CS312

OpenGL basics

What is openGL?

- A low-level graphics library specification.
- A small set of geometric primitives:
  - Points
  - Lines
  - Polygons
  - Images
  - Bitmaps

Points, Lines, Polygons: Geometric primitives

Images, Bitmaps: Image primitives

OpenGL Libraries

- OpenGL core library
  - OpenGL32 on Windows
  - GL/Mesa on most unix/linux systems
- OpenGL Utility Library (GLU)
  - Provides functionality in OpenGL core but avoids having to rewrite code

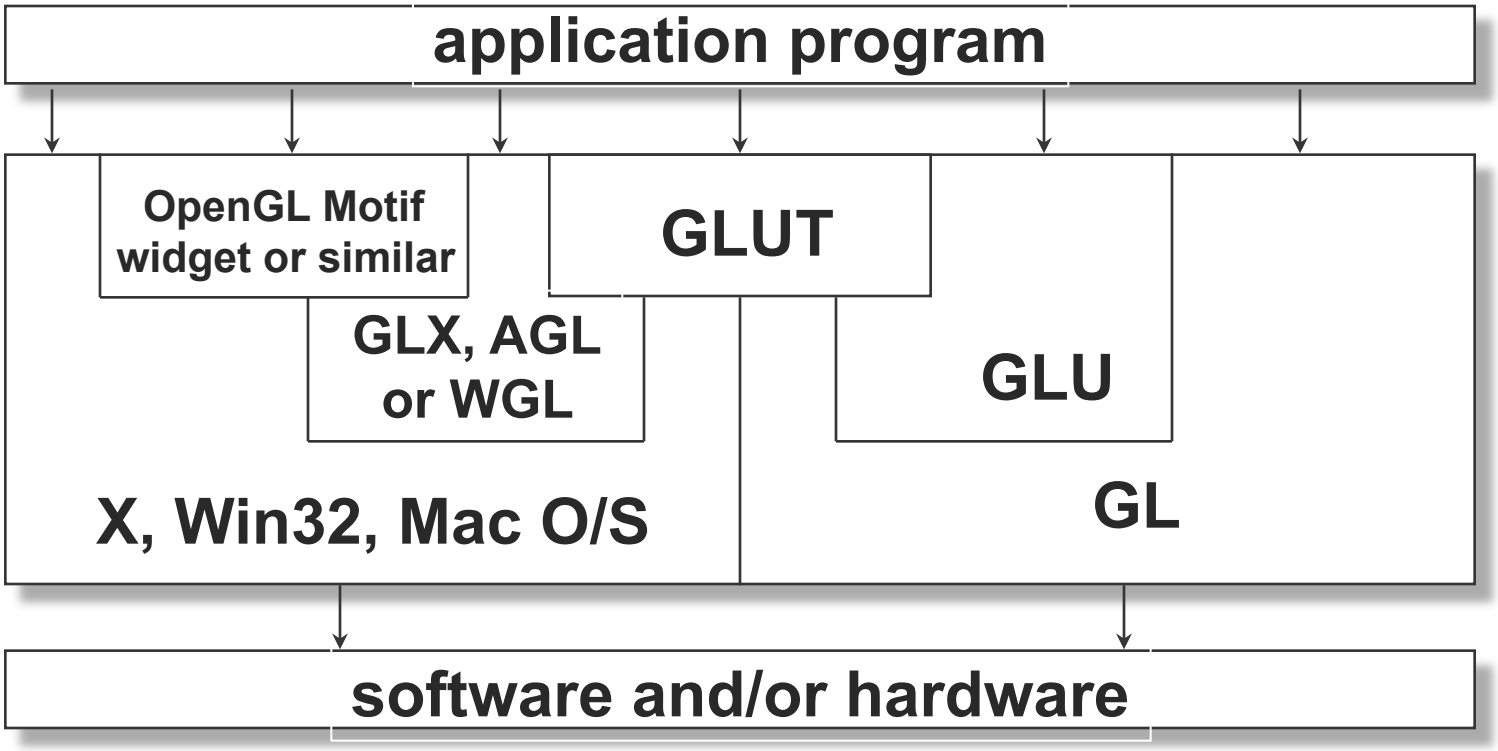
GL is window system independent

- Extra libraries are needed to connect GL to the OS
  - GLX – X windows, Unix
  - AGL – Apple Macintosh
  - WGL – Microsoft Windows

