



## CIS 636

# Introduction to Computer Graphics

## CG Basics 3 of 8: OpenGL Primer, Part 1 of 3

William H. Hsu

Department of Computing and Information Sciences, KSU

KSOL course pages: <http://snipurl.com/1y5gc>

Course web site: <http://www.kddresearch.org/Courses/CIS636>

Instructor home page: <http://www.cis.ksu.edu/~bhsu>

### Readings:

All slides from SIGGRAPH 2000 tutorial on OpenGL, Shreiner, Angel, Shreiner:

<http://www.cs.unm.edu/~angel/SIGGRAPH/>

Sections 2.4 – 2.6, Eberly 2<sup>e</sup> – see <http://snurl.com/1ye72>

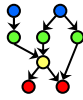
NeHe tutorials: 1 – 10, <http://nehe.gamedev.net>



## Lecture Outline

- **Three tutorials from SIGGRAPH 2000**
- **Vicki Shreiner: OpenGL and GL Utility Toolkit (GLUT)**
  - \* Overall architecture
  - \* Initialization
  - \* Viewport management
- **Vicki Shreiner: Basic Rendering**
  - \* Primitives
  - \* Data types
  - \* Rendering commands: syntax
  - \* Automated part: line and polygon scan conversion
- **Ed Angel: 3-D Viewing**
  - \* Math background (see CG Basics 1)
  - \* Viewing and normalization transformations (see CG Basics 2)
  - \* More on viewing in CG Basics 4
  - \* View volume specification
  - \* Automated part: clipping





## An Interactive Introduction to OpenGL Programming

Dave Shreiner  
Ed Angel  
Vicki Shreiner

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## What You'll See Today

- General OpenGL Introduction
- Rendering Primitives
- Rendering Modes
- Lighting
- Texture Mapping
- Additional Rendering Attributes
- Imaging

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## Goals for Today

- **Demonstrate enough OpenGL to write an interactive graphics program with**
  - \* custom modeled 3D objects or imagery
  - \* lighting
  - \* texture mapping
- **Introduce advanced topics for future investigation**

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## OpenGL and GLUT Overview

Vicki Shreiner

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## OpenGL and GLUT Overview

- What is OpenGL & what can it do for me?
- OpenGL in windowing systems
- Why GLUT
- A GLUT program template

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## What Is OpenGL?

- **Graphics rendering API**
  - \* high-quality color images composed of geometric and image primitives
  - \* window system independent
  - \* operating system independent

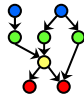
© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

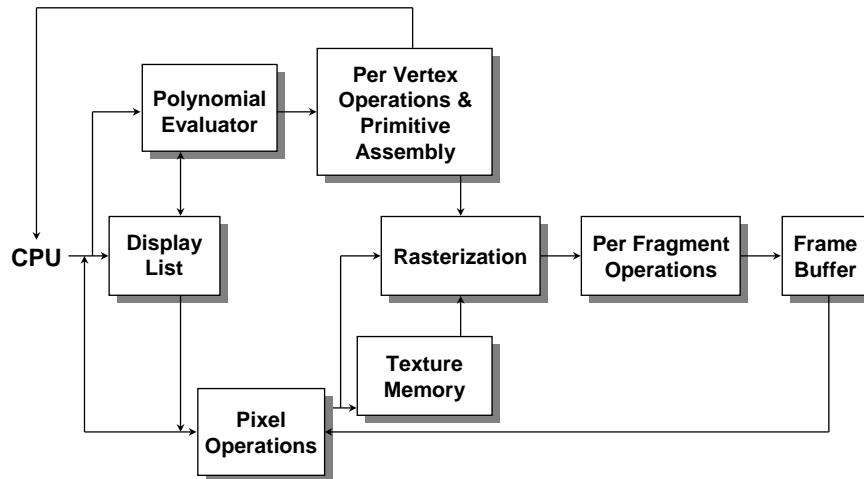
CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## OpenGL Architecture



© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## OpenGL as a Renderer

- **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.

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Related APIs

- **AGL, GLX, WGL**
  - \* glue between OpenGL and windowing systems
- **GLU (OpenGL Utility Library)**
  - \* part of OpenGL
  - \* NURBS, tessellators, quadric shapes, etc.
- **GLUT (OpenGL Utility Toolkit)**
  - \* portable windowing API
  - \* not officially part of OpenGL

© 2000 Shreiner, D., Angel, E., Shreiner, V.

