

Computer Graphics

Bing-Yu Chen
National Taiwan University

Introduction to OpenGL

- General OpenGL Introduction
- An Example OpenGL Program
- Drawing with OpenGL
- Transformations
- Animation and Depth Buffering
- Lighting
- Evaluation and NURBS
- Texture Mapping
- Advanced OpenGL Topics
- Imaging

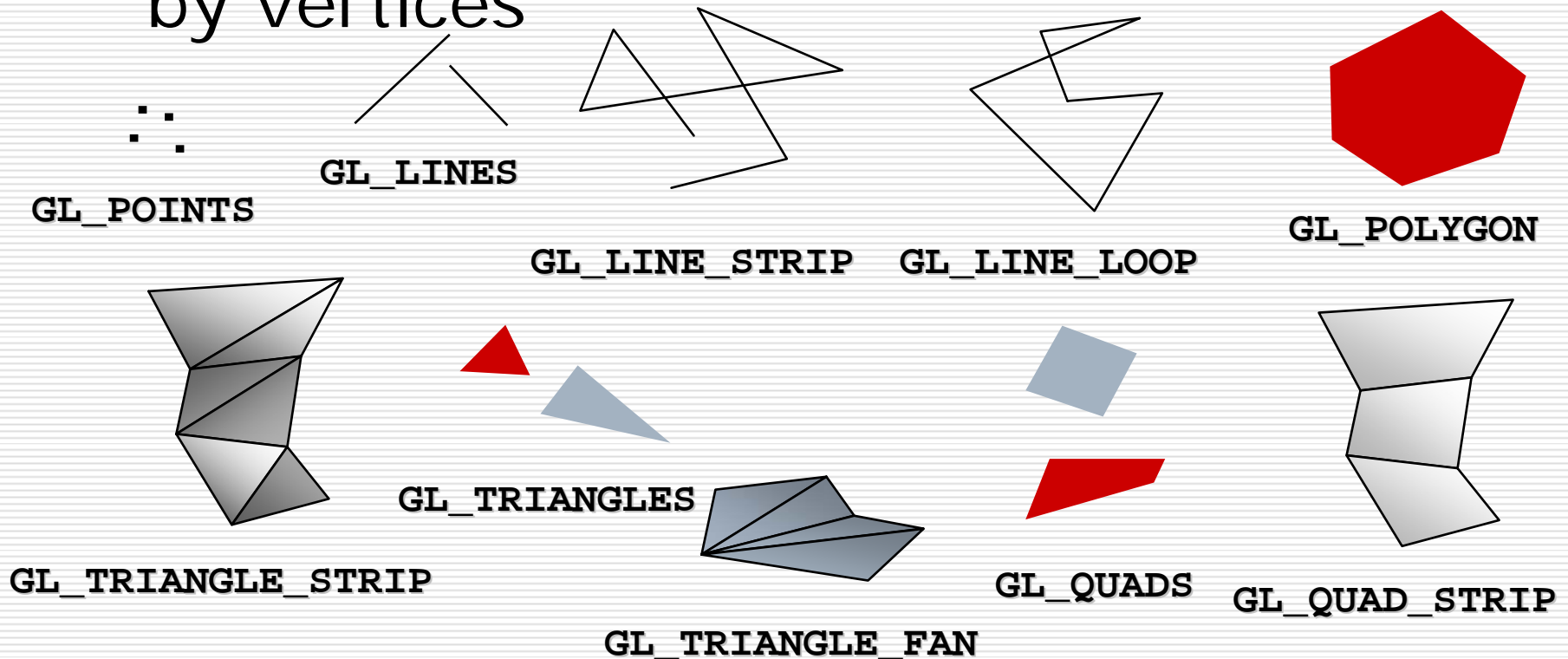
modified from
Dave Shreiner, Ed Angel, and Vicki Shreiner.
An Interactive Introduction to OpenGL Programming.
ACM SIGGRAPH 2001 Conference Course Notes #54.
& *ACM SIGGRAPH 2004 Conference Course Notes #29.*

What can OpenGL Draw?

