Basic OpenGL

- What is OpenGL?
- What is GLUT?
- How does OpenGL work?
- How can I use GLUT?
- Which commands do I need to work with OpenGL?

OpenGL Installation – C/C++

Install OpenGL

OpenGL libraries and header files are

- opengl32.lib
- glu32.lib
- gl.h
- glu.h

Install GLUT

Download GLUT from:

http://www.xmission.com/~nate/glut/glut-3.7.6-bin.zip

GLUT Libraries and header files are

- glut32.lib
- glut.h

OpenGL Installation – C/C++

Use OpenGL & GLUT in your source code
Use the three header files (gl.h, glu.h and glut.h) in your source code.

Change project settings
In Visual C/C++ and .NET you'll have to do the following in order to link an application
using GLUT:
• Select Project/Settings.
• Go to linker settings.
• Add the following files to the Object/library modules: opengl32.lib, glut32.lib, glu32.lib.

OpenGL Installation – Java

•Download the JOGL Windows binaries ("jogl-win32.zip").
https://jogl.dev.java.net/files/documents/27/950/jogl-win32.zip

•unzip jogl-win32.zip.

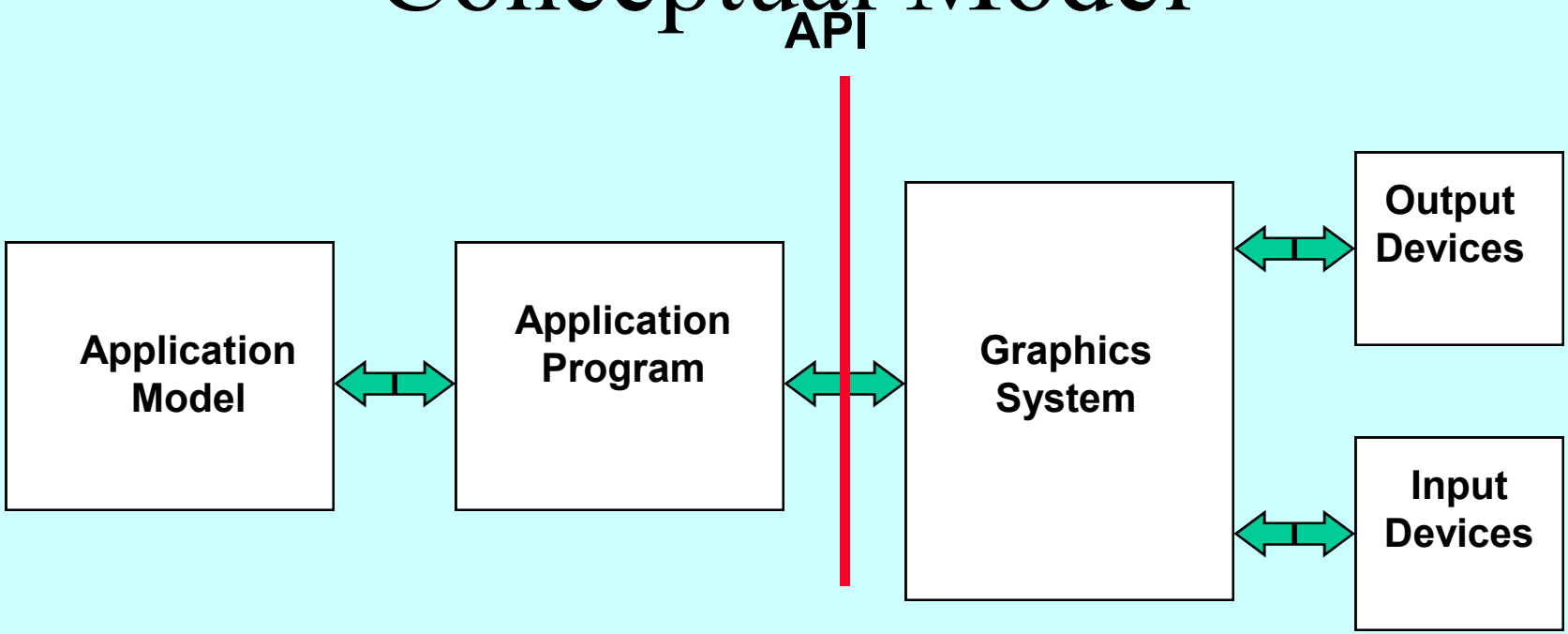
•Locate the Java SDK or JRE installation on your machine. These can typically be found at "C:\" or "C:\Program Files\Java\".
NOTE: You may have several SDKs/JREs installed on your machine so make sure that you locate the one that your compiler/IDE is using. To be safe you can always install it for each version on your system.

OpenGL Installation – Java

•Copy "jogl.dll" into the "\bin" directory for the Java SDK or JRE installed on your machine (i.e. C:\[jre_location]\bin\jogl.dll).

