Lecture 5
OpenGL Basics

Xin (Shane) Li
Sep. 3, 2009

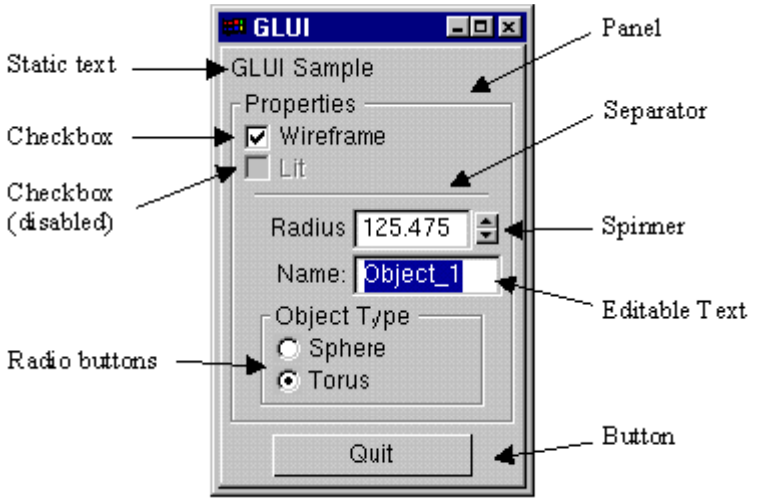
http://www.ece.lsu.edu/xinli/teaching/EE4700Fall2009.htm

What Is OpenGL?

- **A low-level graphics library (Interface to hardware)**
  - About 250 distinct commands (about 200 in the core OpenGL and another 50 in the OpenGL Utility Library)
  - "gl.h" and "glu.h"
- **Provides:**
  - Specialized data types
  - Functions to configure pipeline (most of them)
  - Functions to send geometry and images to it
  - Utility functions
- **Current Versions**
  - 1.2 supported by most drivers
  - 1.1 by MS library
  - Multiple extensions by hardware vendors
  - Latest version is 3.2(Aug. 3, 2009)
  - Check your OpenGL version: gl_version = glGetString(GL_VERSION);
- **Website**
  http://www.opengl.org/

Some Related Open-sourced Libraries

- **OpenGL Utility Library (GLU)**
  - Contains routines that use lower-level OpenGL commands to perform such tasks as performing polygon tessellation, rendering surfaces…
  - Provided as part of every OpenGL implementation
- **OpenGL Utility Toolkit (GLUT)**
  - Window-system independent toolkit
- **GLUI**
  - GLUT-based C++ user interface library which provides controls such as buttons, checkboxes, radio buttons, and spinners to OpenGL applications
  - window-system independent, relying on GLUT to handle all system-dependent issues, such as window and mouse management.
- **FLTK (Fast Light ToolKits)…**
  - http://www.fltk.org/
- …



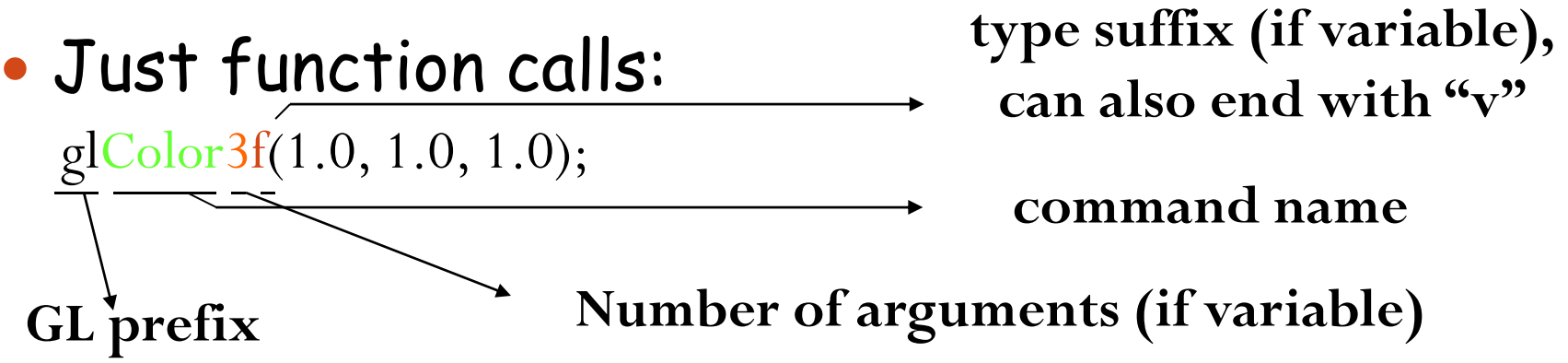
A Simple Example

Program1

We have seen it in our previous class.

