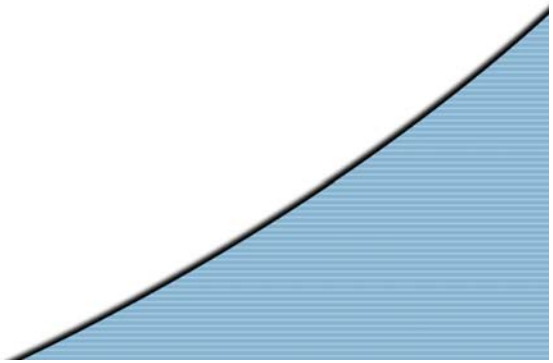
SIGGRAPH 2002, Course 59:

Introduction to OpenGL Programming



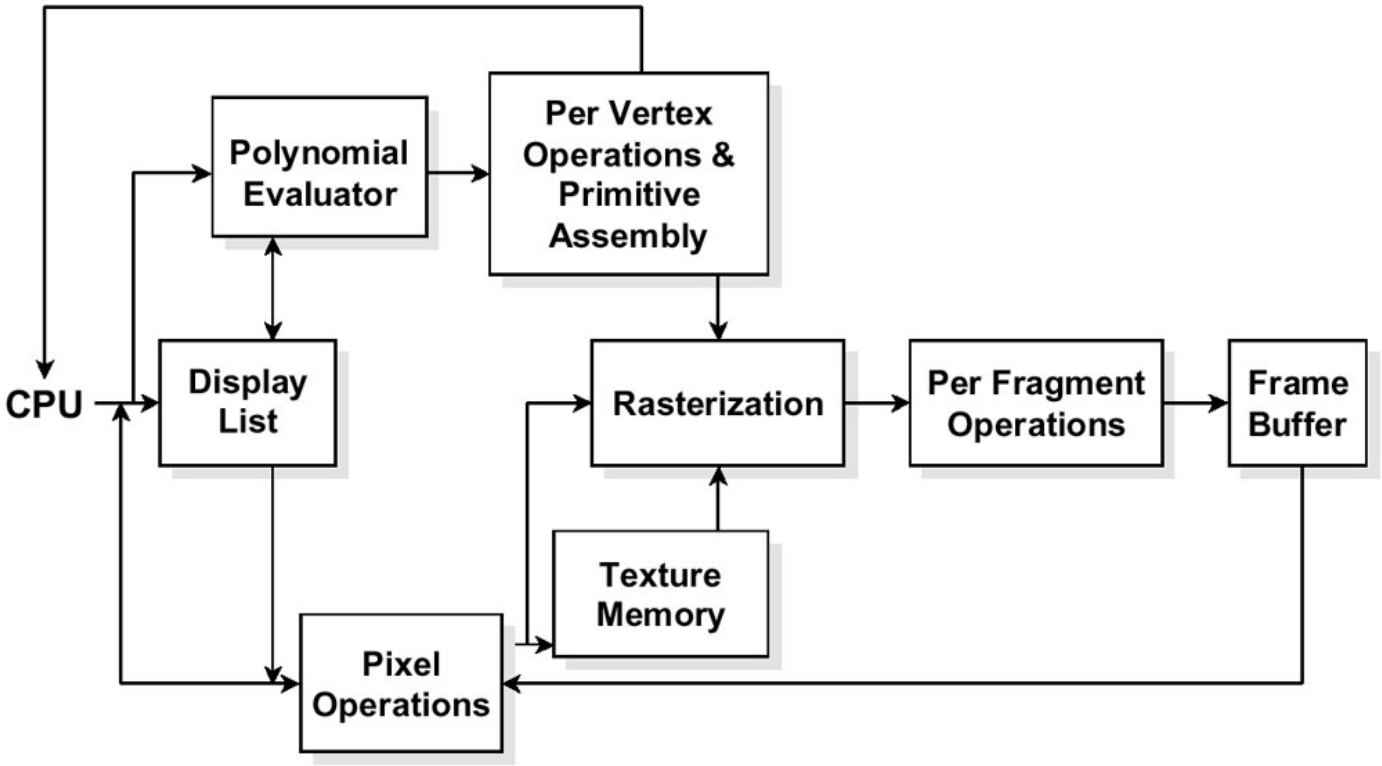
What is OpenGL?