- 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.
-

OpenGL Geometric Primitives

- All geometric primitives are specified by vertices



Simple Example

```
void drawRhombus( GLfloat color[] )
{
    glBegin( GL_QUADS );
    glColor3fv( color );
    glVertex2f( 0.0, 0.0 );
    glVertex2f( 1.0, 0.0 );
    glVertex2f( 1.5, 1.118 );
    glVertex2f( 0.5, 1.118 );
    glEnd();
}
```

Specifying Geometric Primitives

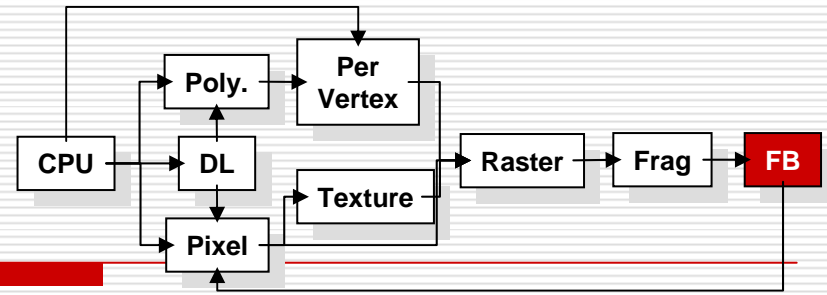
- Primitives are specified using

```
glBegin( primType );  
glEnd();
```

- *primType* determines how vertices are combined

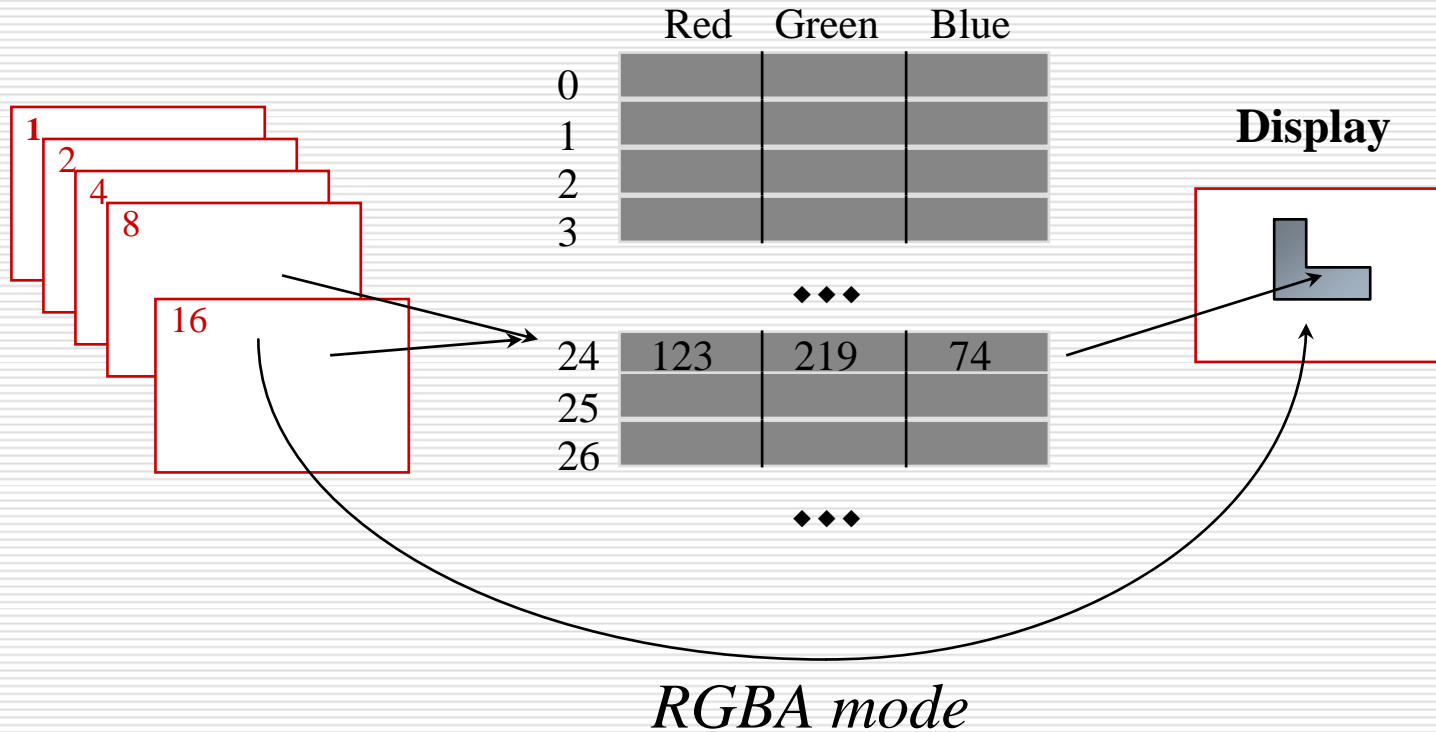
```
GLfloat red, green, blue;  
GLfloat coords[3];  
glBegin( primType );  
for ( i = 0; i < nVerts; ++i ) {  
    glColor3f( red, green, blue );  
    glVertex3fv( coords );  
}  
glEnd();
```