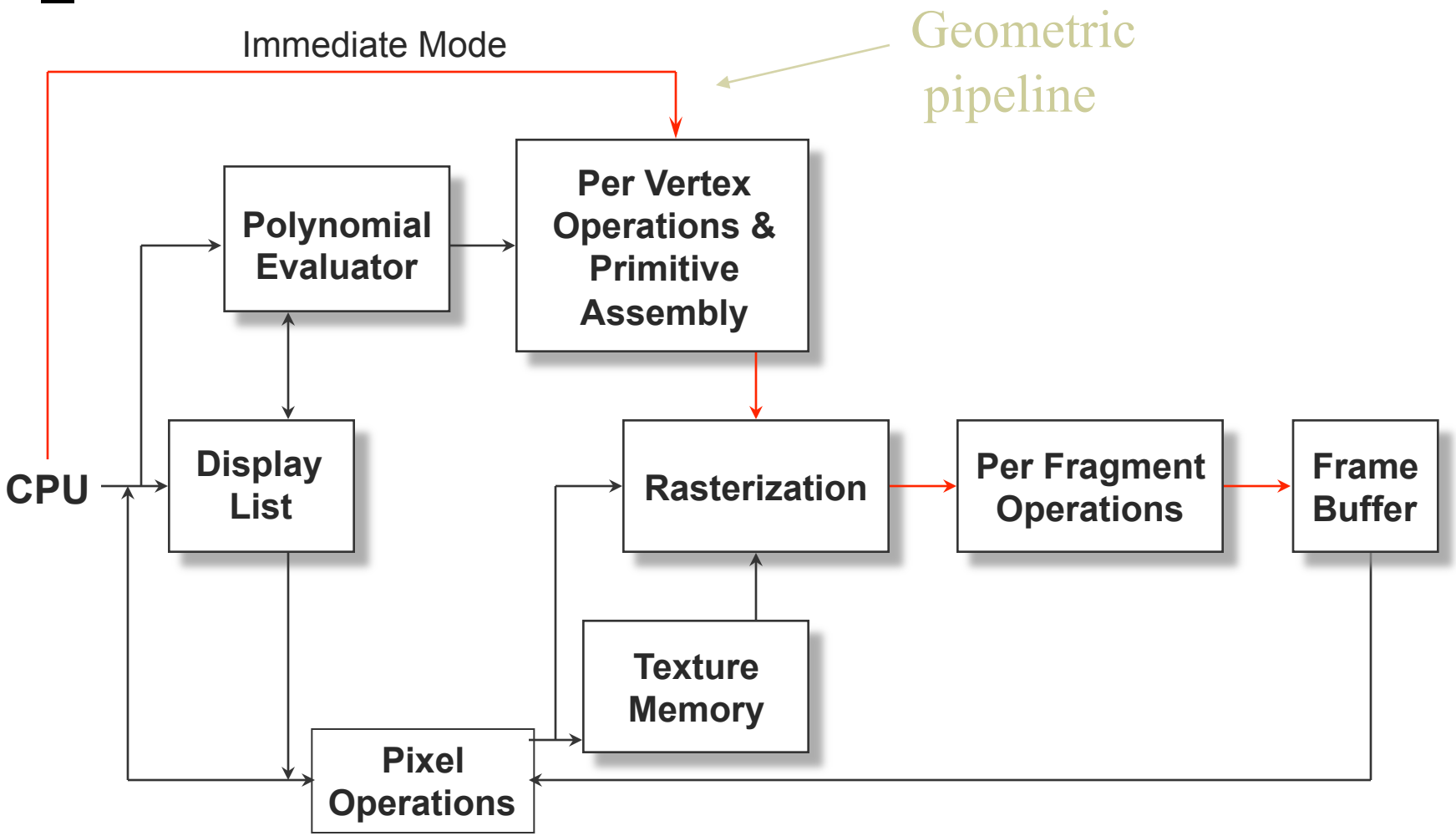
GLUT

- OpenGL Utility Toolkit (GLUT)
  - Provides functionality common to all window systems
    - Open a window
    - Get input from mouse and keyboard
    - Menus
    - Event-driven
  - Code is portable but GLUT is minimal