OpenGL Commands Format

- Just function calls:



  - float box[3]; …… glVertex3fv(box);
- NO high level commands
- Same command, different arguments:
  glColor4ub(255,255,255,255);
  In most cases, same result, ***float*** recommended

OpenGL Data Types

b	8	GLbyte	signed char
s	16	GLshort	short
i	32	GLint, GLsizei	int / long
f	32	GLfloat, GLclampf	float
d	64	GLdouble, GLclampd	double
ub	8	GLubyte, Glboolean	unsigned char
us	16	GLushort	unsigned short
ui	32	GLuint, GLenum, GLbitfield	unsigned int

Particular Implementation may not follow this scheme

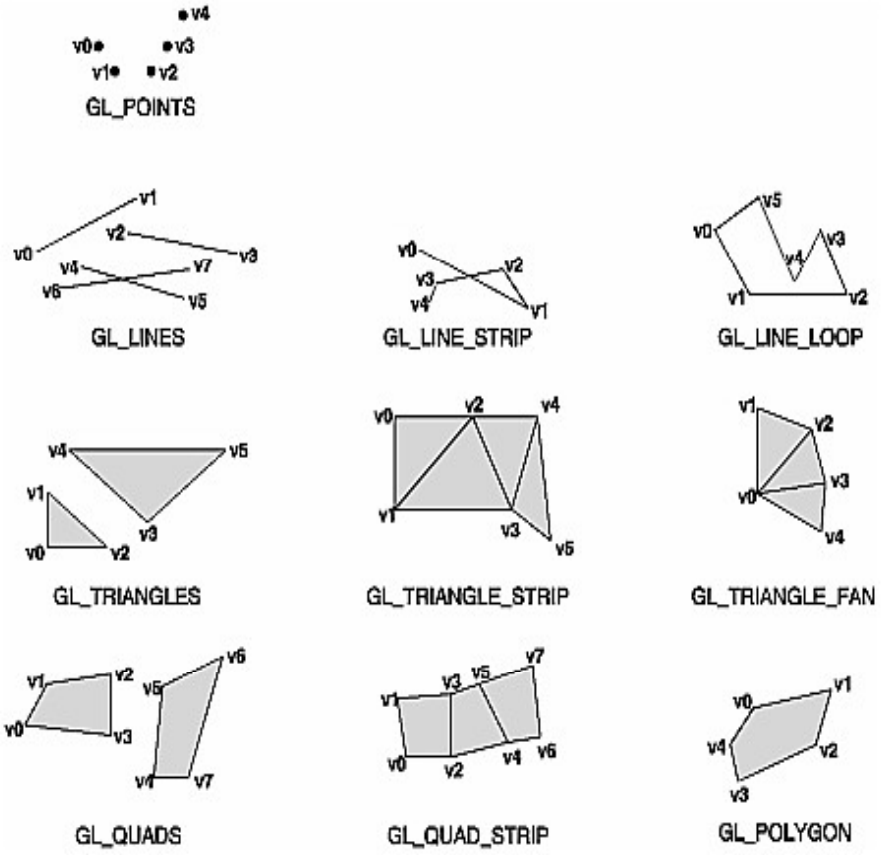
OpenGL Primitives

- **Geometric object is described by a set of vertices (glVertex*) and the type of the primitive to be drawn**
  - GL_POINTS
  - GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP
  - GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN
  - GL_QUADS, GL_QUAD_STRIP
  - GL_POLYGON

OpenGL Primitives

- **Example**

```
glBegin(GL_TRIANGLE_STRIP);
        glColor3f(1,0,0);
        glColor4f(0,1,0,1);
        glVertex3f(0,1.5,-2);
        glVertex3f(0,0.8,0);
        glVertex4f(-3.6,1.2,2,0.5);
        glVertex2f(-7.8,1);
        glVertex3f(0,0,0.6);
glEnd();
glFlush(); //glutSwapBuffers();
```



