MIDTERM EXAM, 2015

Midterm Exam

- Exam day
  - April 21st, 18:00 (6pm)
  - Location: 충무관 102호

HOMEWORK 1

Homework 1

- Posted on lms.sejong.ac.kr
- Due on April 10th, 23:59

HOMEWORK 2

Homework 2

- Posted on lms.sejong.ac.kr
- Due on April 17th, 23:59

OPENGL PIPELINE

Yun Jang
jangy@sejong.edu

Disclaimer

- These slides can only be used as study material for the Computer Graphics at Sejong University
- The slides cannot be distributed or used for another purpose

OpenGL is a State Machine!

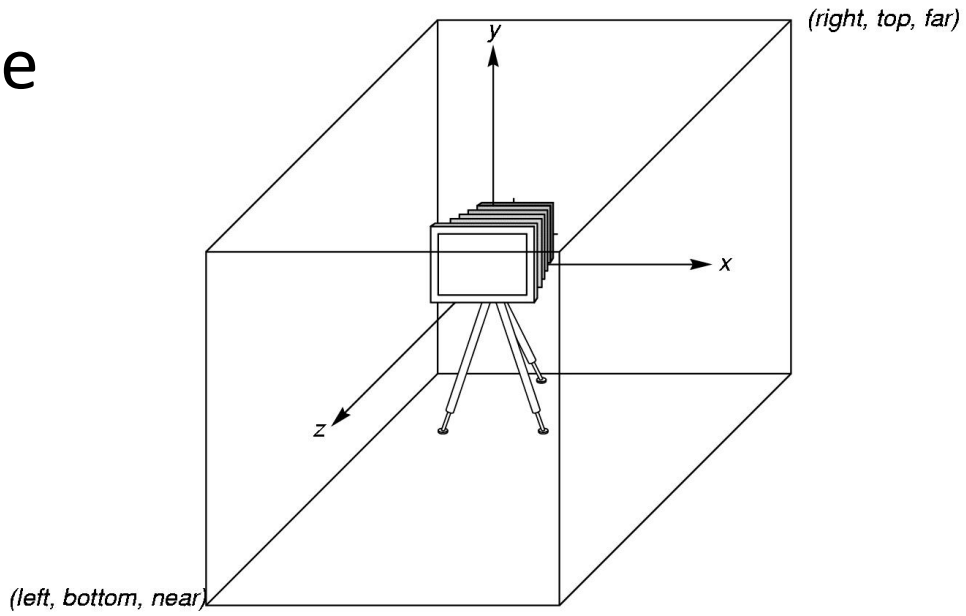
- Stores internal state
  - Color, transformation, line width, point size…
- Two types of operations
  - Change state
  - Draw primitives
- **Metaphor**: artist's kit (palette, canvas, paintbrushes)
  - Color
  - Brush size
  - Position on canvas

Coordinate Systems

- The units in **`glVertex`** are determined by the application and are called *object* or *problem coordinates*
- The viewing specifications are also in object coordinates and it is the size of the viewing volume that determines what will appear in the image
- Internally, OpenGL will convert to *camera (eye) coordinates* and later to *screen coordinates*
- OpenGL also uses some internal representations that usually are not visible to the application