Software Organization



OpenGL Architecture



OpenGL State

- OpenGL is a state machine
- OpenGL functions are of two types
  - Primitive generating
    - Can cause output if primitive is visible
    - How vertices are processed and appearance of primitive are controlled by the state
  - State changing
    - Transformation functions
    - Attribute functions

Typical GL Program Structure

- Configure and open a window
- Initialize GL state
- Register callback functions
  - Render
  - Resize
  - Events
- Enter infinite event processing loop

Render/Display

- Draw simple geometric primitives
- Change states (how GL draws these primitives)
  - How they are lit or colored
  - How they are mapped from the user's two- or three-dimensional model space to the two-dimensional screen.
  - There are also calls to effect direct control of the frame buffer, such as reading and writing pixels.

Header files

- *#include <GL/gl.h>*
- *#include <GL/glu.h>*
- *#include <GL/glut.h>*

Enumerated Types

C pref	openGL type	C type	Data type
b	GLbyte	signed char	8-bit int
s	GLshort	short	16-bit int
i	GLint, GLsizei	int or long	32-bit int
f	GLfloat, GLclampf	float	32-bit float
d	GLdouble, GLclampd	double	64-bit float
ub	GLubyte, GLboolean	unsigned char	8-bit unsigned int
us	GLushort	unsigned short	16-bit unsigned int
ui	GLuint, GLenum, GLbitfield	unsigned int or unsigned long	32-bit unsigned int

OpenGL Function Naming Conventions

glVertex3fv(v)

Library prefix

- gl
- glu
- glut

Root command

Number of components

Data type

- b – byte
- ub – unsigned byte
- s – short
- us – unsigned short
- i – int
- f – float
- d - double

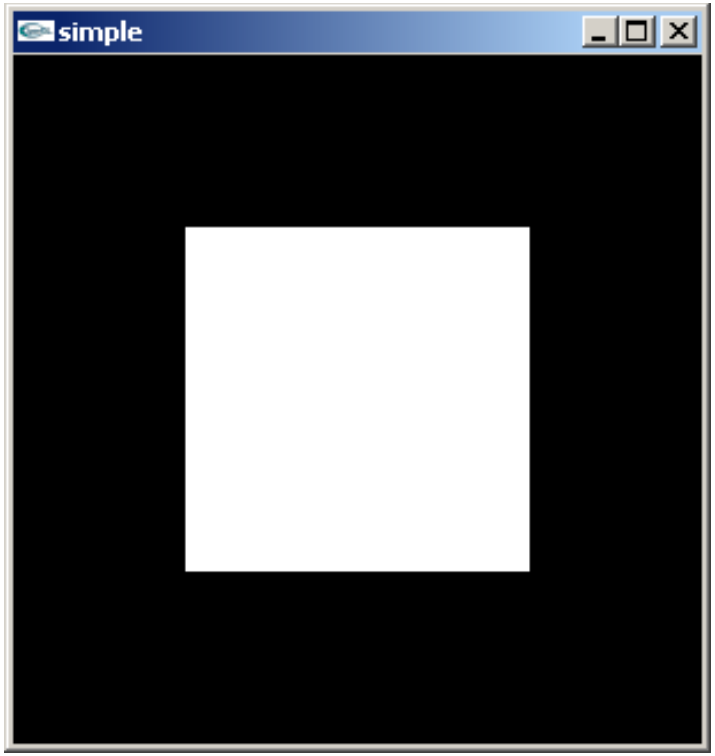
Vector

glVertex*

- Capitalizes first letter of each word
- glVertex{234}{sifd}[v](TYPE coords);
  - *glVertex2i(1, 2);*
  - *glVertex3f(1.5, -2.0, M_PI);*
  - *double v[3] = {0.0, 1.5, 3.6};*
    glVertex3dv(v);
- Must appear btw *glBegin* and *glEnd*