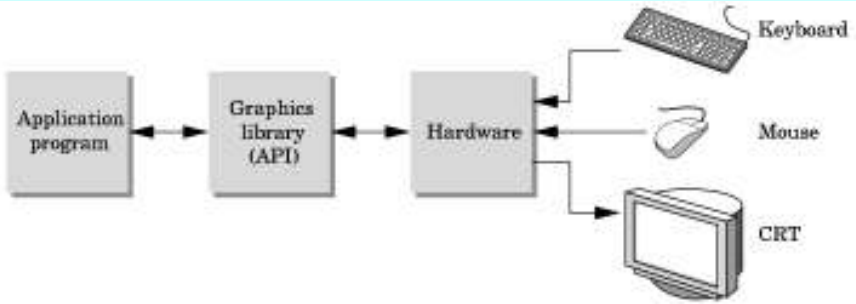
•Copy "jogl.jar" into the "\lib\ext" directory for the Java SDK or JRE installed on your machine (i.e. C:\[jre_location]\lib\ext\jogl.jar).

Computer Graphics Conceptual Model

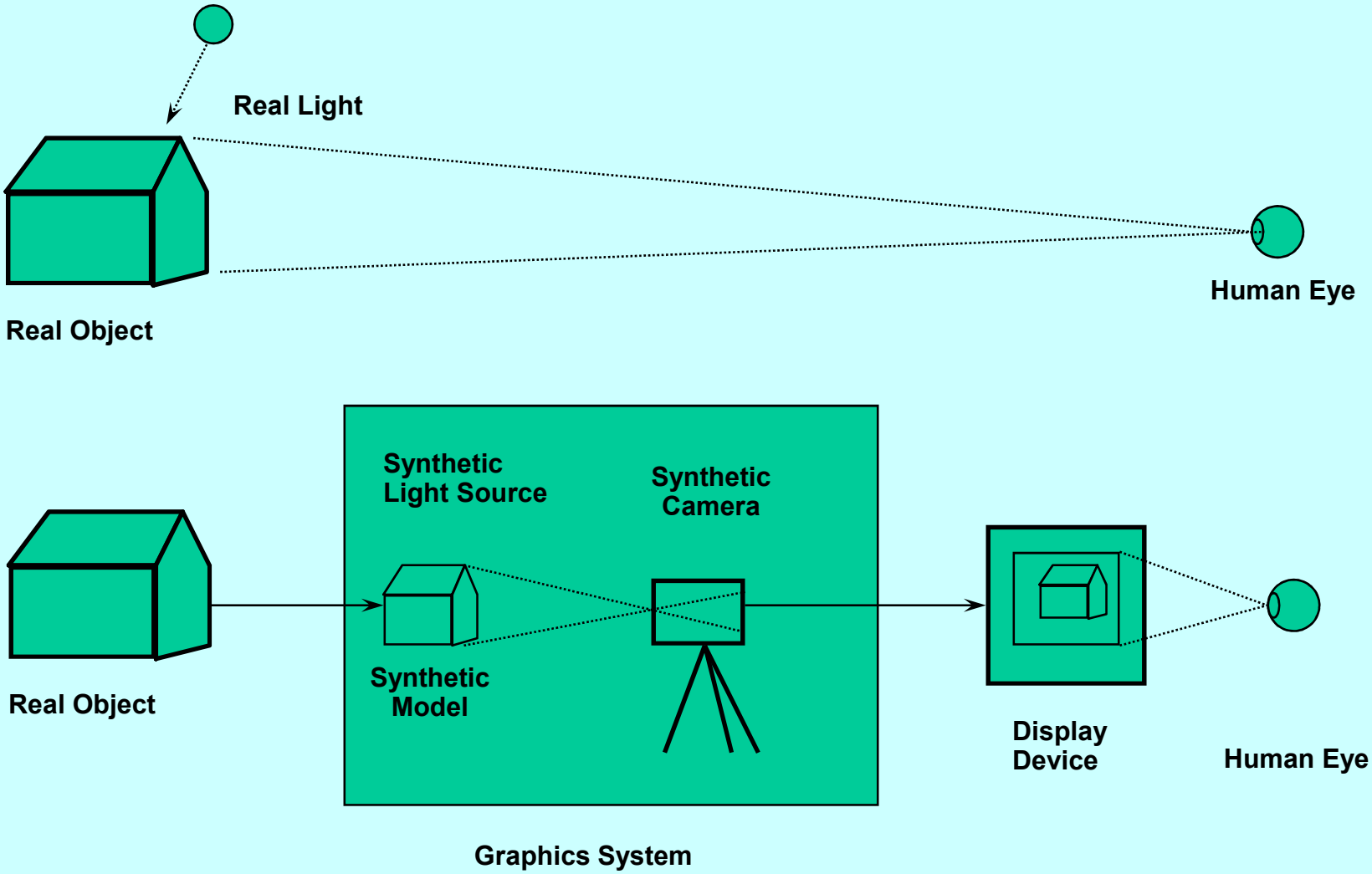


Function Calls or Protocol

Data



Graphics: Conceptual Model



What Is OpenGL?

