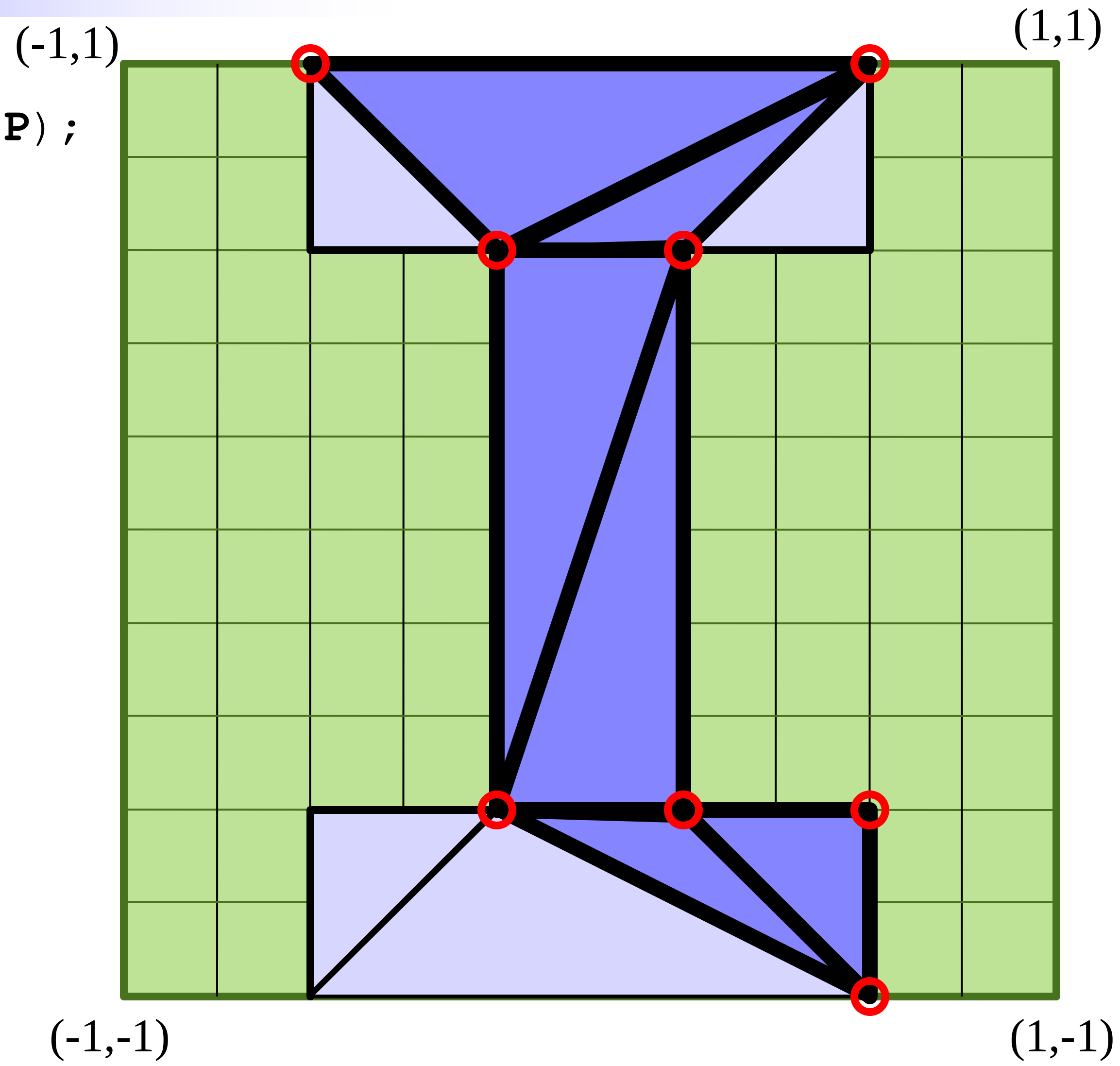
```
glBegin(GL_TRIANGLE_STRIP);
  glVertex2f(-.6,1.);
  glVertex2f(-.6,.6);
  glVertex2f(-.2,.6);
  glVertex2f(-.2,-.6);
  glVertex2f(-.6,-.6);
  glVertex2f(-.6,-1.);
  glVertex2f(.6,-1.);
  glVertex2f(.6,-.6);
  glVertex2f(.2,-.6);
  glVertex2f(.2,.6);
  glVertex2f(.6,.6);
  glVertex2f(.6,1.);
glEnd();
```