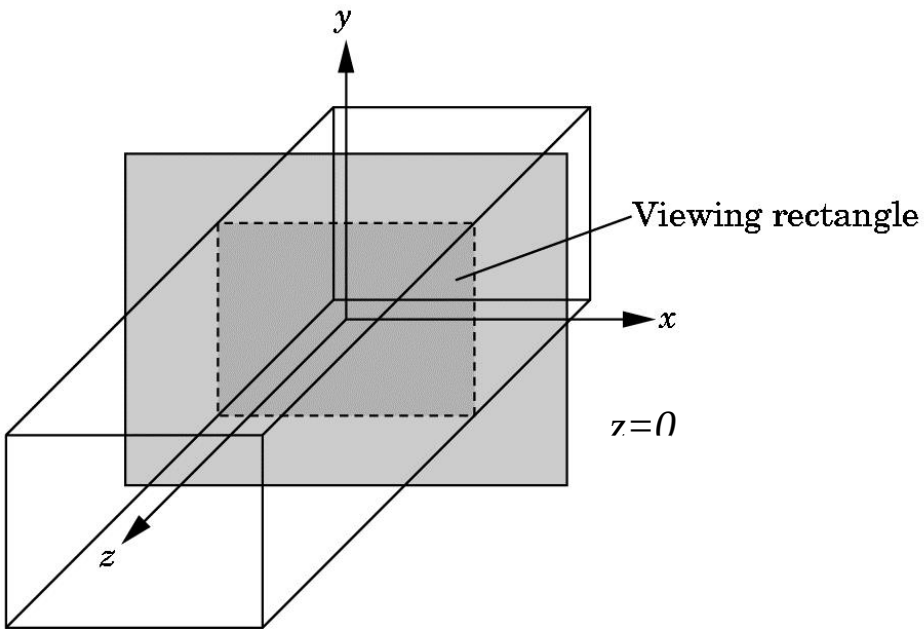
OpenGL Camera

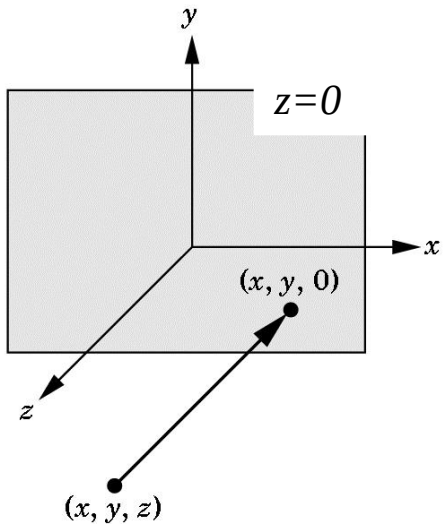
- OpenGL places a camera at the origin in object space pointing in the negative z direction
- The default viewing volume is a box centered at the origin with a side of length 2



Orthographic Viewing

In the default orthographic view, points are projected forward along the z axis onto the plane $z=0$





Transformations and Viewing