OpenGL as a State Machine

- **OpenGL is a state machine**
  - It maintains many state variables (line width, color, … etc.)
  - eg. GL_FLAT & GL_SMOOTH are two states of GL_SHADE_MODEL
  - Some states are binary states that are either GL_FALSE or GL_TRUE
  - Each state variable has a default value.
- **Changing states**
  - For binary state variables, use glEnable/glDisable (eg. GL_LIGHTING)
  - Mode state variables require specific commands (eg. glShadeModel)
  - Value state variables require specific commands too. (eg. glColor3f)
- **Remains in effect until changed!**
- **More examples**
  - glPointSize(GLfloat size);
  - glLineWidth(GLfloat width);

State Queries

- **Checking if enabled: glIsEnabled(GLenum cap)**
  - Cap: symbolic constant indicating an OpenGL capability (eg. GL_DEPTH_TEST)
  - Returns Glboolean
- **Getting state variable value**
  - glGetIntegerv(pname, params)
  - Stores value of pname at location paprams
  - Similarly, glGetBooleanv, glGetFloatv, etc.
- **More specific queries for some important variables**
  - glGetString(GL_VERSION)
  - glGetLight()
  - glGetError()

Vertex Interpretation

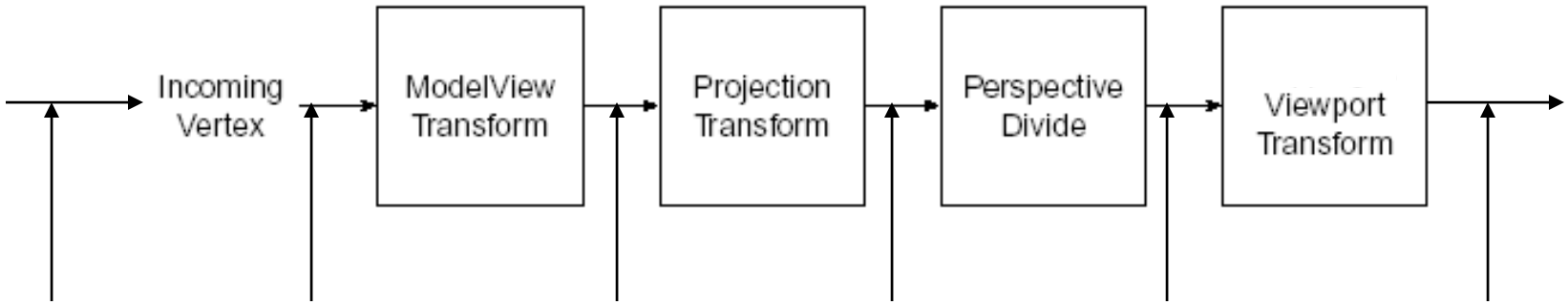
- **Polygons:**
  - All vertices make single polygon
  - Must be convex, no self-intersections
    - Results undefined otherwise
- **If insufficient data?**
  - nothing is drawn
- **If more than necessary?**
  - extra ignored
  - eg. 4 vertices sent after glBegin(GL_TRIANGLES)
- **Strip and fan modes are more efficient**
  - Why?

Depth Testing

- **determining what falls in front of what...**
  - z-buffer
    - tests each pixel to see if it is unhidden and should therefore be drawn
    - In your GL initialization
      glEnable(GL_DEPTH_TEST);
    - clear your z-buffer at the end of every frame
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

OpenGL Coordinate System

- **OpenGL Pipeline**



Object coords

World coords

Viewing Coords (Eye Coords)