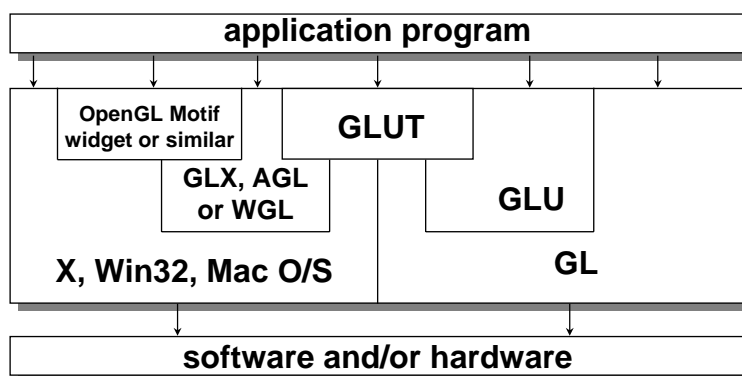
CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## OpenGL and Related APIs



© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## Preliminaries

- **Headers Files**
  - ⇒ #include <GL/gl.h>
  - ⇒ #include <GL/glu.h>
  - ⇒ #include <GL/glut.h>
- **Libraries**
- **Enumerated Types**
  - \* **OpenGL defines numerous types for compatibility**
    - ◆ GLfloat, GLint, GLenum, etc.

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## GLUT Basics

- **Application Structure**
  - \* **Configure and open window**
  - \* **Initialize OpenGL state**
  - \* **Register input callback functions**
    - ⇒ render
    - ⇒ resize
    - ⇒ input: keyboard, mouse, etc.
  - \* **Enter event processing loop**

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## Sample Program

```
void main( int argc, char** argv )
{
    int mode = GLUT_RGB|GLUT_DOUBLE;
    glutInitDisplayMode( mode );
    glutCreateWindow( argv[0] );
    init();
    glutDisplayFunc( display );
    glutReshapeFunc( resize );
    glutKeyboardFunc( key );
    glutIdleFunc( idle );
    glutMainLoop();
}
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## OpenGL Initialization

- Set up whatever state you're going to use

```
void init( void )
{
    glClearColor( 0.0, 0.0, 0.0, 1.0 );
    glClearDepth( 1.0 );

    glEnable( GL_LIGHT0 );
    glEnable( GL_LIGHTING );
    glEnable( GL_DEPTH_TEST );
}
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## GLUT Callback Functions

- Routine to call when something happens
  - \* window resize or redraw
  - \* user input
  - \* animation

- “Register” callbacks with GLUT

```
glutDisplayFunc( display );  
glutIdleFunc( idle );  
glutKeyboardFunc( keyboard );
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Rendering Callback

- Do all of your drawing here

```
glutDisplayFunc( display );
```

```
void display( void )  
{  
    glClear( GL_COLOR_BUFFER_BIT );  
    glBegin( GL_TRIANGLE_STRIP );  
        glVertex3fv( v[0] );  
        glVertex3fv( v[1] );  
        glVertex3fv( v[2] );  
        glVertex3fv( v[3] );  
    glEnd();  
    glutSwapBuffers();  
}
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## Idle Callbacks

- Use for animation and continuous update

```
glutIdleFunc( idle );
```

```
void idle( void )  
{  
    t += dt;  
    glutPostRedisplay();  
}
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## User Input Callbacks

- Process user input