- In OpenGL, projection is carried out by a projection matrix (transformation)
- There is only one set of transformation functions so we must set the matrix mode first

```
glMatrixMode(GL_PROJECTION)
```

- Transformation functions are incremental so we start with an identity matrix and alter it with a projection matrix that gives the view volume

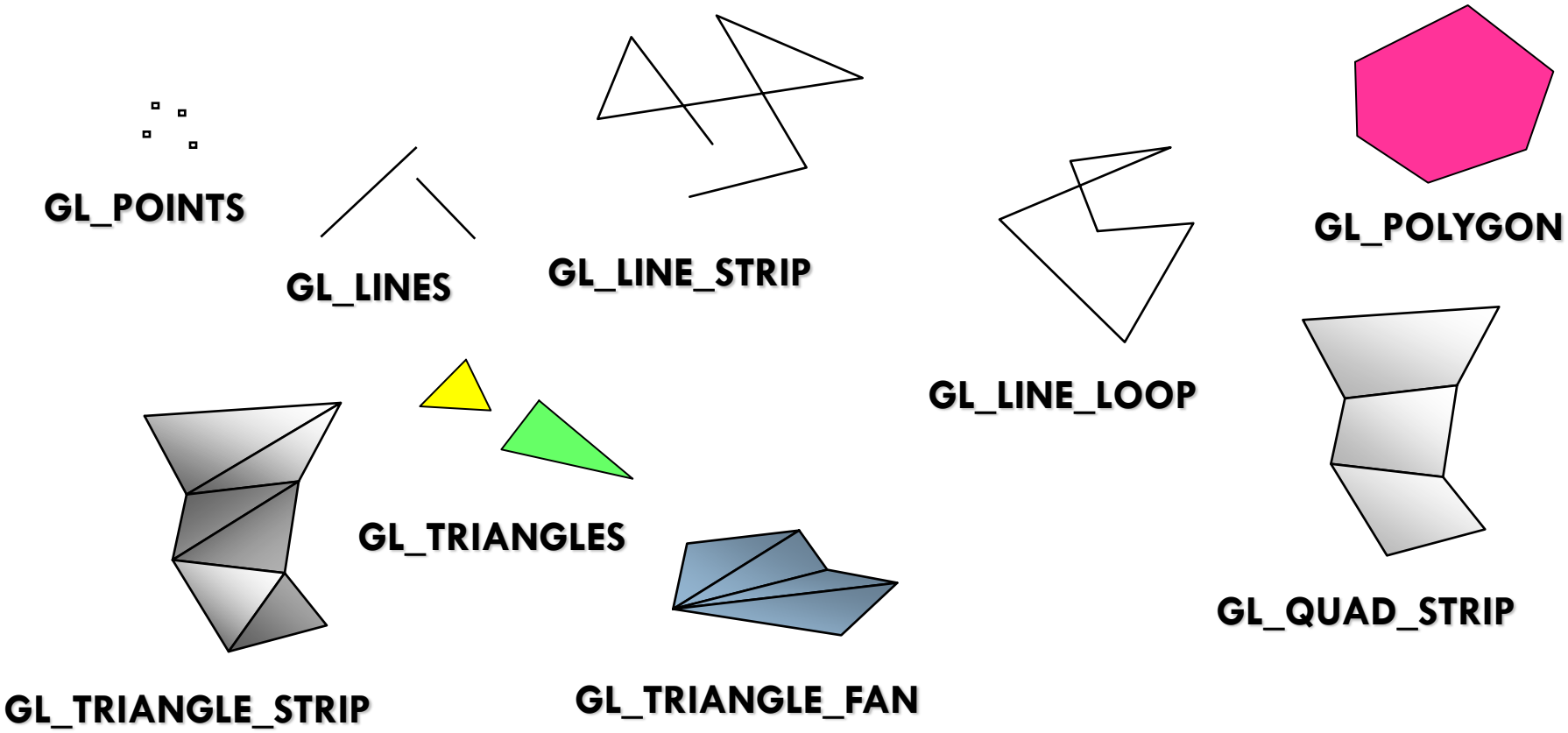
```
glLoadIdentity();
glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
```