Clipping coords

Normalized Device Coords

Window coords

Check details at:
http://research.cs.queensu.ca/~jstewart/454/notes/pipeline/

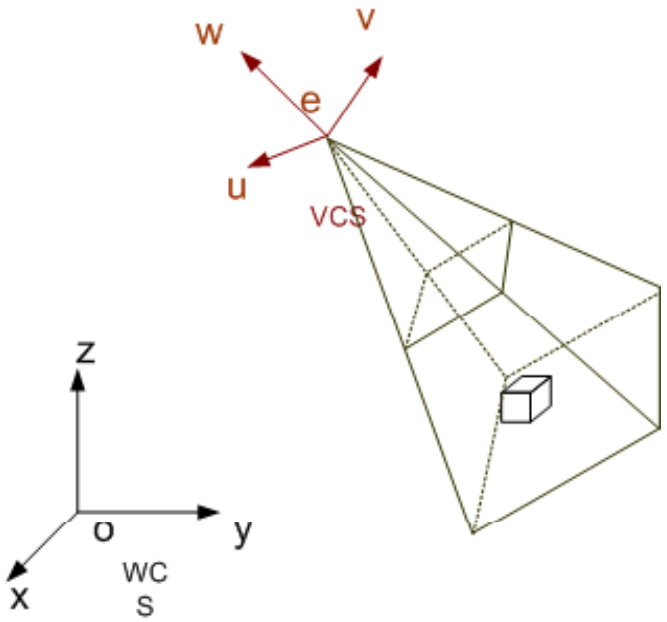
Viewing and Modeling Transformations

- **Modeling Transformations**
  - void glTranslatef(float x, float y, float z);
  - void glRotatef(float angle, float x, float y, float z);
  - void glScalef(float x, float y, float z);
  - Your own matrix:
    - float m[] = {…}
    - glMultMatrixf(m)

- **Viewing Transformations**
  - void gluLookAt(GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ, GLdouble centerX, GLdouble centerY, GLdouble centerZ, GLdouble upX, GLdouble upY, GLdouble upZ)
    - defines a line of sight (most convenient)
    - encapsulates a series of rotation and translation
    - Same effect can be achieved by glTranslate*(), glRotate*(), glScale*() …

Projection Transformations

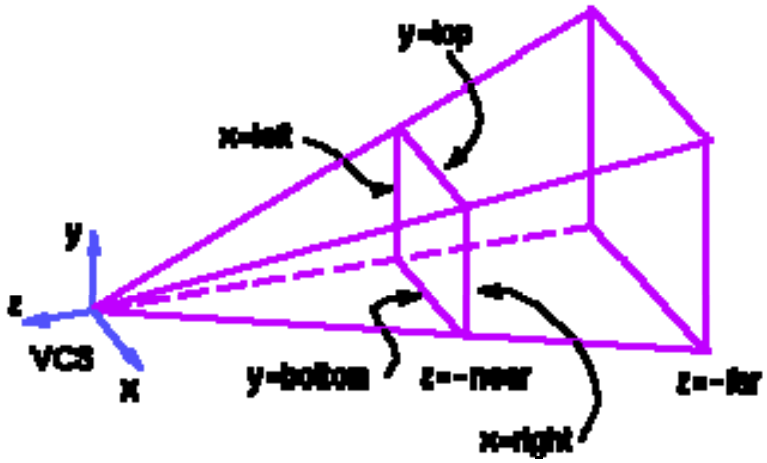
- **Perspective Projection**
  - Things farther away get smaller
  - Parallel lines no longer parallel: vanishing point



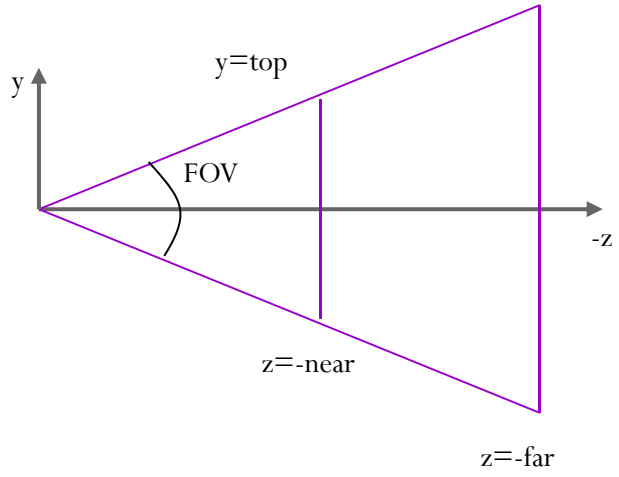
Viewing Coordinate System (*VCS*)

Projection Transformations(cont.)