Triangle Strip

First two vertices prime the pump,
then every new vertex creates a triangle
connecting it to the previous two vertices

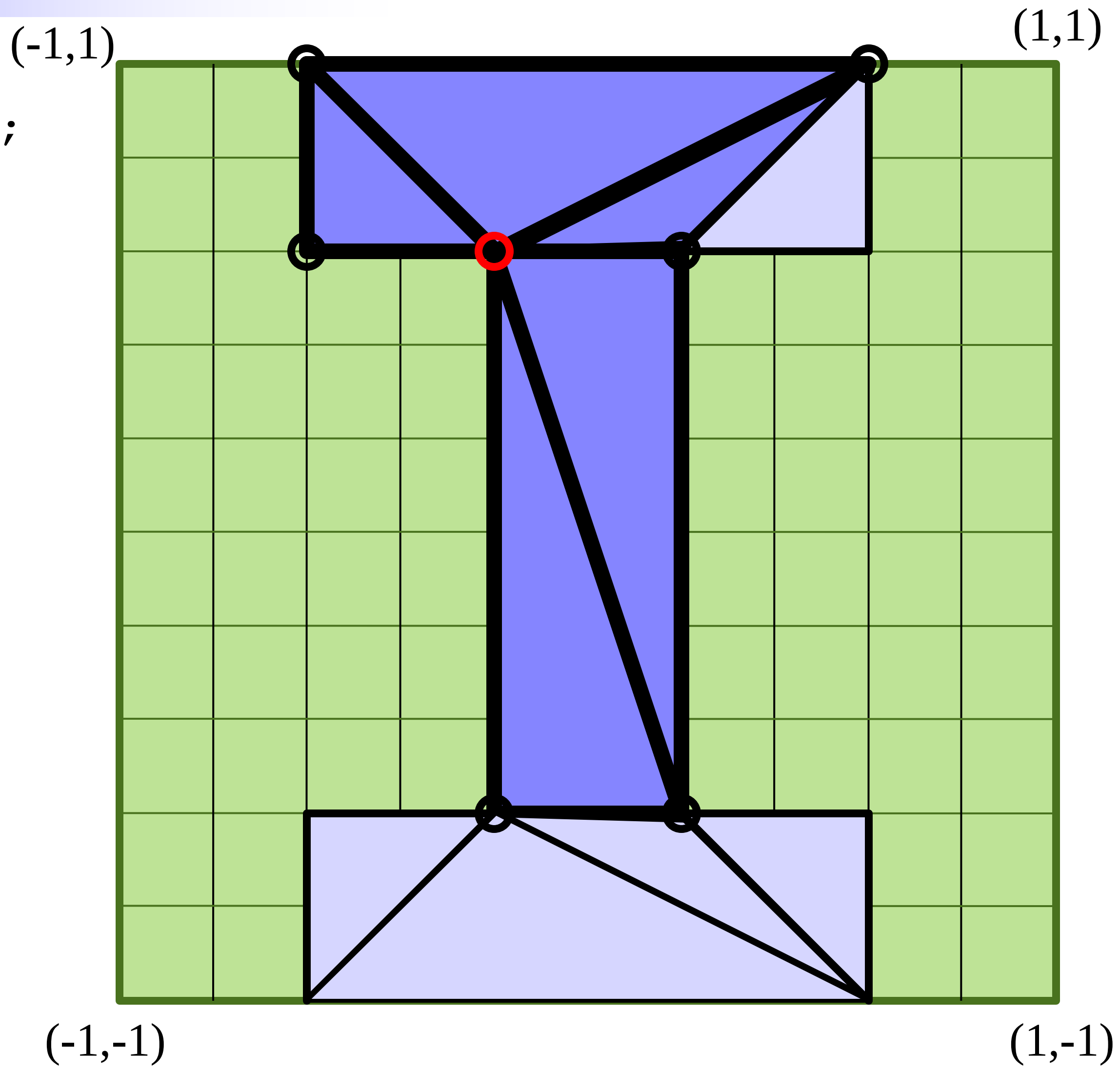
```
glBegin(GL_TRIANGLE_STRIP);
  glVertex2f(-.6,1.);
  glVertex2f(.6,1.);
  glVertex2f(-.2,.6);
  glVertex2f(.2,.6);
  glVertex2f(-.2,-.6);
  glVertex2f(.2,-.6);
  glVertex2f(.6,-1.);
  glVertex2f(.6,-.6);
glEnd();
…
```



