

Assignment 3

Introduction to OpenGL (1)

Ming Chuang

T.A CS – 357 / Computer Graphics 2009

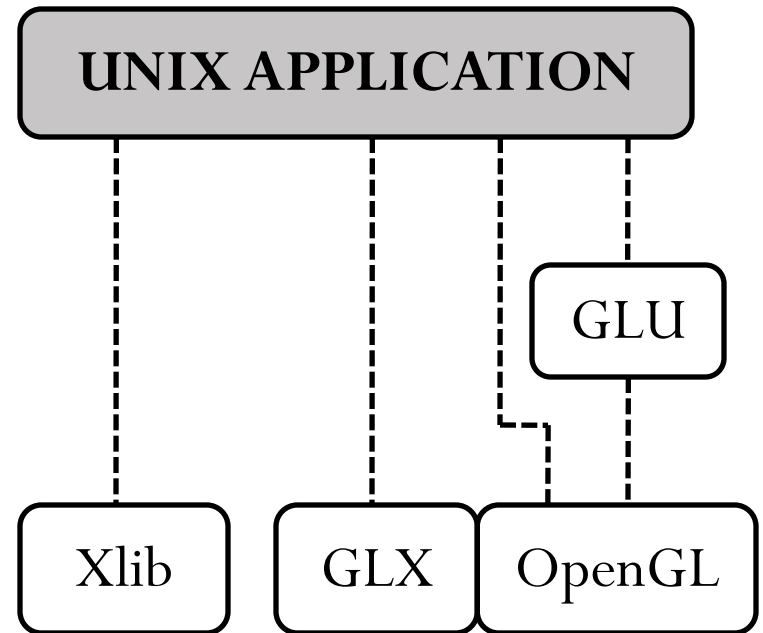
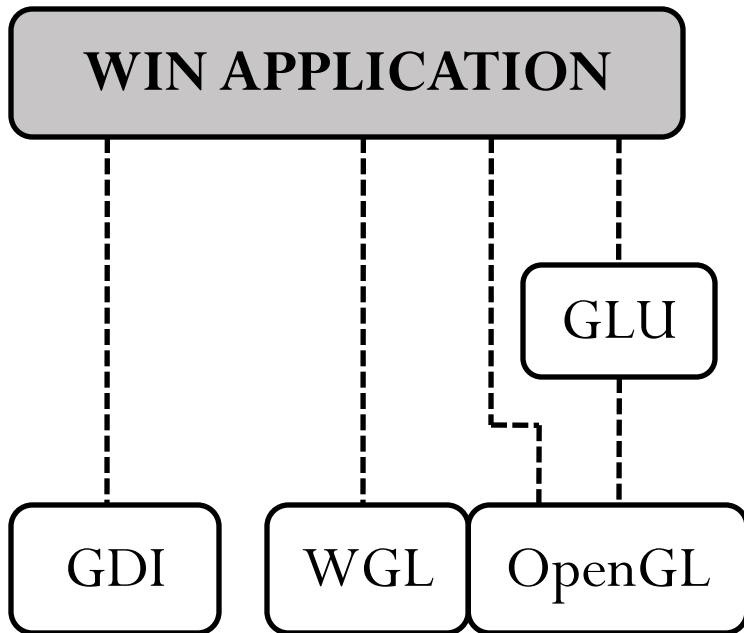
Outline

- What is Open GL
- Getting Started
- Matrix Manipulation
- Drawing
- Example Program

What is OpenGL?

- A 2D / 3D graphics rendering API
- Cross Language / Cross Platform with related API
- Related API
 - ◇ GLU (Open GL Utility library)
 - ◇ AGL, GLX, WGL
 - ◇ GLUT

API Hierarchy



GLU / GLUT

□ GLU

- ◇ Provides additional routines
- ◇ Quadric Surfaces, NURBS, Tessellation
- ◇ Spheres, Disks and Cylinders