- Graphics rendering API
  - high-quality color images composed of geometric and image primitives
  - window system independent
  - operating system independent
  - close to hardware

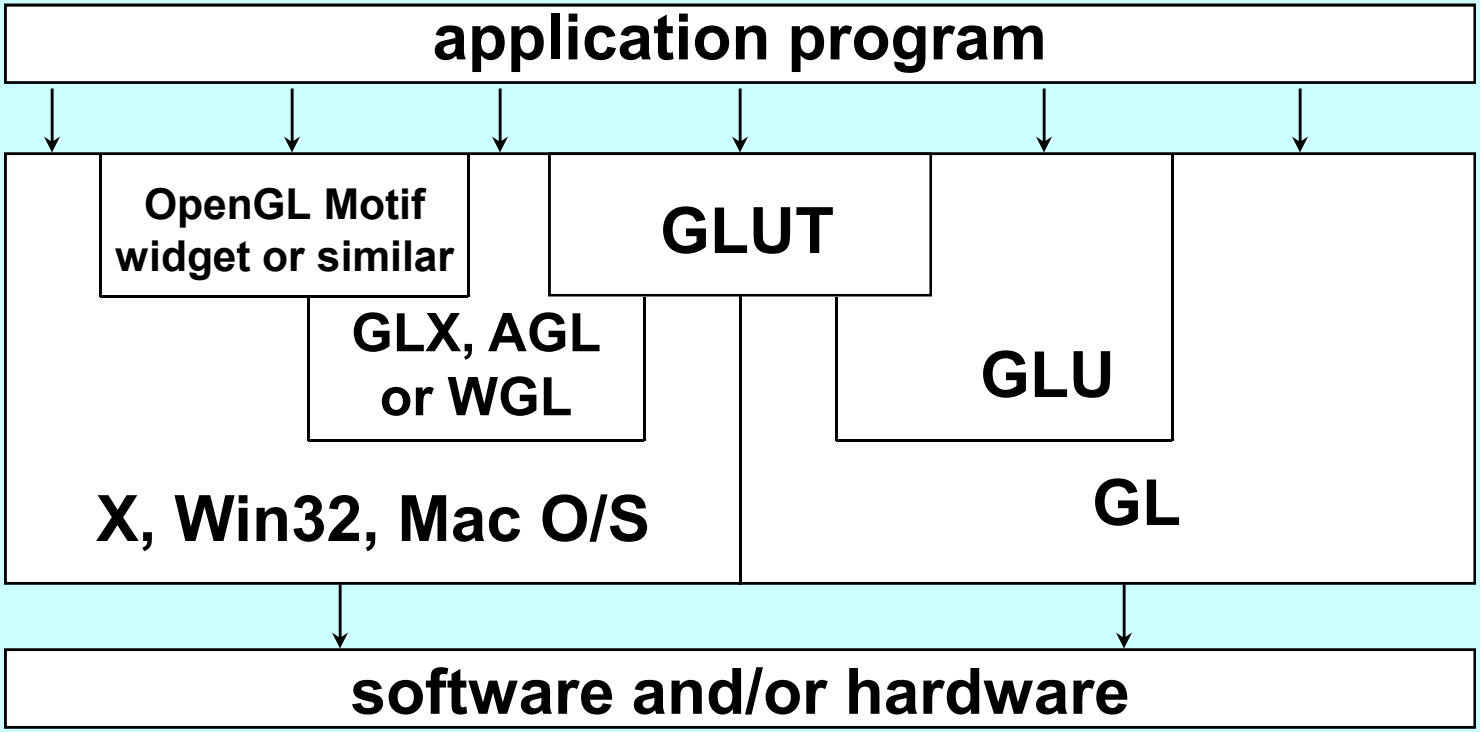
OpenGL and the Windowing System

- OpenGL is concerned only with rendering
  - Window system independent
  - No input functions
- OpenGL must interact with underlying OS and windowing system
  - Need minimal interface which may be system dependent

GLU and GLUT

- GLU (OpenGL Utility Library)
  - part of OpenGL
  - NURBS, quadric shapes, etc.
- GLUT (OpenGL Utility Toolkit)
  - portable windowing API
  - not officially part of OpenGL