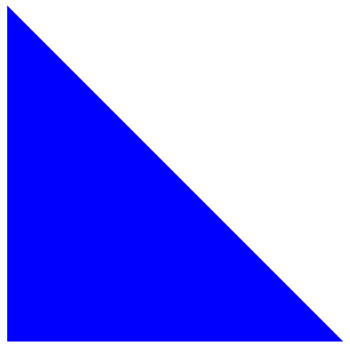
Triangle Fan

First two vertices prime the pump, then every new vertex creates a triangle connecting it to the previous vertex and the first vertex

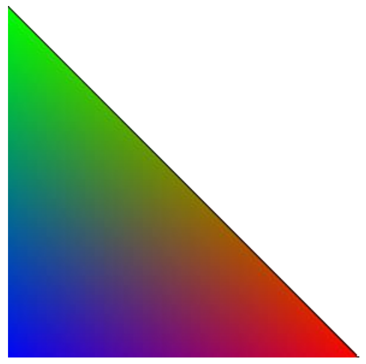
```
glBegin(GL_TRIANGLE_FAN);
  glVertex2f(-.2,.6);
  glVertex2f(-.6,.6);
  glVertex2f(-.6,1.);
  glVertex2f(.6,1.);
  glVertex2f(.2,.6);
  glVertex2f(.2,-.6);
  glVertex2f(-.2,-.6);
glEnd();
…
```



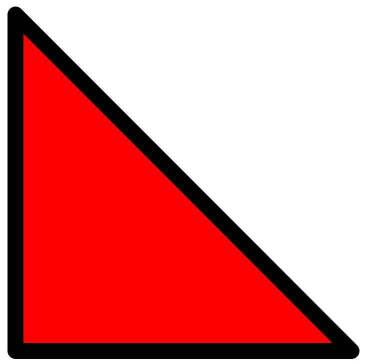
Assigning Color



```
glColor3f(0,0,1);

glBegin(GL_POLYGON);
  glVertex2f(-1,1);
  glVertex2f(-1,-1);
  glVertex2f(1,-1);
glEnd();
```

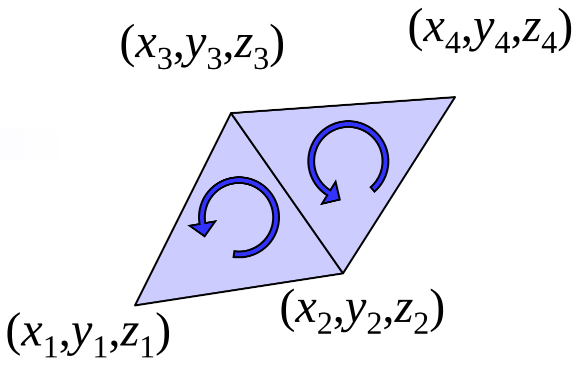


```
glBegin(GL_POLYGON);
  glColor3f(0,1,0);
  glVertex2f(-1,1);

  glColor3f(0,0,1);
  glVertex2f(-1,-1);

  glColor3f(1,0,0);
  glVertex2f(1,-1);
glEnd();
```



```
glColor3f(1,0,0);
glBegin(GL_POLYGON);
  glVertex2f(-1,1);
  glVertex2f(-1,-1);
  glVertex2f(1,-1);
glEnd();

glColor3f(0,0,0);
glBegin(GL_LINE_LOOP);
  glVertex2f(-1,1);
  glVertex2f(-1,-1);
  glVertex2f(1,-1);
glEnd();
```