Two- and three-dimensional viewing

- In **`glOrtho(left, right, bottom, top, near, far)`** the near and far distances are measured from the camera
- Two-dimensional vertex commands place all vertices in the plane z=0
- If the application is in two dimensions, we can use the function

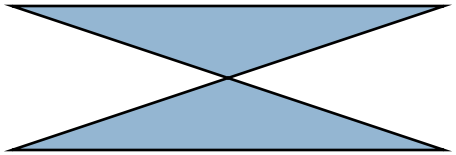
  `gluOrtho2D(left, right,bottom,top)`
- In two dimensions, the view or clipping volume becomes a *clipping window*

OpenGL Primitives



Polygon Issues

- OpenGL will only display polygons correctly 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
- User program can check if above true
  - OpenGL will produce output if these conditions are violated but it may not be what is desired
- Triangles satisfy all conditions



nonsimple polygon

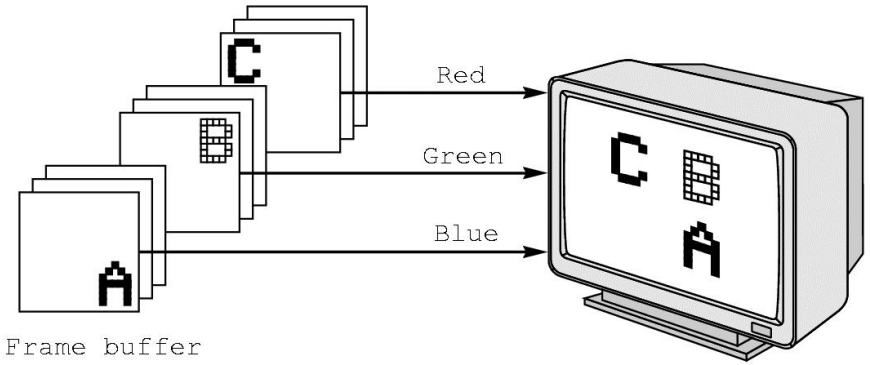
nonconvex polygon

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
    - Display vertices

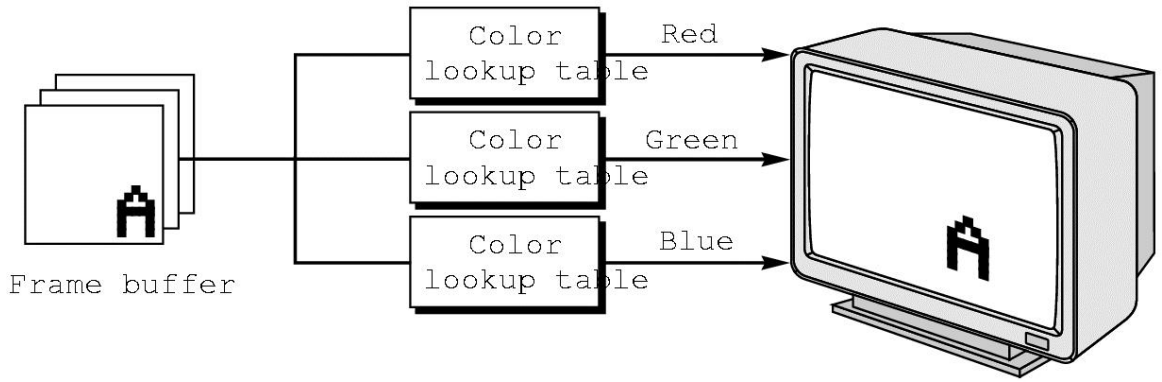
RGB color

- Each color component is stored separately in the frame buffer
- Usually 8 bits per component in buffer
- Note in **`glColor3f`** the color values range from 0.0 (none) to 1.0 (all), whereas in **`glColor3ub`** the values range from 0 to 255



Indexed Color

- Colors are indices into tables of RGB values
- Requires less memory
  - indices usually 8 bits
  - not as important now
    - Memory inexpensive
    - Need more colors for shading