OpenGL and Related APIs



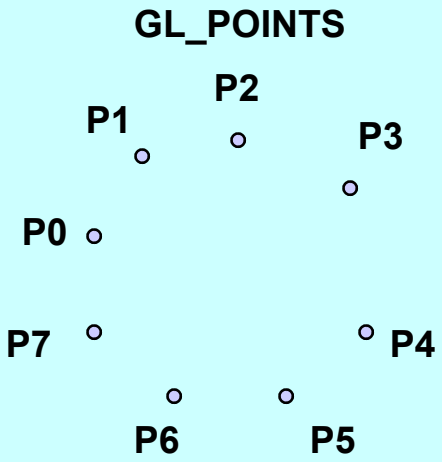
OpenGL as a Renderer

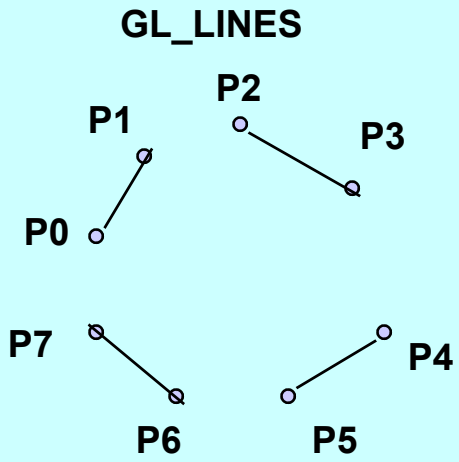
- Geometric primitives
  - points, lines and polygons
- Image Primitives
  - images and bitmaps
  - separate pipeline for images and geometry
    - linked through texture mapping
- Rendering depends on state
  - colors, materials, light sources, etc.

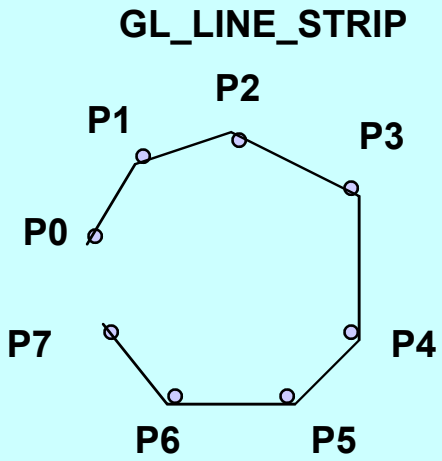
Components of a Graphics API

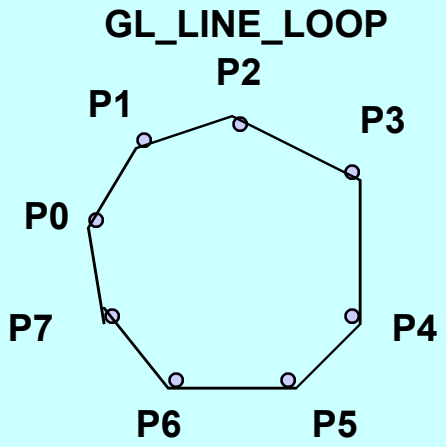
- Primitive functions
  - What to draw (points, polygons, pixels, text)
- Primitive attributes
  - How to draw it (color, pattern, style)
- Synthetic camera
  - Viewing functions (position, orientation, lens)
- Transformation functions
  - Rotate, scale, translate objects (models)
- Input functions
  - Handle interactivity (keyboards, mouse)
- Control functions
  - Communicate with window system
  - Initialization, error handling