A Simple Program

Generate a square on a solid background



Simple program

```
int main(int argc, char** argv){
  glutCreateWindow("simple");
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
}
```

display()

```
void display(){
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_POLYGON);
    glVertex2f(-0.5, -0.5);
    glVertex2f(-0.5, 0.5);
    glVertex2f(0.5, 0.5);
    glVertex2f(0.5, -0.5);
  glEnd();
  glFlush();
}
```

Event Loop

- The program defines a *display callback* function named **`display`**
  - Every glut program must have a display callback
  - The display callback is executed whenever OpenGL decides the display must be refreshed, for example when the window is opened
  - The **`main`** function ends with the program entering an event loop

Defaults

- `simple.c` is too simple
- Makes heavy use of state variable default values for
  - Viewing
  - Colors
  - Window parameters
- Next version will make the defaults more explicit

Notes on compilation

- Unix/linux
  - Include files usually in **`/usr/include/`**
  - Compile with **`-lglut -lGLU -lGL`** loader flags
  - May have to add **`-l`** flag for X libraries
  - Mesa implementation included with most linux distributions

simple.c revisited

- In this version, we will see the same output but we have defined relevant state values through function calls with the default values
- In particular, we set
  - Colors
  - Window properties

main.c

```
int main(int argc, char** argv) {
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
 glutInitWindowSize(500,500);
 glutInitWindowPosition(0,0);
 glutCreateWindow(argv[0]);
 glutDisplayFunc(display);

 init();

 glutMainLoop();
}
```

define window properties

display callback

set OpenGL state

enter event loop

GLUT functions

- **`glutInit`** allows application to get command line arguments and initializes system
- **`gluInitDisplayMode`** requests properties for the window (the *rendering context*)
  - RGB color
  - Single buffering
  - Properties logically ORed together
- **`glutWindowSize`** in pixels
- **`glutWindowPosition`** from top-left corner of display
- **`glutCreateWindow`** create window with title
- **`glutDisplayFunc`** display callback
- **`glutMainLoop`** enter infinite event loop

init.c

black clear color

```
void init() {
  glClearColor (0.0, 0.0, 0.0, 0.0);

  glColor3f(1.0, 1.0, 1.0);
}
```

draw with white

Specifiying geometric primitives

- Each geometric object is described by:
  - A set of vertices
  - Type of the primitive