□ GLUT

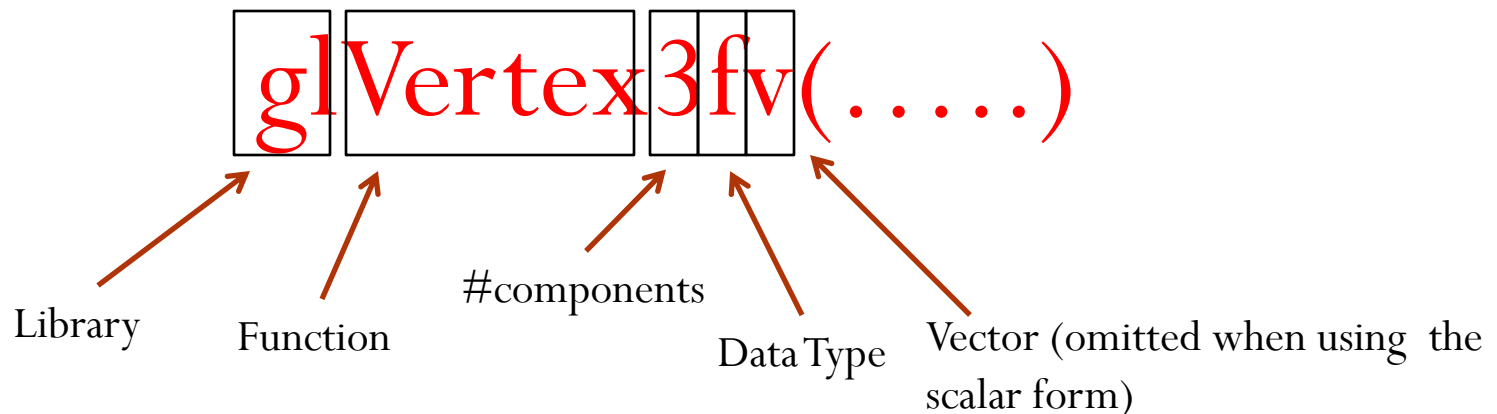
- ◇ Window system independent toolkit
- ◇ Implements simple windowing application programming interface
- ◇ Includes functions for monitoring keyboard and mouse

Getting Started

- Function Name Prefixes
 - GL API calls start with “gl....”
 - glOrtho(...), glVertex3f(...)
 - GLU API calls start with “glu....”
 - glusphere(...), glulookat(...)
 - GLUT API calls start with “glut....”
 - glutInitDisplayMode(...), glutCreateWindow(....)

Getting Started

- Function Name Prefixes
 - GL API calls start with “gl...”
 - glOrtho(...), glVertex3f(...)
 - GLU API calls start with “glu...”
 - glusphere(...), glulookat(...)
 - GLUT API calls start with “glut...”
 - glutInitDisplayMode(...), glutCreateWindow(...)



Steps

1. Create a window
2. Initialize Open GL states
 - Setup lighting, camera position, matrices etc.
3. Per frame setup
 - Viewpoint, clearing buffers etc.
4. Draw frame
5. Process Events
 - Keyboard / Mouse input
6. Loop 3 -

Steps

1. Create a window
2. Initialize Open GL states
 - Setup lighting, camera position, matrices etc.
3. Per frame setup
 - Viewpoint, clearing buffers etc.
4. Draw frame
5. Process Events
 - Keyboard / Mouse input
6. Loop 3 -



By registering a set
of callback functions

Sample

- `void main(int argc, char *argv[])`
- `{`
- `glutInit (&argc, argv);`
- `glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);`
- `glutInitWindowSize (500,500);`
- `glutInitWindowPosition (100,100);`
- `glutCreateWindow (argv[0]);`

- `glutIdleFunc (mglIdle);`
- `glutDisplayFunc (mglDisplay);`
- `glutReshapeFunc (mglReshape);`
- `glutKeyboardFunc (mglKeyboard);`
- `glutMouseFunc (mglMouse);`

- `mglInit();`
- `glutMainLoop();`
- `return;`
- `}`

Sample

- `void main(int argc, char *argv[])`
- `{`
- `glutInit (&argc, argv);`
- `glutInitDisplayMode (GLUT_DOUBLE GLUT_RGB);`
- `glutInitWindowSize (500,500);`
- `glutInitWindowPosition (100,100);`
- `glutCreateWindow (argv[0]);`

- `glutIdleFunc (mglIdle);`
- `glutDisplayFunc (mglDisplay);`
- `glutReshapeFunc (mglReshape);`
- `glutKeyboardFunc (mglKeyboard);`
- `glutMouseFunc (mglMouse);`

- `mglInit();`
- `glutMainLoop();`
- `return;`
- `}`

Double Buffering, cooperated with `glutSwapBuffers()`.