Point and Line Segment Primitives

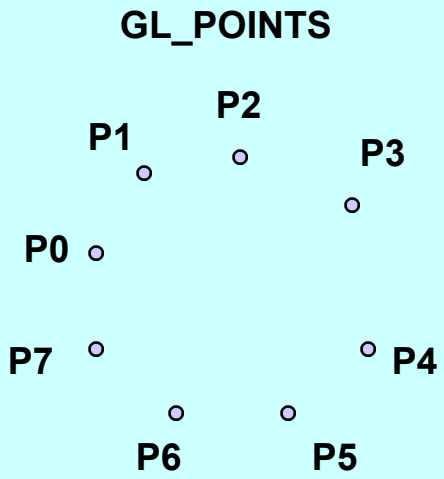


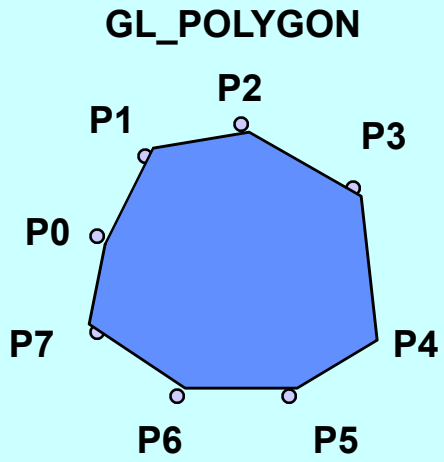


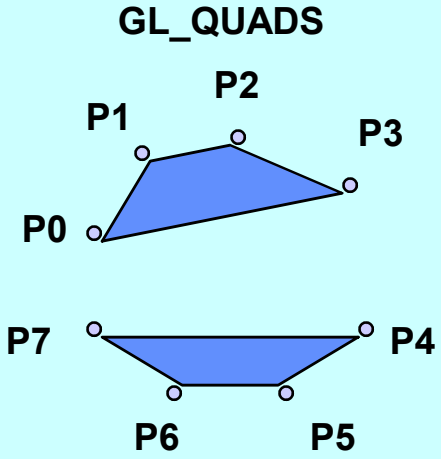


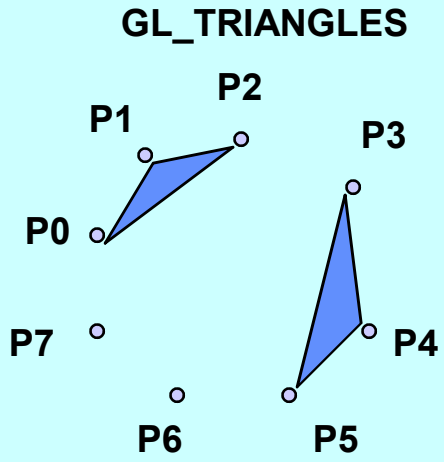


Polygon Primitives

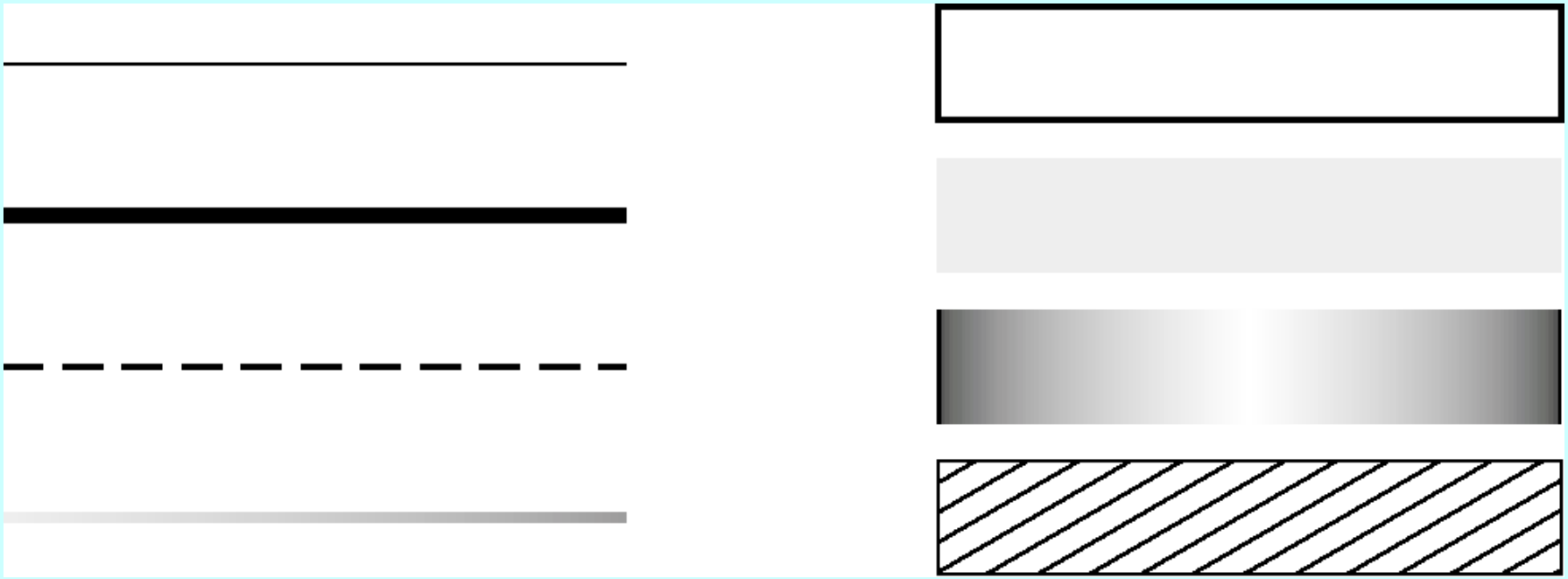




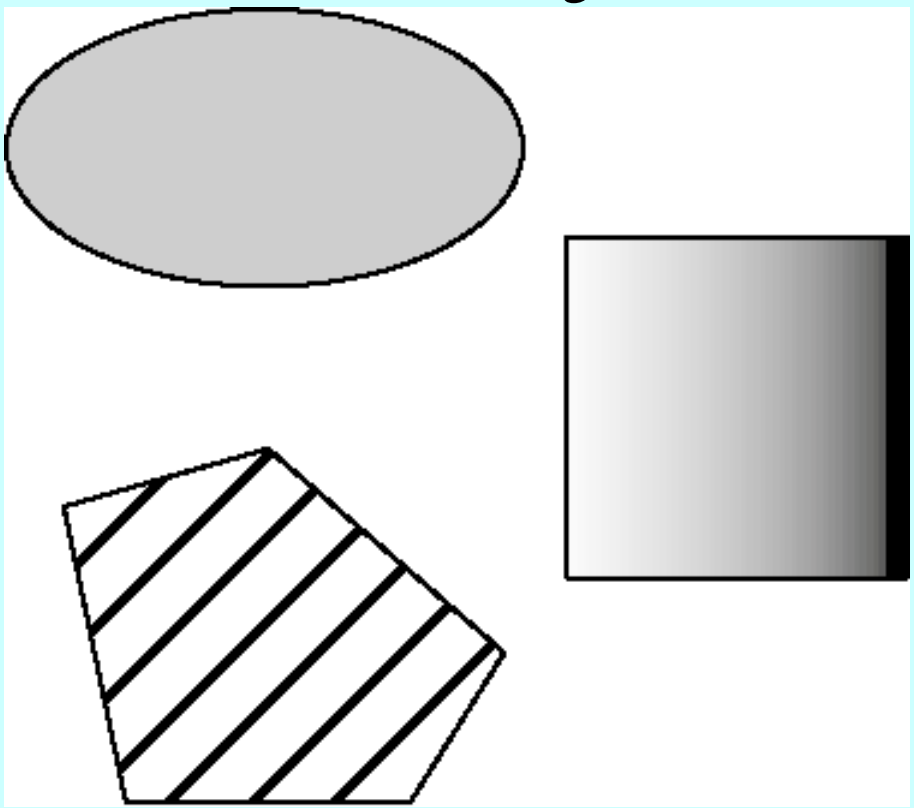




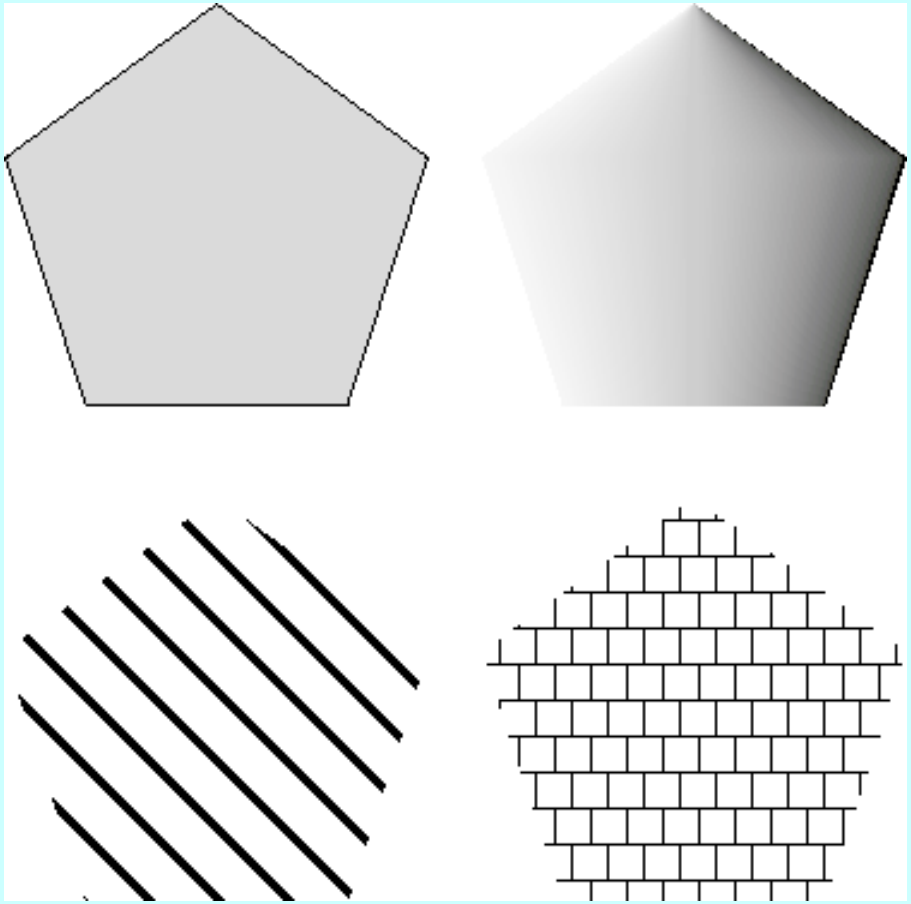
Attributes for lines and polygons



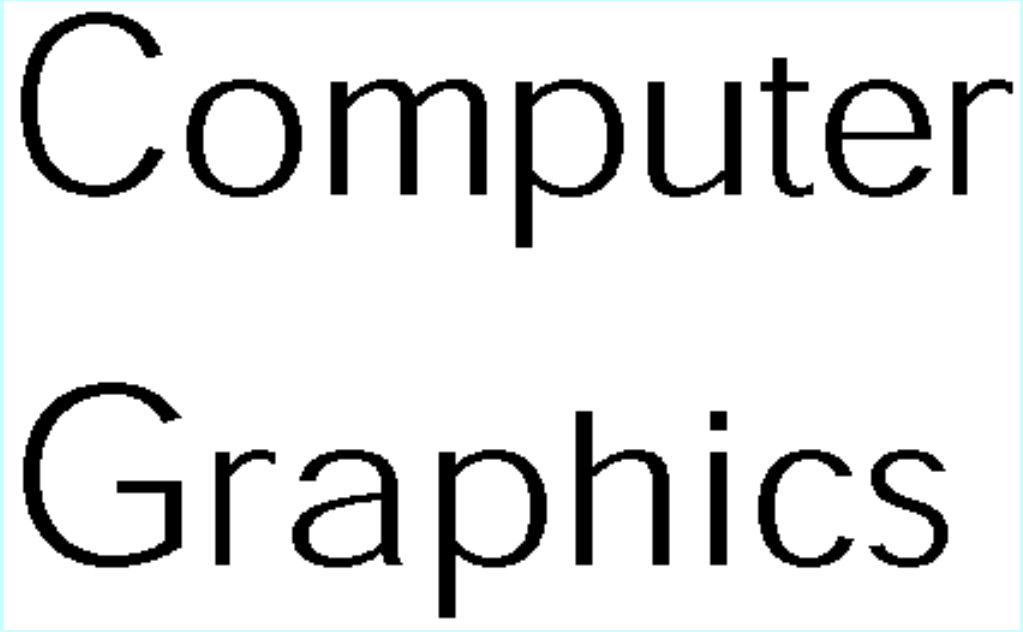
Filled Objects