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
```

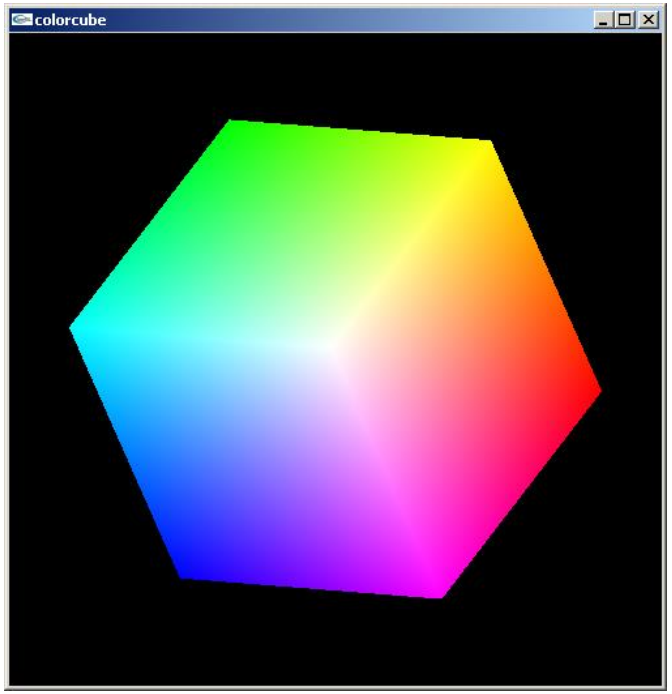
Smooth Color

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

  determines fill color
- **`glShadeModel`**

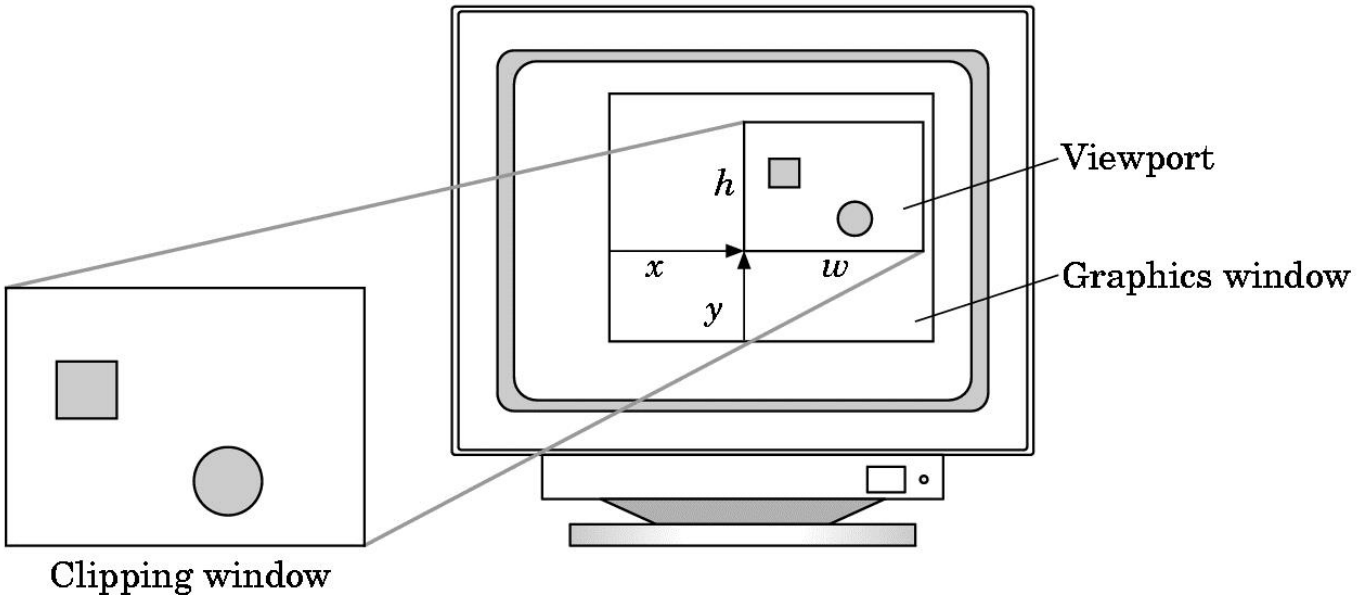
  `(GL_SMOOTH)`

  or **`GL_FLAT`**



Viewports

- Do not have use the entire window for the image: **`glViewport(x,y,w,h)`**
- Values in pixels (screen coordinates)

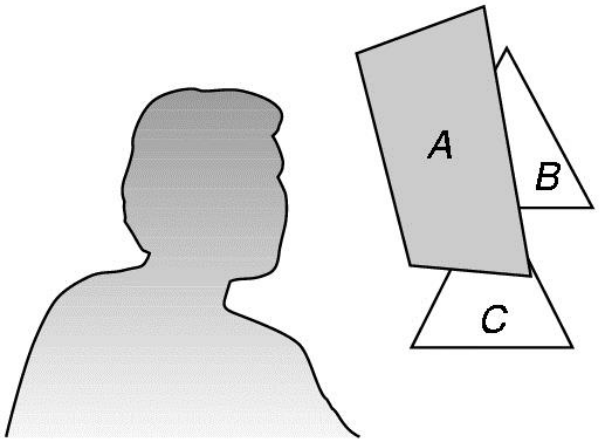


Three-dimensional Applications

- In OpenGL, two-dimensional applications are a special case of three-dimensional graphics
- Going to 3D
  - Not much changes
  - Use **`glVertex3*( )`**
  - Have to worry about the order in which polygons are drawn or use hidden-surface removal
  - Polygons should be simple, convex, flat

Hidden-Surface Removal

- We want to see only those surfaces in front of other surfaces
- OpenGL uses a *hidden-surface* method called the *z*-buffer algorithm that saves depth information as objects are rendered so that only the front objects appear in the image



Using the z-buffer Algorithm

- The algorithm uses an extra buffer, the z-buffer, to store depth information as geometry travels down the pipeline
- It must be
  - Requested in **`main.c`**
    - **`glutInitDisplayMode`**
      `(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH)`
  - Enabled in **`init.c`**
    - **`glEnable(GL_DEPTH_TEST)`**
  - Cleared in the display callback
    - **`glClear(GL_COLOR_BUFFER_BIT |`**
      `GL_DEPTH_BUFFER_BIT)`

Questions?

- Ask now or e-mail later
- Acknowledgements
  - Previous instructors at Purdue
    - David Ebert, ECE
    - Niklas Elmqvist, ECE
  - Previous instructors at Arizona state university
    - Ross Maciejewski
  - Textbook (Ed Angel)
  - Google Image Search
    - Copyright respective owners