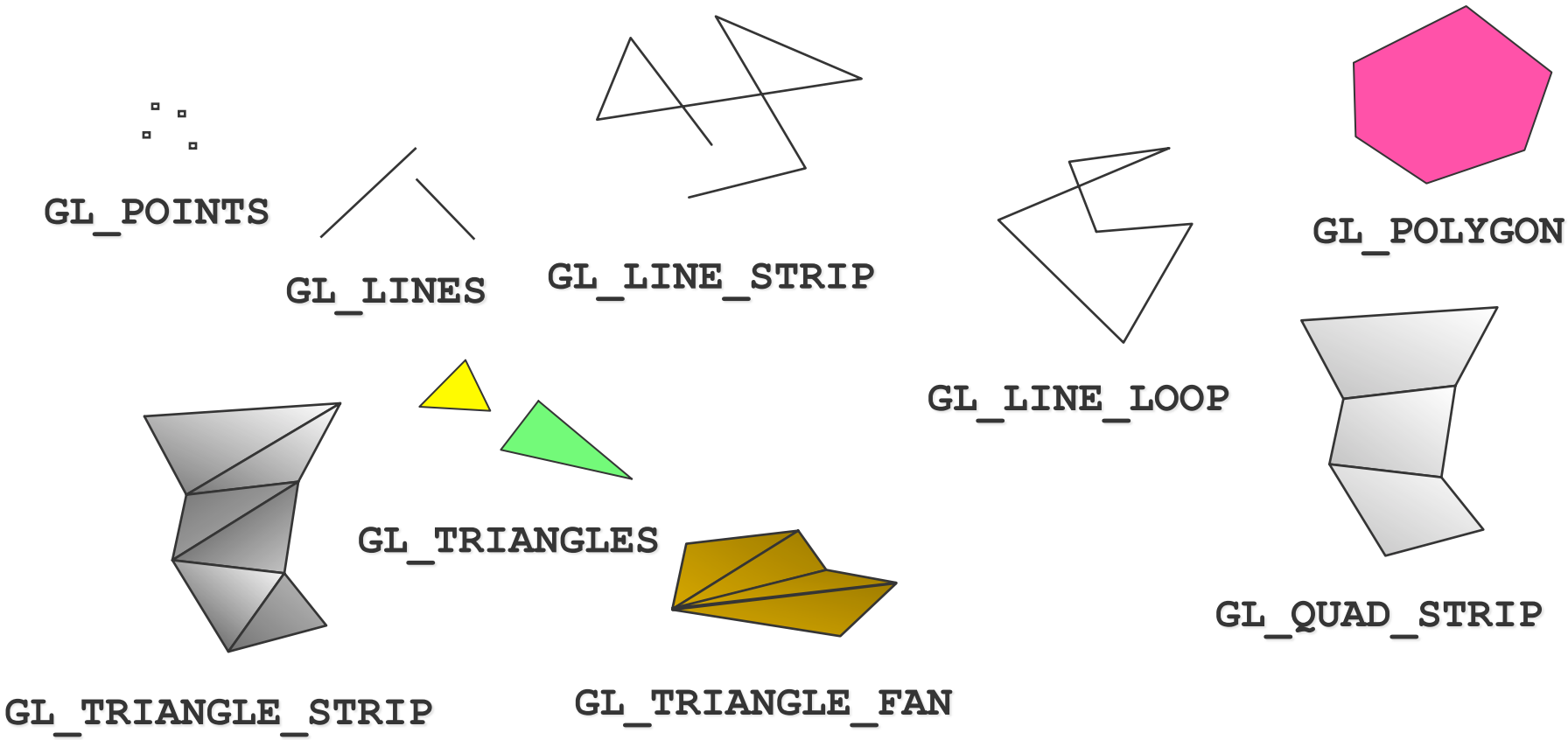
Specifying geometric primitives

glBegin( )