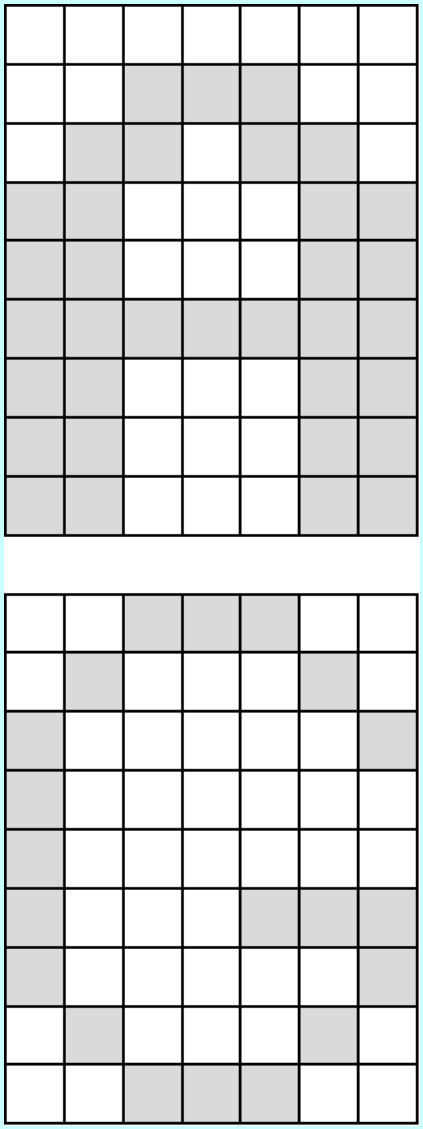
Different Attributes for Polygon



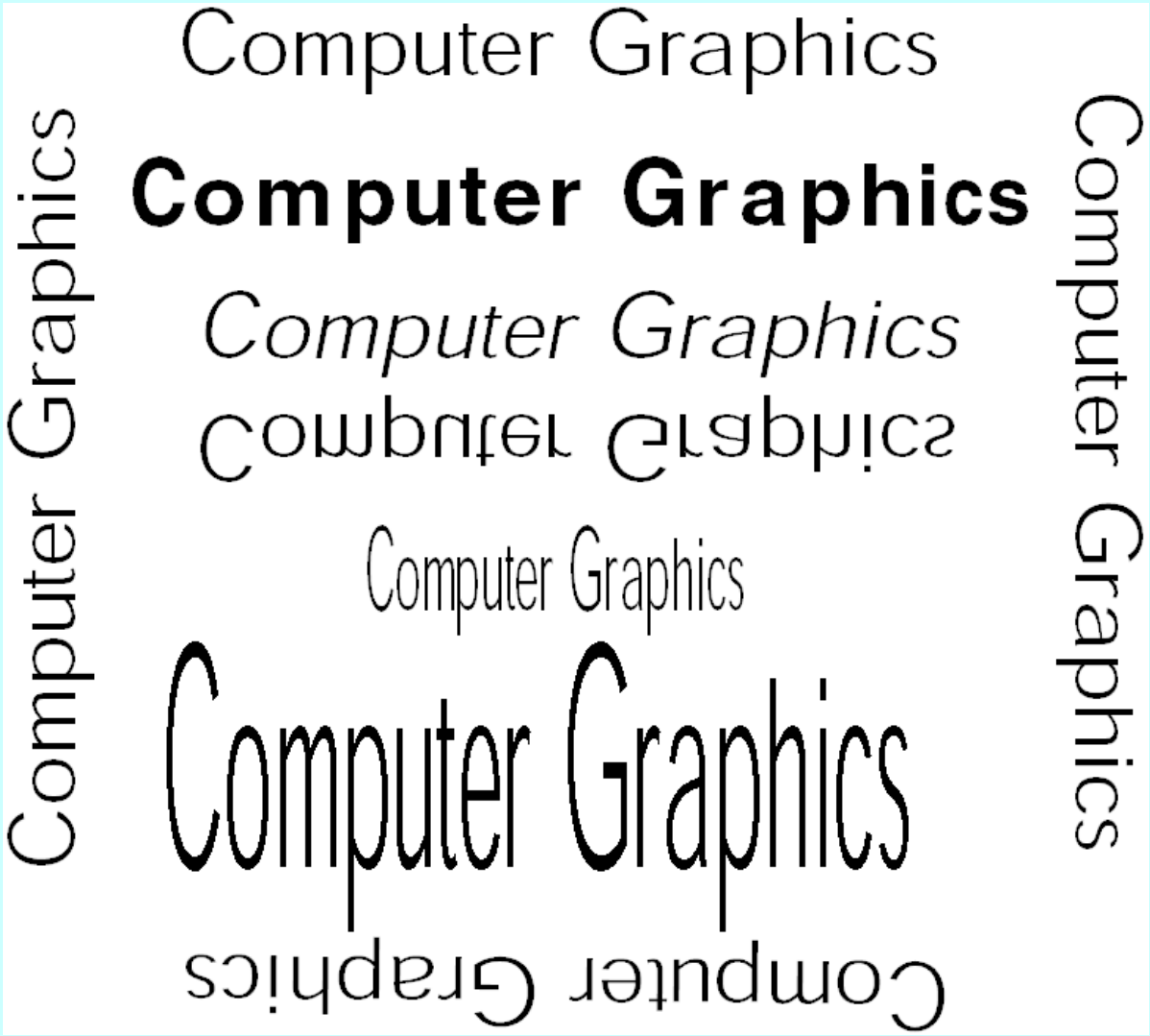
Example of Stroke text



Example of Raster text



Display of Stoke text



Display of Raster text

OpenGL Primitive Syntax

```
glBegin ( type );
    glVertex* (  .  .  .  );
        .
        .
        .
        .

    glVertex* (  .  .  .  );
glEnd ( );
```

OpenGL Primitive Syntax

```
glBegin(GL_POINTS);
    glVertex3f(100.0f, 100.0f, -25.0f);