OpenGL Color Models

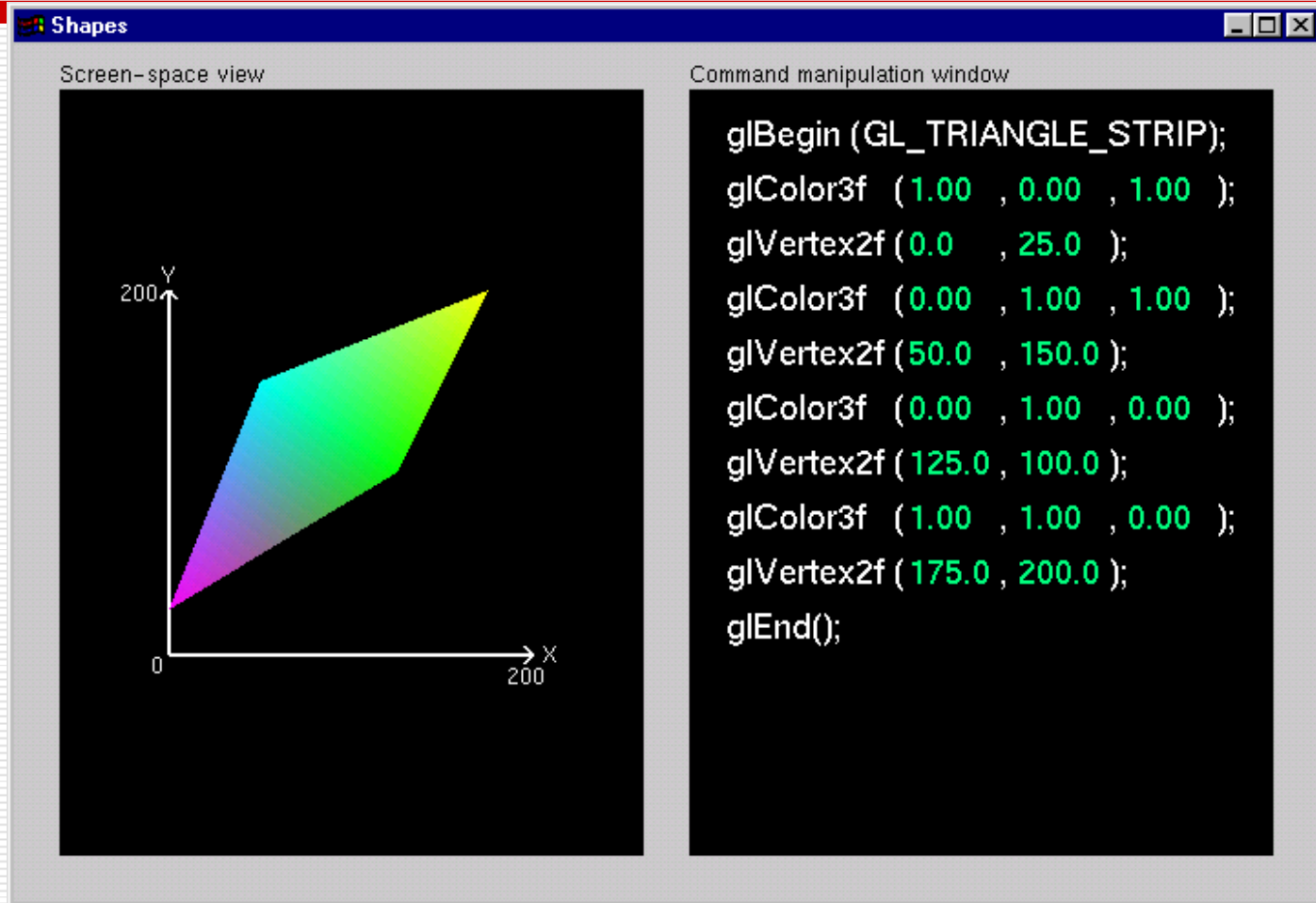


□ RGBA or Color Index

color index mode



Shapes Tutorial

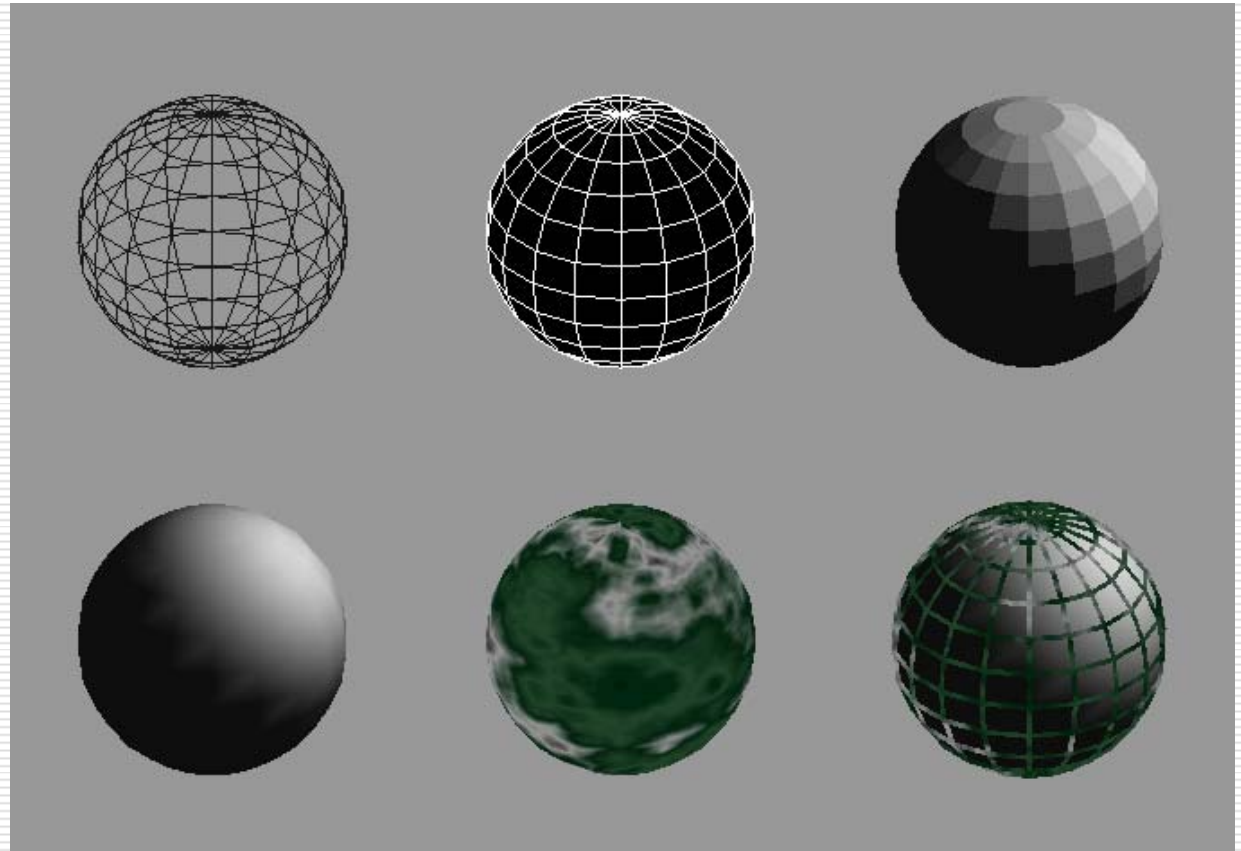


The screenshot shows a window titled "Shapes" with two panes. The left pane, labeled "Screen-space view", displays a 2D coordinate system with X and Y axes ranging from 0 to 200. A quadrilateral is plotted, starting at the origin (0,0) and extending to approximately (175, 200). The shape is filled with a color gradient from purple at the origin to yellow at the top-right corner. The right pane, labeled "Command manipulation window", contains the following OpenGL code:

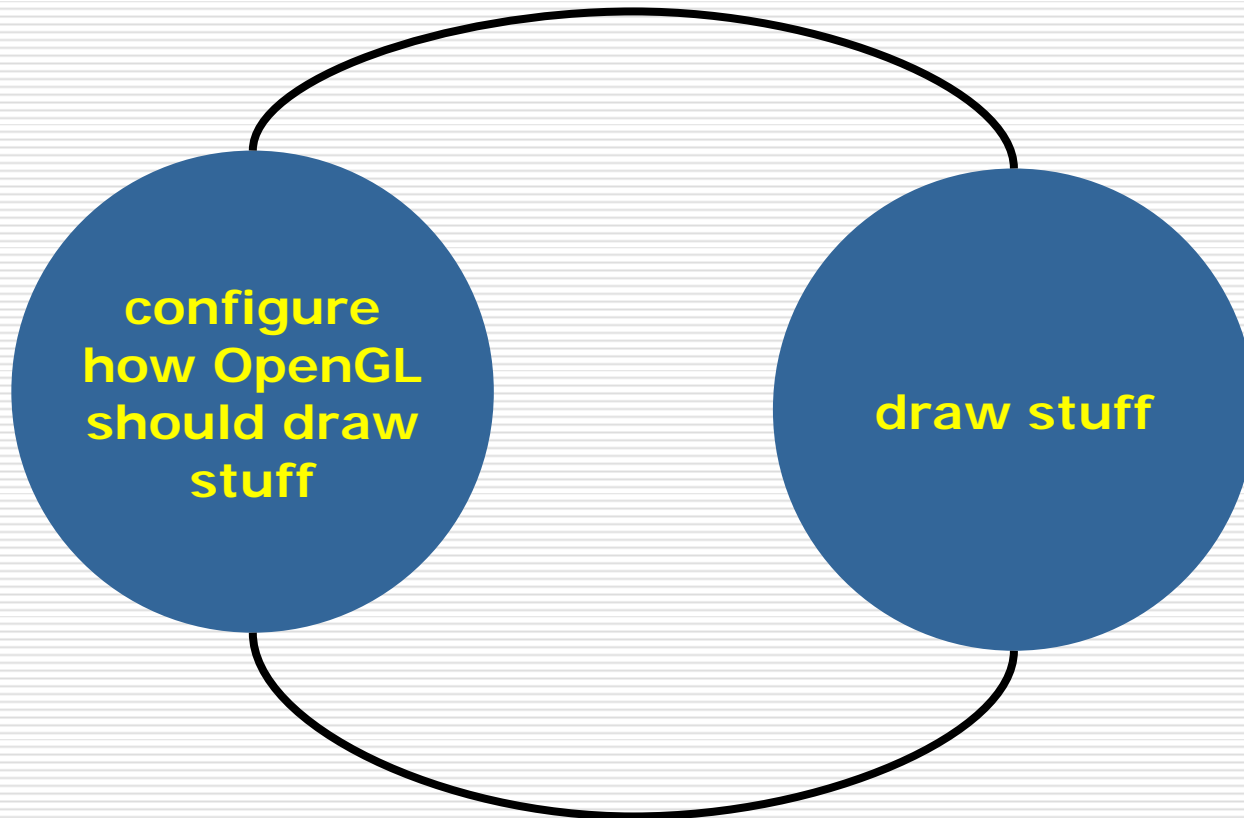
```
glBegin (GL_TRIANGLE_STRIP);  
glColor3f (1.00 , 0.00 , 1.00 ) ;  
glVertex2f (0.0 , 25.0 ) ;  
glColor3f (0.00 , 1.00 , 1.00 ) ;  
glVertex2f (50.0 , 150.0 ) ;  
glColor3f (0.00 , 1.00 , 0.00 ) ;  
glVertex2f (125.0 , 100.0 ) ;  
glColor3f (1.00 , 1.00 , 0.00 ) ;  
glVertex2f (175.0 , 200.0 ) ;  
glEnd();
```


Controlling Rendering Appearance

From
Wireframe
to
Texture
Mapped



How OpenGL Works: The Conceptual Model



OpenGL's State Machine

- All rendering attributes are encapsulated in the OpenGL State
 - rendering styles
 - shading
 - lighting
 - texture mapping
-

Manipulating OpenGL State

- Appearance is controlled by current state

```
for each ( primitive to render ) {  
    update OpenGL state  
    render primitive  
}
```

- Manipulating vertex attributes is most common way to manipulate state

`glColor*() / glIndex*()`

`glNormal*()`

`glTexCoord*()`

Controlling current state

□ Setting State

```
glPointSize( size );
```

```
glLineStipple( repeat, pattern );
```

```
glShadeModel( GL_SMOOTH );
```

□ Enabling Features

```
glEnable( GL_LIGHTING );
```

```
glDisable( GL_TEXTURE_2D );
```
