

2. Basic OpenGL Programming

Global state

OpenGL is *not object oriented!*

It has a *global state* organized as a tree-like stack via 'push' and 'pop' *separators*.

```
image = GlobState(attributes; objects)
```

- *Primitives*: (what, shape node) points, line segments, polygons, pixels, curves
- *Attributes system state*: (how, property) accum buffer, color, current, depth buffer, enable, eval, fog, hint, lighting, line, list, pixel mode, point, polygon, polygon stipple, scissor, stencil buffer, texture, transform, viewport.
- **2 matrix stacks**:
 glMatrixMode(*GL_PROJECTION*) Viewing: projection, clipping
 glMatrixMode(*GL_MODELVIEW*) Transformation: rotate, scale, translate
- *events* (input, error handling, network connection, etc.)

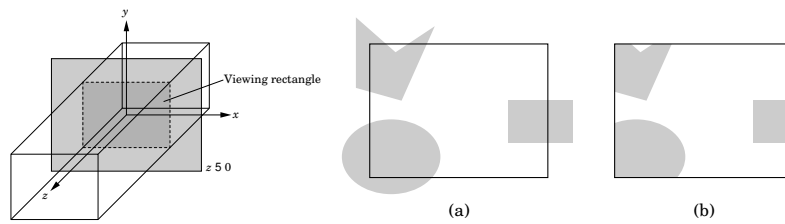
Example:

```
glClear()
glColor3f()
glBegin (GL_POINTS)
    glVertex2fv()
glEnd (GL_POINTS)
glFlush()
```

```
OPENGL/1bufftst
```

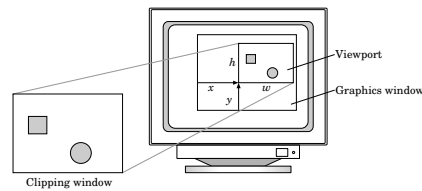
Clipping and viewing volume

- *glOrtho*(left, right, bottom, top, near, far)
- camera always looks down *negative z-axis*.
- camera 'views' *everything* in the viewing volume (frustum) – also objects 'behind' the camera.
- *glOrtho2D*(left, right, bottom, top) = *glOrtho*(left, right, bottom, top, -1, 1)

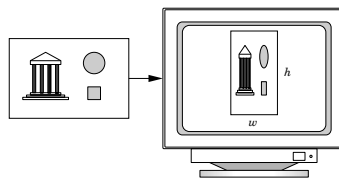


GLUT organizes interaction with X windows, input devices (Xlib,Xtk) see *glut.h*

- *glutInitWindowSize()* How large is the window? Note: image may be smaller than window



- *glutInitWindowPosition()*
Where on the screen does the image appear? (0,0) = upper left
2D (Physical) device coordinates = raster coordinates = screen coordinates
vs. 3D world coordinates
distortion if width, height ratio of *gluOrtho* \neq *glutInitWindowSize*



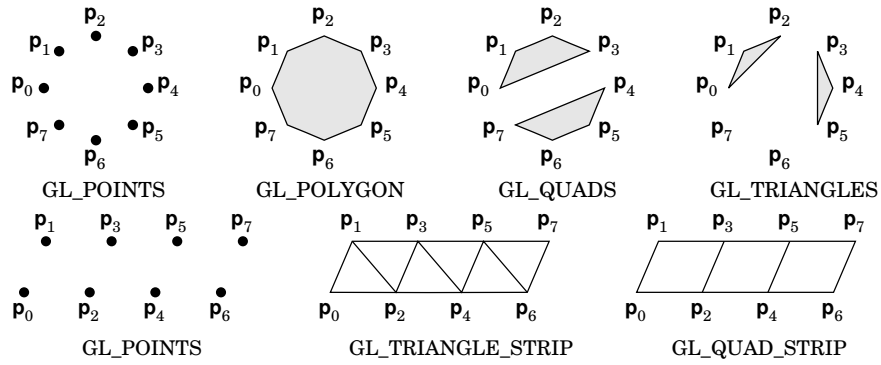
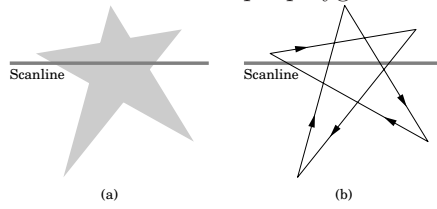
avoid distortion: *glViewport*(ll_x, ll_y, width, height) in pixels
part of *state* – hence interactive change possible

- OpenGL (scene), Postscript: origin lower left, *x* right, *y* up
- X: upper left, *x* right, *y* down
- ascii: upper left, *x* down, *y* right (landscape)

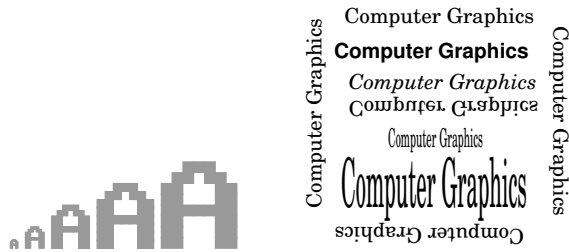
Primitives *glBegin*(GL_SOMETHING) ... *glEnd*();
GL_POINTS, GL_LINES (segments 1-2, 3-4) vs GL_LINE_STRIP (poly-line)
Polygon: simple, convex, flat (triangles will do!)

simple polygon: no holes, no self-intersections.

Trouble with non-simple polygons:

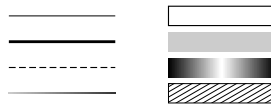


Stroke Text, Raster Text

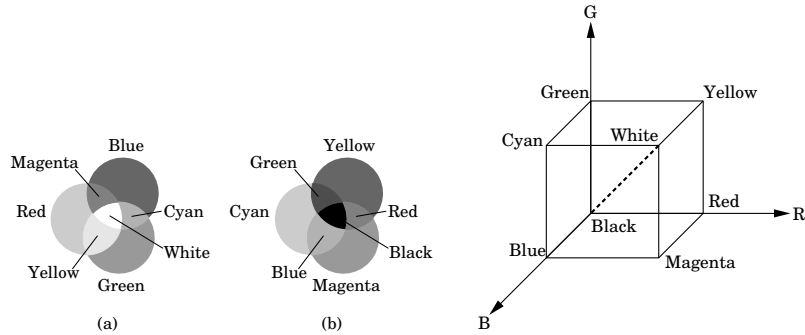


glPointSize (pixels) has screen raster dependence.
glRasterPos determines the placement.

Attributes are *bound to* primitives. The binding may change.
 The present attribute values are part of the *state of the graphics system*.



Color – additive, subtractive colors



$glColor(r,g,b)$, $glClearColor(r,g,b,\alpha)$: 2^{24} colors
 1280×1024 pixels with 24 bits (RGB) + $\alpha > 4 \cdot 2^{20}$ bytes = 4MB.
 Color gamut
 Hue, Saturation, Intensity model

lookup table (old)

Event Processing

- *Immediate mode*: primitive is rendered as soon as defined.
Present system state determines appearance.
- disappears “too fast”
- $glutMainLoop()$:
event-processing loop displays current until interrupted
- $glutDisplay(*func)$ *display callback* called: initially, when moving windows, etc.