```
glutKeyboardFunc( keyboard );
```

```
void keyboard( char key, int x, int y )  
{  
    switch( key ) {  
        case 'q' : case 'Q' :  
            exit( EXIT_SUCCESS );  
            break;  
        case 'r' : case 'R' :  
            rotate = GL_TRUE;  
            break;  
    }  
}
```

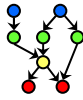
© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





# Elementary Rendering

Vicki Shreiner

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



# Elementary Rendering

- Geometric Primitives
- Managing OpenGL State
- OpenGL Buffers

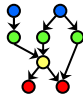
© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

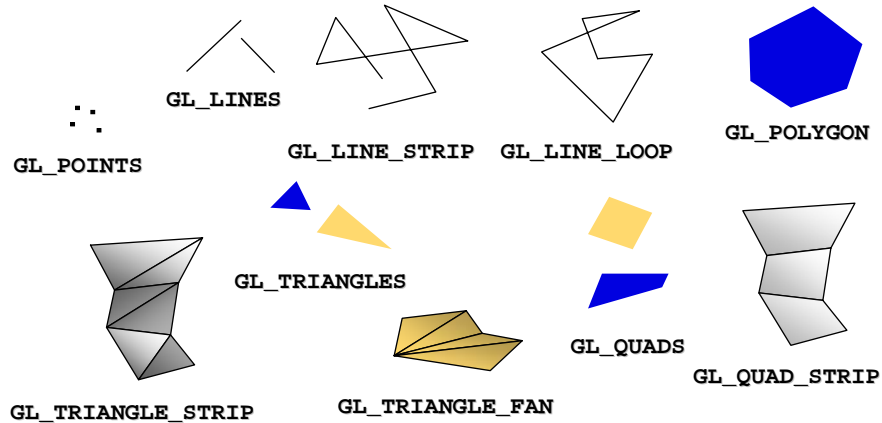
Computing & Information Sciences  
Kansas State University





## OpenGL Geometric Primitives

- All geometric primitives are specified by vertices



© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## 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();
}
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## OpenGL Command Formats

`glVertex3fv( v )`

### Number of components

2 - (x,y)  
3 - (x,y,z)  
4 - (x,y,z,w)

### Data Type

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

### Vector

omit "v" for scalar form  
`glVertex2f( x, y )`

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Specifying Geometric Primitives

- Primitives are specified using

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

- \* *primType* determines how vertices are combined

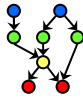
```
GLfloat red, green, blue;  
GLfloat coords[3];  
glBegin( primType );  
for ( i = 0; i < nVerts; ++i ) {  
    glColor3f( red, green, blue );  
    glVertex3fv( coords );  
}  
glEnd();
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

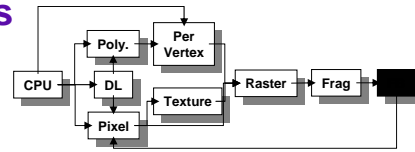
CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University

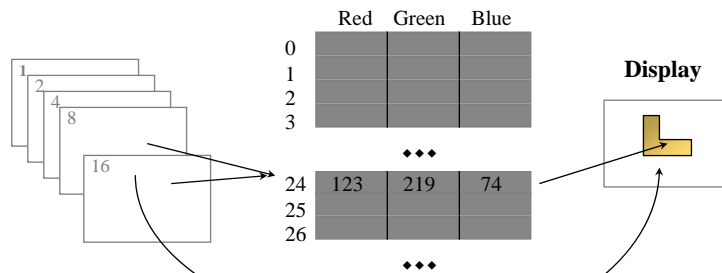


## OpenGL Color Models

- **RGBA or Color Index**



*color index mode*



*RGBA mode*

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Shapes Tutorial

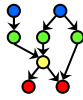
```
glBegin (GL_TRIANGLE_STRIP);
glColor3f (1.00 , 0.00 , 1.00 );
glVertex2f (0.0 , 25.0 );
glColor3f (0.00 , 1.00 , 1.00 );
glVertex2f (50.0 , 150.0 );
glColor3f (0.00 , 1.00 , 0.00 );
glVertex2f (125.0 , 100.0 );
glColor3f (1.00 , 1.00 , 0.00 );
glVertex2f (175.0 , 200.0 );
glEnd();
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

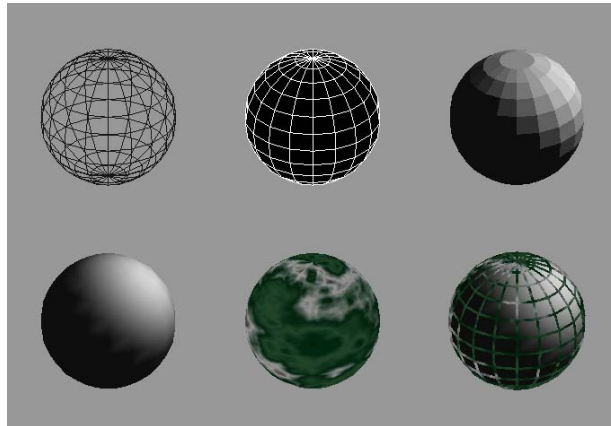
CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Controlling Rendering Appearance

- From Wireframe to Texture Mapped



© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## OpenGL's State Machine

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

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## 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*()
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Controlling current state

- Setting State