- high-quality color images composed of geometric and image primitives
- window system independent
- operating system independent

OpenGL as a Renderer

- Geometric primitives
  - Points, lines and polygons
- Image primitives
  - Images and bitmaps
  - Separate pipeline for images & geometry
    ...linked through texture mapping
- Rendering depends on current state
  - Colors, materials, light sources, etc.

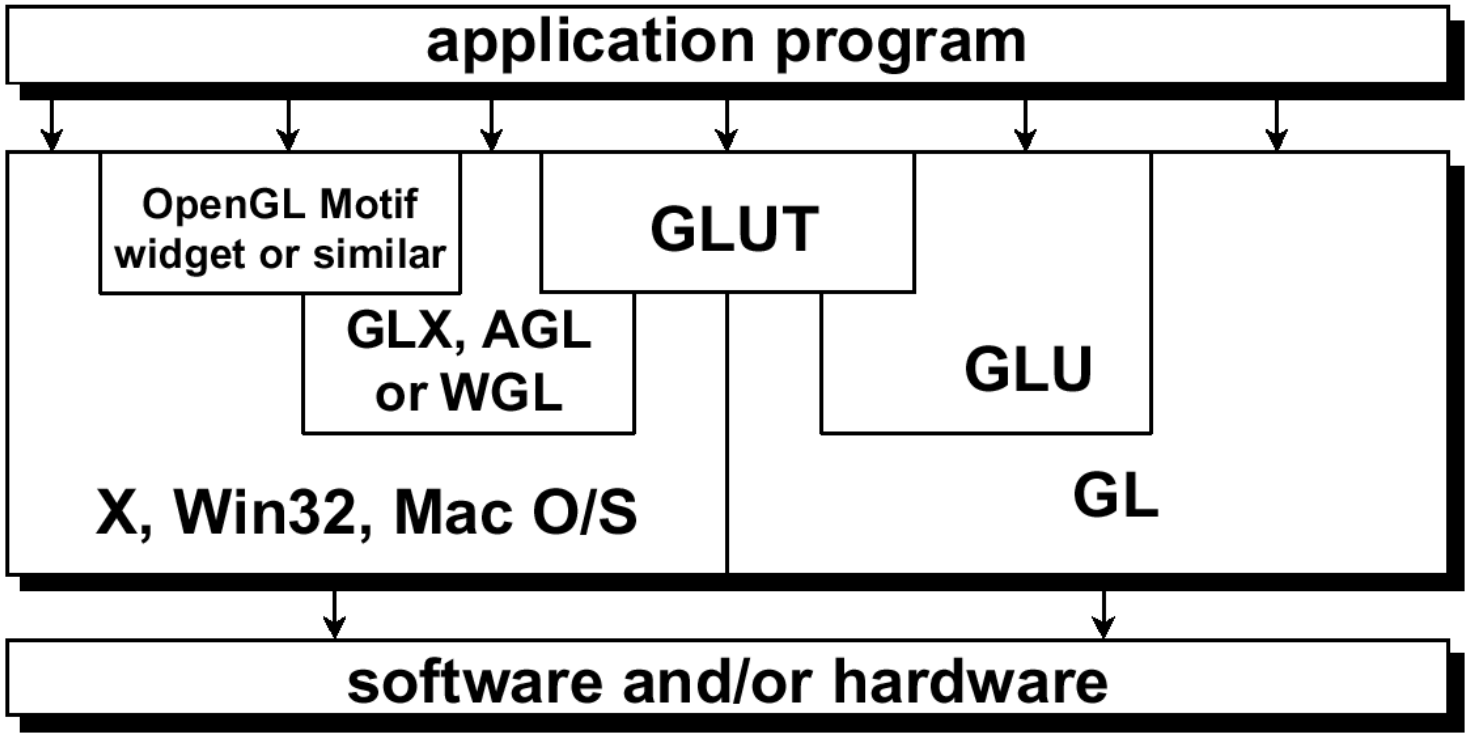
OpenGL Architecture



Related APIs

- AGL, GLX, WGL
  - Glue between OpenGL and windowing system
- GLU (OpenGL Utility Library)
  - Part of OpenGL
  - NURBS, tessellators, quadric shapes, etc.
- GLUT (OpenGL Utility Toolkit)
  - Portable windowing API
  - not offically part of OpenGL