(We will study complete standalones later.)

Matrix Manipulation

- Open GL has built in support for matrices and matrix manipulation
- 3 matrix stacks
 - GL_MODELVIEW
 - GL_PROJECTION
 - GL_TEXTURE (**covered next week**)
- Open GL uses the matrix on the top of the stack

Common Commands

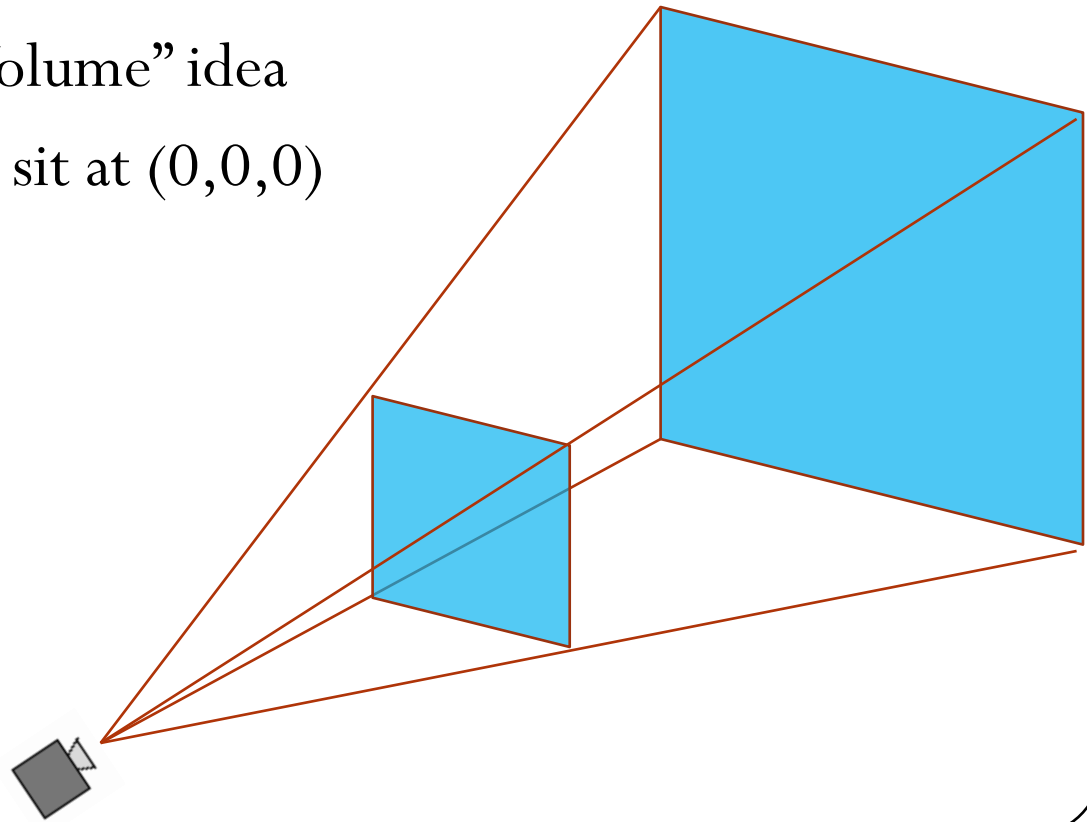
- Modifying matrix state
 - Select matrix stack type with `glMatrixMode(...)`
 - Pushing and Popping on to the stack
 - `glPushMatrix()`, `glPopMatrix()`
 - Loading Matrices
 - `glLoadIdentity()`, `glLoadMatrixf(float *)`
 - Matrix Manipulation
 - `glMultMatrixf(float *)`, `glTranslatef(...)`, `glScalef(...)`

Common Commands

- Modifying matrix state
 - Select matrix stack type with `glMatrixMode(...)`
 - Pushing and Popping on to the stack
 - `glPushMatrix(), glPopMatrix()`
 - Loading Matrices
 - `glLoadIdentity(), glLoadMatrixf(float *)`
 - Matrix Manipulation
 - `glMultMatrixf(float *), glTranslatef(...), glScalef(...)`
- Recall the “Scene Graph” idea