```
glPointSize( size );  
glLineStipple( repeat, pattern );  
glShadeModel( GL_SMOOTH );
```

- Enabling Features

```
glEnable( GL_LIGHTING );  
glDisable( GL_TEXTURE_2D );
```

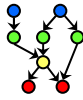
© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





# Transformations

Ed Angel

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



# Transformations in OpenGL

- **Modeling**
- **Viewing**
  - \* orient camera
  - \* projection
- **Animation**
- **Map to screen**

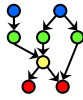
© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

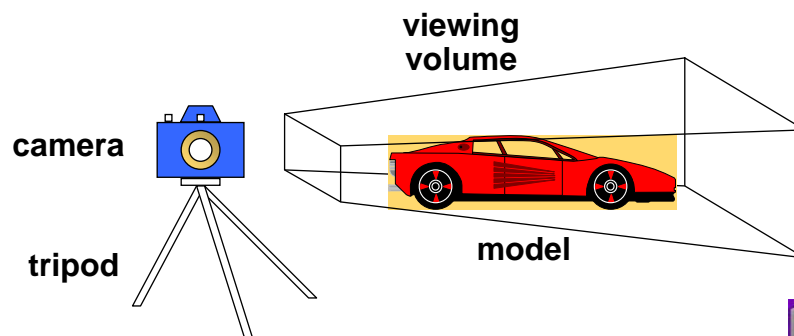
Computing & Information Sciences  
Kansas State University





## Camera Analogy

- 3D is just like taking a photograph (lots of photographs!)



© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Camera Analogy and Transformations

- **Projection transformations**
  - \* adjust the lens of the camera
- **Viewing transformations**
  - \* tripod—define position and orientation of the viewing volume in the world
- **Modeling transformations**
  - \* moving the model
- **Viewport transformations**
  - \* enlarge or reduce the physical photograph

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Coordinate Systems and Transformations

- **Steps in Forming an Image**
  - \* specify geometry (world coordinates)
  - \* specify camera (camera coordinates)
  - \* project (window coordinates)
  - \* map to viewport (screen coordinates)
- **Each step uses transformations**
- **Every transformation is equivalent to a change in coordinate systems (frames)**

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Affine Transformations

- **Want transformations which preserve geometry**
  - \* lines, polygons, quadrics
- **Affine = line preserving**
  - \* Rotation, translation, scaling
  - \* Projection