Indexed Face Set



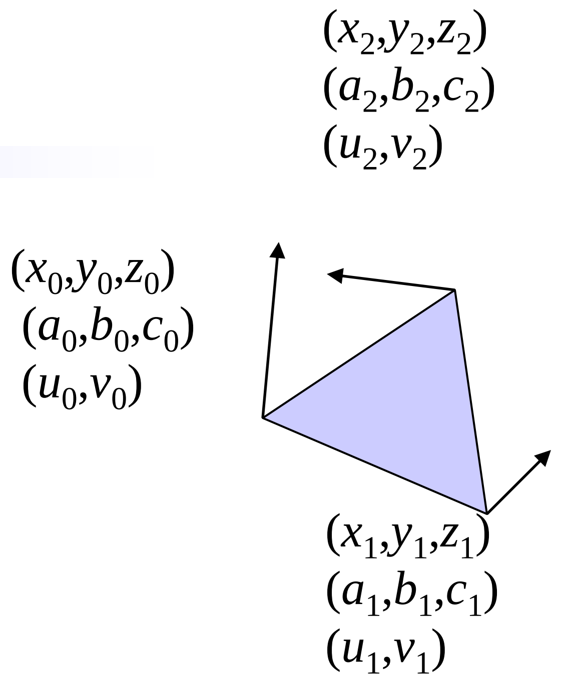
- Popular file format
  - VRML, Wavefront ".obj", etc.
- Ordered list of vertices
  - Prefaced by "v" (Wavefront)
  - Spatial coordinates x,y,z
  - Index given by order
- List of polygons
  - Prefaced by "f" (Wavefront)
  - Ordered list of vertex indices
  - Length = # of sides
  - Orientation given by order

v x_1 y_1 z_1
v x_2 y_2 z_2
v x_3 y_3 z_3
v x_4 y_4 z_4

```
f 1 2 3
f 2 4 3
```

Other Attributes

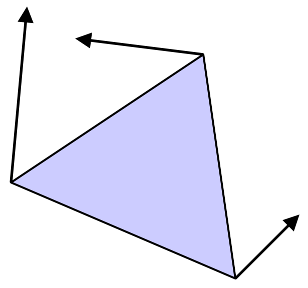
- Vertex normals
  - Prefixed w/ “vn” (Wavefront)
  - Contains x,y,z of normal
  - Not necessarily unit length
  - Not necessarily in vertex order
  - Indexed as with vertices
- Texture coordinates
  - Prefixed with “vt” (Wavefront)
  - Not necessarily in vertex order
  - Contains u,v surface parameters
- Faces
  - Uses “/” to separate indices
  - Vertex “/” normal “/” texture
  - Normal and texture optional
  - Can eliminate normal with “//”

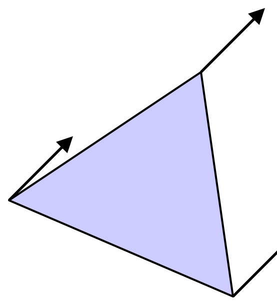


v x_0 y_0 z_0
v x_1 y_1 z_1
v x_2 y_2 z_2

vn a_0 b_0 c_0
vn a_1 b_1 c_1
vn a_2 b_2 c_2

vt u_0 v_0
vt u_1 v_1
vt u_2 v_2


f 0/0/0 1/1/1 2/2/2


f 0/0/0 1/0/1 2/0/2