glEnd( );
```

OpenGL Primitive Syntax

```
glBegin(GL_LINES);
    glVertex3f(100.0f, 100.0f, 0.0f); // origin of the
    glVertex3f(200.0f, 140.0f, 5.0f); // ending point o
glEnd( );
```

OpenGL Primitive Syntax

```
glBegin(GL_TRIANGLES);
    glVertex3f(100.0f, 100.0f, 0.0f);
    glVertex3f(150.0f, 100.0f, 0.0f);
    glVertex3f(125.0f, 50.0f, 0.0f);
glEnd( );
```

OpenGL Primitive Syntax

```
glBegin(GL_LINES);
    glVertex3f(100.0f, 100.0f, 0.0f); // origin of the
    glVertex3f(200.0f, 140.0f, 5.0f); // ending point o
    glVertex3f(120.0f, 170.0f, 10.0f); // origin of the
    glVertex3f(240.0f, 120.0f, 5.0f); // ending point o
glEnd( );
```

OpenGL Primitive Syntax

```
glBegin(GL_QUADS); // 4 vertices forming a quad
      glColor3f(1.0f, 0.0f, 0.0f); // Red
      glVertex2f(-0.5f, -0.5f);    // x, y
      glVertex2f( 0.5f, -0.5f);
      glVertex2f( 0.5f,  0.5f);
      glVertex2f(-0.5f,  0.5f);
glEnd();
```