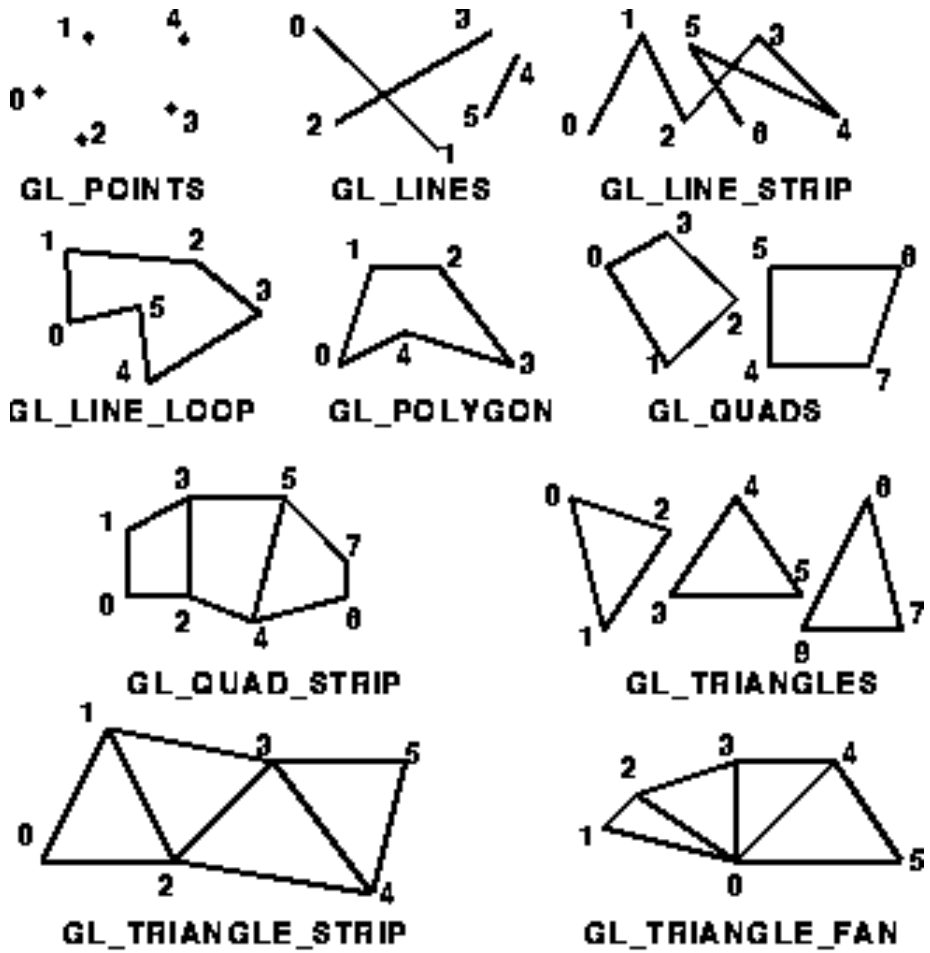
GL_POINTS	GL_LINE_STRIP
GL_LINES	GL_LINE_LOOP
GL_TRIANGLES	GL_TRIANGLE_STRIP
GL_QUADS	GL_TRIANGLE_FAN
GL_POLYGON	GL_QUAD_STRIP

glEnd()

OpenGL Primitives

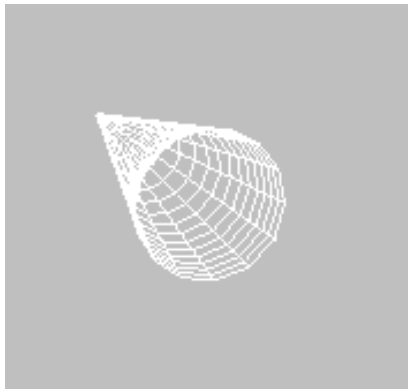


GL geometric primitives

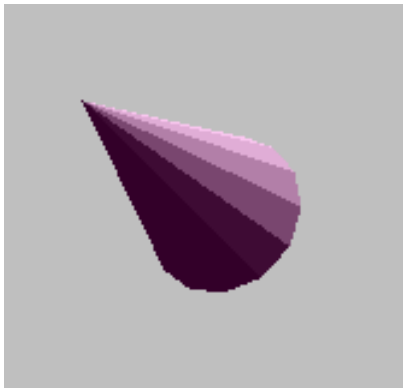


Geometric primitives: examples

```
glBegin(GL_LINES);
      [lots of
      glVertex
      calls];
glEnd();
```

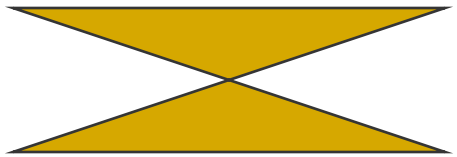


```
glBegin(GL_QUADS);
      [lots of
      glVertex
      calls];
glEnd();
```



Polygon Issues

- OpenGL only correctly displays polygons that are
  - Simple: edges cannot cross
  - Convex: All points on line segment between two points in a polygon are also in the polygon
  - Flat: all vertices are in the same plane
- Triangles satisfy all conditions