  - \* Concatenation (composition)

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## Homogeneous Coordinates

- \* each vertex is a column vector

$$\vec{v} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- \*  $w$  is usually 1.0
- \* all operations are matrix multiplications
- \* directions (directed line segments) can be represented with  $w = 0.0$

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## 3D Transformations

- A vertex is transformed by 4 x 4 matrices
  - \* all affine operations are matrix multiplications
  - \* all matrices are stored column-major in OpenGL
  - \* matrices are always post-multiplied
  - \* product of matrix and vector is

$$\mathbf{M} = \begin{bmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{bmatrix} \mathbf{M}\vec{v}$$

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## Specifying Transformations

- **Programmer has two styles of specifying transformations**
  - \* specify matrices (`glLoadMatrix`, `glMultMatrix`)
  - \* specify operation (`glRotate`, `glOrtho`)
- **Programmer does not have to remember the exact matrices**
  - \* check appendix of Red Book (Programming Guide)

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Programming Transformations

- **Prior to rendering, view, locate, and orient:**
  - \* eye/camera position
  - \* 3D geometry
- **Manage the matrices**
  - \* including matrix stack
- **Combine (composite) transformations**

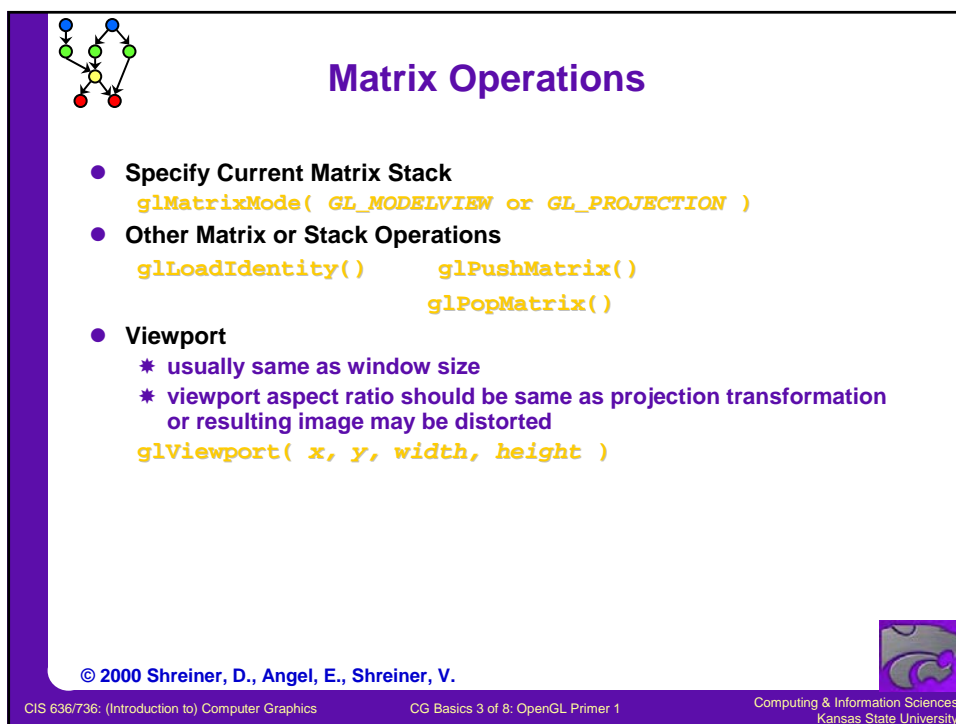
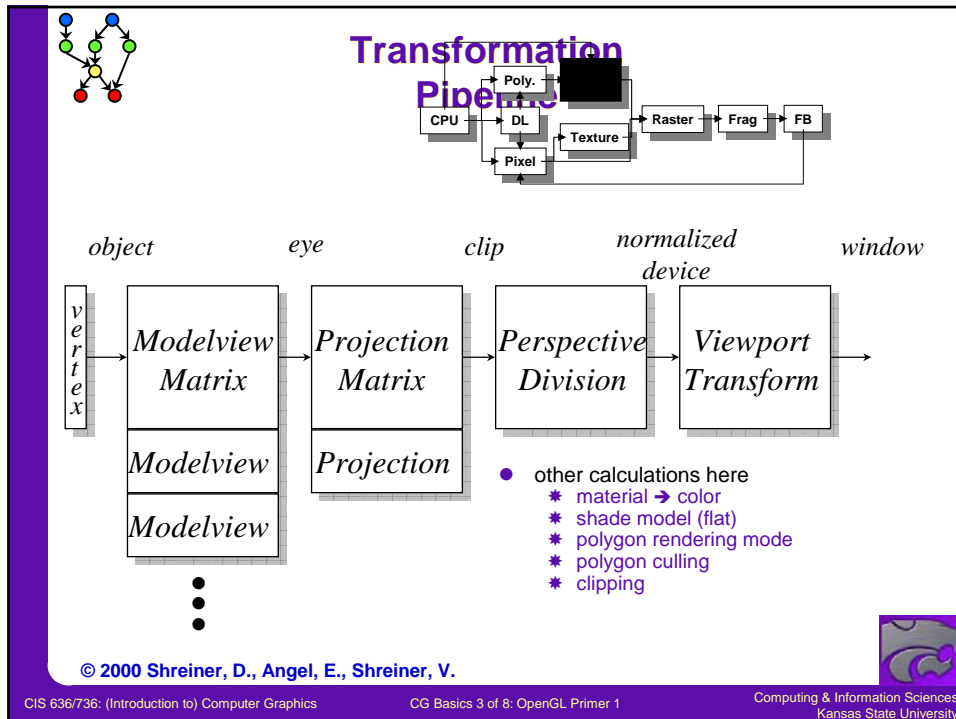
© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University







## Projection Transformation

- Shape of viewing frustum

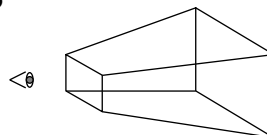
- Perspective projection

```
gluPerspective( fovy, aspect, zNear, zFar )  
glFrustum( left, right, bottom, top, zNear, zFar )
```

- Orthographic parallel projection