Setting the Projection

- Projection is done by the `GL_PROJECTION` matrix
- The only thing that is left-handed (!?)
- Usually only set it up once.
- Recall the “Viewing Volume” idea
- Assuming the camera sit at $(0,0,0)$
- Helper functions:
 - `glOrtho(...)`
 - `glFrustum(...)`
 - `gluPerspective(...)`



Setting the Viewpoint

- Viewpoint setting is done by the `GL_MODELVIEW` matrix, same as object transformations...**(why?)**
- Load the camera transform for every frame
- Helper functions:
 - `gluLookAt (GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ,
GLdouble centerX, GLdouble centerY, GLdouble centerZ,
GLdouble upX, GLdouble upY, GLdouble upZ)`

Setting the Viewpoint

- Viewpoint setting is done by the `GL_MODELVIEW` matrix, same as object transformations...*(why?)*
- Load the camera transform for every frame
- Helper functions:
 - `gluLookAt (GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ,
GLdouble centerX, GLdouble centerY, GLdouble centerZ,
GLdouble upX, GLdouble upY, GLdouble upZ)`

~ DEMO ~

Object Transformation

- Again, done by the `GL_MODELVIEW` matrix
- Transform the models to the states we want
- Helper functions:
 - `glTranslatef(...)`
 - `glScalef(...)`
 - `glRotatef(...)`

Object Transformation

- Again, done by the `GL_MODELVIEW` matrix
- Transform the models to the states we want
- Helper functions:
 - `glTranslatef(...)`
 - `glScalef(...)`
 - `glRotatef(...)`

~ DEMO ~

Drawing

- Basic primitives can be drawn by the following syntax:

```
glBegin( primitive type)
```

```
glVertex3f(....)
```

```
.....
```

```
glEnd()
```

- To make your life easier, you may consider using GLU's helper functions for more complex primitives (such as quadratic surfaces)
 - *gluCylinder()*
 - *gluDisk()*
 - *gluSphere()*

Basic Primitive Types

- 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

Add Normal, Color and Texture...

- Use following functions
 - `glNormal*(....)`,
 - `glColor*(....)`,
 - `glTexCoords*(....)`
- Issue these commands **before** a vertex command
(always keep in mind you are dealing with a state machine)

Add Normal, Color and Texture...

- Use following functions
 - `glNormal*(....)`,
 - `glColor*(....)`,
 - `glTexCoords*(....)`
- Issue these commands **before** a vertex command
(always keep in mind you are dealing with a state machine)

~ DEMO ~

Example Code

```
#include <stdlib.h>
#include <stdio.h>
#include "GL/glut.h"

int main(int argc, char* argv[])
{
    glutInitDisplayMode(GLUT_DOUBLE |
                       GLUT_RGBA );

    glutInit(&argc, argv);
    glutInitWindowSize(1024, 768);

    glutCreateWindow("OpenGL Banzai");
    glutDisplayFunc(redraw);
    init();
    glutMainLoop();

    return 0;
}
```

```
void init()
{
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    glClearColor( 1.0, 1.0, 1.0, 1.0 );
    glEnable( GL_DEPTH_TEST );
}

void redraw()
{
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    glClear( GL_COLOR_BUFFER_BIT |
            GL_DEPTH_BUFFER_BIT );

    glBegin( GL_QUADS );

    glColor3f( 1.0, 0.0, 0.0 );
    glVertex2f( -0.5, 0.0 );

    glColor3f( 0.0, 1.0, 0.0 );
    glVertex2f( 0.0, 0.5 );

    glColor3f( 0.0, 0.0, 1.0 );
    glVertex2f( 0.5, 0.0 );

    glColor3f( 0.0, 1.0, 0.0 );
    glVertex2f( 0.0, -0.5 );

    glEnd();
    glutSwapBuffers();
}
```

More Example Code...

More Reading

- <http://www.sgi.com/products/software/opengl/examples/index.html>
- <http://www.glprogramming.com/blue/>
- http://www.opengl.org/documentation/red_book/
- <http://www.opengl.org/resources/code/samples/s2001/>
- <http://www.opengl.org/documentation/>
- <http://nehe.gamedev.net/>

Following Week(s)

- Display Lists
- Texturing
- Shadow
- Mirror
- Interaction
- Luxo Jr. Lamp

Questions?