nonsimple polygon

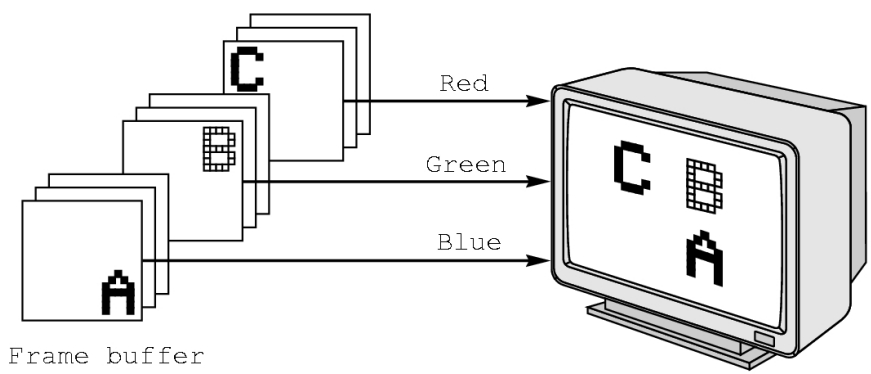
nonconvex polygon

Attributes

- Attributes are part of the OpenGL state and determine the appearance of objects
  - Color (points, lines, polygons)
  - Size and width (points, lines)
  - Stipple pattern (lines, polygons)
  - Polygon mode
    - Display as filled: solid color or stipple pattern
    - Display edges

RGB color

- Each color component is stored separately in the frame buffer
- Usually 8 bits per component in buffer
- In **`glColor3f`** the color values range from 0.0 (none) to 1.0 (all)



RGB: glColor*

glColor3f(0.0, 0.0, 0.0) – black

glColor3f(1.0, 0.0, 0.0) – red

glColor3f(0.0, 1.0, 0.0) – green

glColor3f(0.0, 0.0, 1.0) – blue

glColor3f(1.0, 1.0, 0.0) – yellow

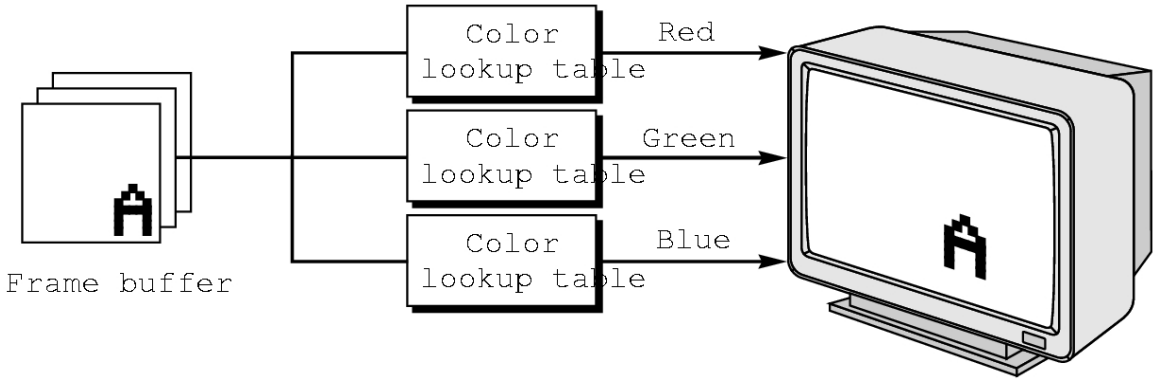
glColor3f(1.0, 0.0, 1.0) – magenta

glColor3f(0.0, 1.0, 1.0) – cyan

glColor3f(1.0, 1.0, 1.0) – white

Indexed Color

- Colors are indices into tables of RGB values
- Requires less memory
  - indices usually 8 bits



GL color models

- *glutInitDisplayMode()*
  - GLUT_RGBA == GLUT_RGB
  - GLUT_INDEX
- *glClearColor(1.0, 1.0, 1.0, 0.0);*

Alpha value –
controls transparency
Set to 0.0 for now

Color and State

- The color as set by **`glColor`** becomes part of the state and will be used until changed
  - Colors and other attributes are not part of the object but are assigned when the object is rendered
- We can create conceptual *vertex colors* by code such as

```
    glColor
glVertex
glColor
glVertex
```

Smooth Color

- Default is *smooth* shading
  - OpenGL interpolates vertex colors across visible polygons
- Alternative is *flat shading*
  - Color of first vertex

  determines fill color
- **`glShadeModel`**

  `(GL_SMOOTH)`

  or **`GL_FLAT`**