```
glOrtho( left, right, bottom, top, zNear, zFar )  
gluOrtho2D( left, right, bottom, top )
```

⇒ calls glOrtho with z values near zero



© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

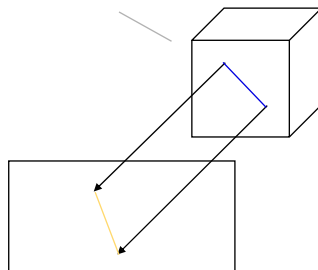
Computing & Information Sciences  
Kansas State University



## Applying Projection Transformations

- Typical use (orthographic projection)

```
glMatrixMode( GL_PROJECTION );  
glLoadIdentity();  
glOrtho( left, right, bottom, top, zNear, zFar );
```



© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

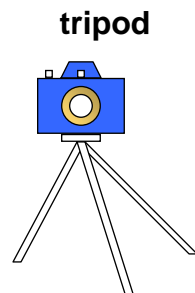
CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Viewing Transformations

- Position the camera/eye in the scene
  - \* place the tripod down; aim camera
- To “fly through” a scene
  - \* change viewing transformation and redraw scene
- `gluLookAt( eye_x, eye_y, eye_z, aim_x, aim_y, aim_z, up_x, up_y, up_z )`
  - \* up vector determines unique orientation
  - \* careful of degenerate positions



© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Projection Tutorial

```
fovy aspect zNear zFar
gluPerspective( 60.0 , 1.00 , 1.0 , 10.0 );
gluLookAt( 0.00 , 0.00 , 2.00 , <- eye
           0.00 , 0.00 , 0.00 , <- center
           0.00 , 1.00 , 0.00 ); <- up
```

Click on the arguments and move the mouse to modify values.

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Modeling Transformations

- Move object  
`glTranslate{fd}( x, y, z )`
- Rotate object around arbitrary axis  
`glRotate{fd}( angle, x, y, z )`  
\* angle is in degrees
- Dilate (stretch or shrink) or mirror object  
`glScale{fd}( x, y, z )`

$(x \ y \ z)$

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Transformation Tutorial

```
glTranslatef( 0.00 , 0.00 , 0.00 );
glRotatef( -52.0 , 0.00 , 1.00 , 0.00 );
glScalef( 1.00 , 1.00 , 1.00 );
glBegin( ... );
...
```

Click on the arguments and move the mouse to modify values.

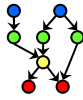
© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## Connection: Viewing and Modeling

- Moving camera is equivalent to moving every object in the world towards a stationary camera
- Viewing transformations are equivalent to several modeling transformations

`gluLookAt()` has its own command

can make your own *polar view* or *pilot view*

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

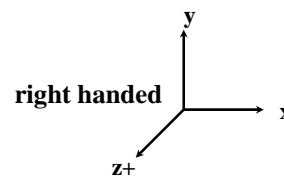
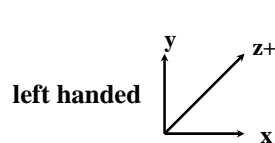
CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Projection is left handed

- Projection transformations (`gluPerspective`, `glOrtho`) are left handed
  - \* think of  $z_{Near}$  and  $z_{Far}$  as distance from view point
- Everything else is right handed, including the vertices to be rendered



© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## Common Transformation Usage

- 3 examples of `resize()` routine
  - \* restate projection & viewing transformations
- Usually called when window resized
- Registered as callback for `glutReshapeFunc()`

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## `resize()`: Perspective & LookAt

```
void resize( int w, int h )
{
    glViewport( 0, 0, (GLsizei) w, (GLsizei) h );
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluPerspective( 65.0, (GLfloat) w / h,
                  1.0, 100.0 );
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    gluLookAt( 0.0, 0.0, 5.0,
              0.0, 0.0, 0.0,
              0.0, 1.0, 0.0 );
}
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## resize(): Perspective & Translate

- Same effect as previous LookAt

```
void resize( int w, int h )
{
    glViewport( 0, 0, (GLsizei) w, (GLsizei) h );
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluPerspective( 65.0, (GLfloat) w/h,
                  1.0, 100.0 );
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    glTranslatef( 0.0, 0.0, -5.0 );
}
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

Computing & Information Sciences  
Kansas State University



## resize(): Ortho (part 1)