OpenGL and related APIs



Preliminaries

- Header files

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
```

- Libraries
- Enumerated types
  - OpenGL defines numerous types for compatibility (**`Glfloat`**, **`Glint`**, etc.)

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

Sample program

```
void main( int argc, char** argv )
{
    glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE);
    glutCreateWindow(argv[0]);
    init();

    glutDisplayFunc(display);
    glutReshapeFunc(resize);
    glutKeyboardFunc(key);
    glutIdleFunc(idle);

    glutMainLoop();
}
```

OpenGL initialization

- Set up whatever state you are 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 );
}
```

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);
```

Rendering callback

- Do all of your drawing here

```
glutDisplayFunc(render);
```

- Code Example

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

Idle callback

- Use for animation and continuous update

```
glutIdleFunc(idle);
```

- Code example

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

User input callback

- Process user input

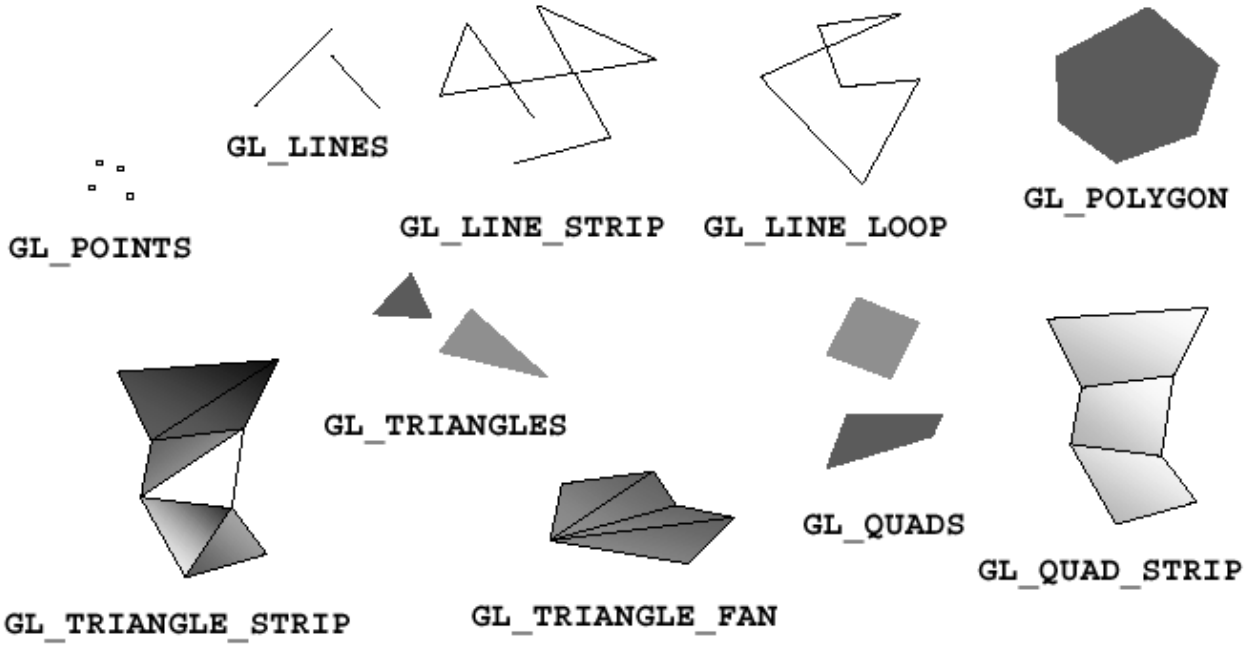
```
glutKeyboardFunc(keyboard);
```

- Code example

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

OpenGL geometric primitives

- All geometric primitives are specified by vertices



Specifying geometric primitives

- Primitives are specified using:

```
glBegin(type);  ...  glEnd();
```

- Code example

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

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)
```