- **Perspective Projection**
  - void glFrustum(double left, double right, double bottom, double top, double near, double far);

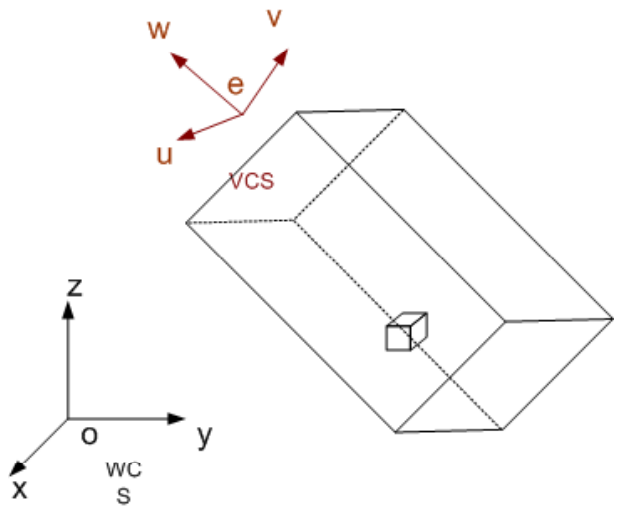


  - void gluPerspective(double fovy, double aspect, double near, double far);

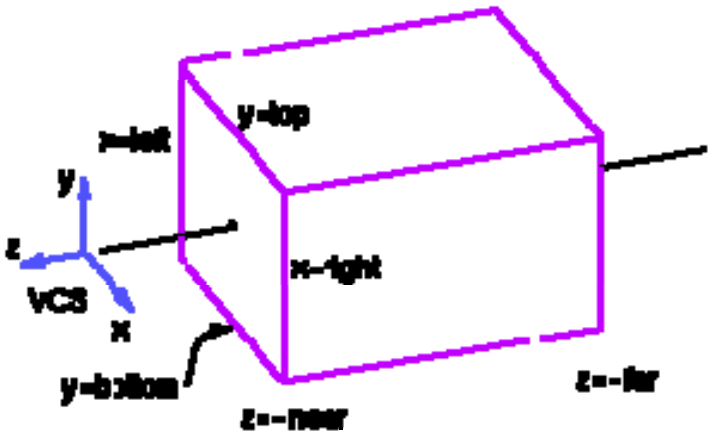


Projection Transformations(cont.)

- **Orthographic Projection**
  - void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar);
  - void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);



- **Routines:**
  - First, put:
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
  - Then, put one of:
    glFrustrum/gluPerspective or glOrtho



Viewport Transformation

- **The final transform of the pipeline**
- **Specify image size in pixels**
  - Analogy: choosing the size of your photo paper
- **Use glViewport(x,y,width, height)**
- **Typically:**
  - x,y=0
  - width, height – those of the window
- **Possible distortion of image?**
  - eg. gluOrtho2d(-100, 100, -100, 100); glViewport(0,0, 200, 100);

GLUT

- void glutInit(int *argcp, char **argv);
  - glutInit will initialize the GLUT library and negotiate a session with the window system
- void glutInitDisplayMode(unsigned int mode);
  - Sets the initial display mode.
- void glutInitWindowSize(int width , int height);
- void glutInitWindowPosition(int x, int y);
- int glutCreateWindow(char *name);
- void glutDisplayFunc(void (*func)(void));
- void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));
- void glutMouseFunc(void (*func)(int button, int state, int x, int y));
- void glutReshapeFunc(void (*func)(int width, int height));
- void glutMainLoop(void);
- ……

Typical OpenGL Program Organization

- **main:**

  find GL visual and create window
  initialize GL states (e.g. viewing, color, lighting)
  initialize display lists
  loop
  check for events (and process them)
  if window event (window moved, exposed, etc.)
  modify viewport, if needed
  redraw
  else if mouse or keyboard
  do something, e.g., change states and redraw

- **redraw:**

  clear screen (to background color)
  change state(s), if needed
  render some graphics
  change more states
  render some more graphics
  .
  .
  .
  swap buffers

Resources

- Some useful links listed at:
  www.ece.lsu.edu/xinli/OpenGL/GLUTSetup.htm

- Google…