```
void resize( int width, int height )
{
    GLdouble aspect = (GLdouble) width / height;
    GLdouble left = -2.5, right = 2.5;
    GLdouble bottom = -2.5, top = 2.5;
    glViewport( 0, 0, (GLsizei) w, (GLsizei) h );
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    ... continued ...
}
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## resize(): Ortho (part 2)

```
if ( aspect < 1.0 ) {
    left /= aspect;
    right /= aspect;
} else {
    bottom *= aspect;
    top *= aspect;
}
glOrtho( left, right, bottom, top, near, far );
glMatrixMode( GL_MODELVIEW );
glLoadIdentity();
}
```

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Compositing Modeling Transformations

- **Problem 1: hierarchical objects**
  - \* one position depends upon a previous position
  - \* robot arm or hand; sub-assemblies
- **Solution 1: moving local coordinate system**
  - \* modeling transformations move coordinate system
  - \* post-multiply column-major matrices
  - \* OpenGL post-multiplies matrices

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Compositing Modeling Transformations

- **Problem 2: objects move relative to absolute world origin**
  - \* **my object rotates around the wrong origin**
    - ⇒ make it spin around its center or something else
- **Solution 2: fixed coordinate system**
  - \* **modeling transformations move objects around fixed coordinate system**
  - \* **pre-multiply column-major matrices**
  - \* **OpenGL post-multiplies matrices**
  - \* **must reverse order of operations to achieve desired effect**

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Additional Clipping Planes

- **At least 6 more clipping planes available**
- **Good for cross-sections**
- **Modelview matrix moves clipping plane**
- **clipped**
- `glEnable( GL_CLIP_PLANEi )`
- `glClipPlane( GL_CLIP_PLANEi, GLdouble* coeff )`

$$Ax + By + Cz + D < 0$$

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## Reversing Coordinate Projection

- **Screen space back to world space**

```
glGetIntegerv( GL_VIEWPORT, GLint viewport[4] )
glGetDoublev( GL_MODELVIEW_MATRIX, GLdouble mvmatrix[16] )
glGetDoublev( GL_PROJECTION_MATRIX,
              GLdouble projmatrix[16] )
gluUnProject( GLdouble winx, winy, winz,
             mvmatrix[16], projmatrix[16],
             GLint viewport[4],
             GLdouble *objx, *objy, *objz )
```

- **gluProject goes from world to screen space**

© 2000 Shreiner, D., Angel, E., Shreiner, V.

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University



## Summary

- **Three tutorials from SIGGRAPH 2000**
- **Vicki Shreiner: OpenGL and GL Utility Toolkit (GLUT)**
  - \* Overall architecture
  - \* Initialization
  - \* Viewport management
- **Vicki Shreiner: Basic Rendering**
  - \* Primitives
  - \* Data types
  - \* Rendering commands: syntax
  - \* Automated part: line and polygon scan conversion
- **Ed Angel: 3-D Viewing**
  - \* Math background (see CG Basics 1)
  - \* Viewing and normalization transformations (see CG Basics 2)
  - \* More on viewing in CG Basics 4
  - \* View volume specification
  - \* Automated part: clipping

CIS 636/736: (Introduction to) Computer Graphics

CG Basics 3 of 8: OpenGL Primer 1

Computing & Information Sciences  
Kansas State University





## Terminology

- **Aspects of Graphics**
  - \* Geometry
  - \* Rendering
  - \* Animation
  - \* Imaging
- **OpenGL and GL Utility Toolkit (GLUT)**
  - \* State machine
  - \* Using GLUT
  - \* Specifying perspective, parallel projections
- **Modelview Transformation**
- **Normalizing Transformation**
- **Viewing Transformation**



## Next: Viewing, OpenGL Tutorial 2

- **Vicki Shreiner: Animation and Depth Buffering**
  - \* Double buffering
  - \* Lights: positioning
  - \* Illumination: light models, attenuation
  - \* Material properties
  - \* Animation basics in OpenGL
- **Vicki Schreiner: Imaging and Raster Primitives**
- **Ed Angel: Texture Mapping**
- **Dave Shreiner: Advanced Topics**
  - \* Display lists and vertex arrays
  - \* Accumulation buffer
  - \* Fog
  - \* Stencil buffering
  - \* Fragment programs (to be concluded in Tutorial 3)

