

Graphics APIs

- Provide a software interface to graphics hardware
- Reduce development time and increases software portability
- Examples: OpenGL (www.opengl.org), DirectX (Microsoft)

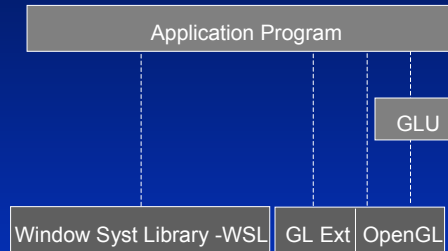
Copyright © Manuel M. Oliveira

OpenGL

- Industry's most widely used/supported 2-D and 3-D graphics API
 - Runs on UNIX, LINUX, Windows 95/98/NT/2000, Mac OS, OS/2, etc.
 - Works with X-Windows, Win32, Mac OS, etc.
 - Bindings for Ada, C, C++, Fortran, Python, Perl and Java
- Truly open, independent of hardware, OS and window system
- Scalable, reliable, portable, easy to use and well documented
- Contains commands for rendering but not for windowing tasks or for handling input events

Copyright © Manuel M. Oliveira

API Hierarchy



GLU - Utility Library: contains routines that use lower-level OpenGL commands to perform tasks such as setting up matrices for viewing and projections, performing polygon tessellation and rendering surfaces (e.g., quadrics, NURBS)

WSL - Xlib (UNIX) and GDU (MS Windows)

GL Ext - OpenGL extensions for the Window System (e.g., GLX (UNIX) and WGL (MS Windows))

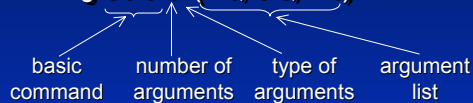
Copyright © Manuel M. Oliveira

OpenGL Command Syntax

- `gl<basic command><# arguments><type argum.>(<argum. list>)`

- Example

– `glColor3f(2.0, 3.5, 1.3)`



- Argument types

suffix	type name	C type name	suffix	type name	C type name
b	GLbyte	signed char	d	GLdouble	double
s	GLshort	short	ub	GLubyte	unsigned char
l	GLint	int	us	GLushort	unsigned short
f	GLfloat	float	ui	GLuint	unsigned int

Copyright © Manuel M. Oliveira

Geometric Model and Vertices

- Geometric models are usually represented using polygonal meshes
- Each polygon is defined by a sequence of vertices
- Examples of vertex specification
 - `glVertex2i (50, 10)` 2-D point defined using integer coordinates
 - `glVertex3d (2.0, 3.5, 1.3)` 3-D point defined using double precision coordinates
 - `glVertex4fv(vtx)` 4-D point defined using an array of floating point coordinates
 - with: `float vtx[4] = {1.5, 4.0, 3.2, 2.0};`

Copyright © Manuel M. Oliveira

Geometric Primitive Types

- Points, Lines and Polygons
 - `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`,
- Complex objects created by grouping primitive types

```
glBegin(<primitive type>);
    <OpenGL geometric commands>;
glEnd();
```
- Commonly used OpenGL geometric commands
 - `glVertex*()`, `glColor*()`, `glNormal*()`, `glTexCoord*()`

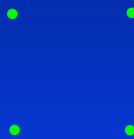
Copyright © Manuel M. Oliveira

Object Rendering

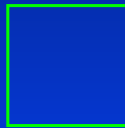
```
glColor3f(0.0, 1.0, 0.0);
glBegin(<OpenGL primitive type>);
glVertex2f(0.0, 0.0);
glVertex2f(0.0, 10.0);
glVertex2f(10.0, 10.0);
glVertex2f(10.0, 0.0);
glEnd();
```

OpenGL is a state machine

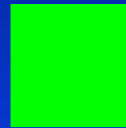
• GL_POINTS



GL_LINE_LOOP

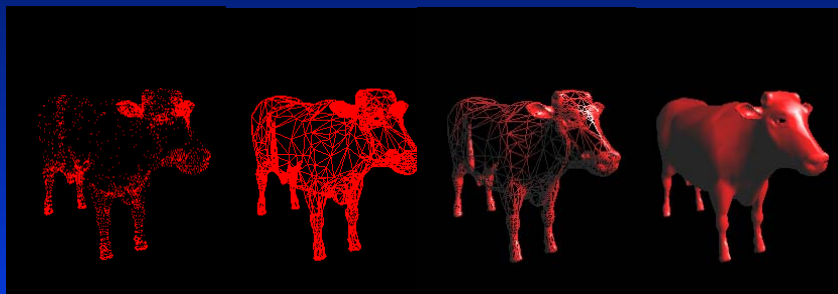


GL_POLYGON



Copyright © Manuel M. Oliveira

Rendering Modes



GL_POINTS

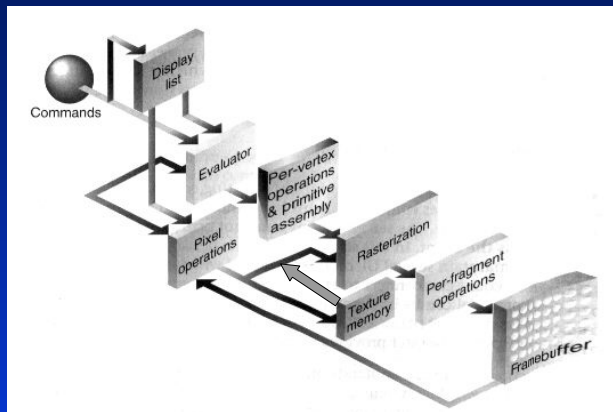
glPolygonMode
(GL_FRONT,
GL_LINE)

glPolygonMode
(GL_FRONT,
GL_LINE)
with lighting

glPolygonMode
(GL_FRONT,
GL_FILL)
with lighting

Copyright © Manuel M. Oliveira

OpenGL Pipeline - Overview

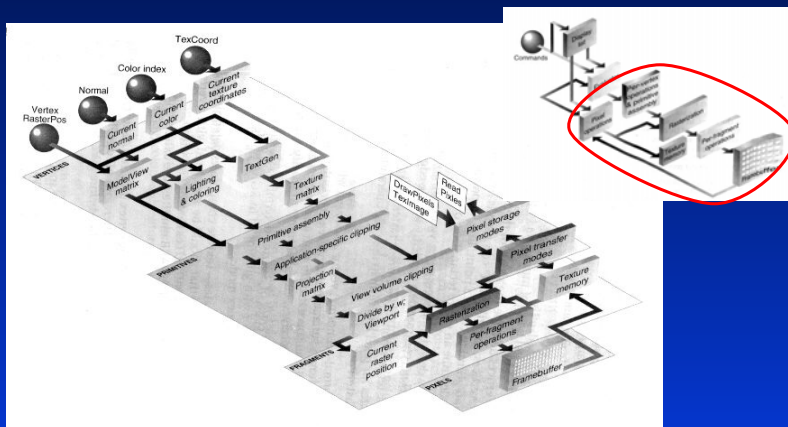


How OpenGL Processes Data

Reproduced from the **OpenGL Reference Manual**, 3rd Ed. OpenGL V1.2 (Addison Wesley 2000)

Copyright © Manuel M. Oliveira

OpenGL Pipeline - More Details

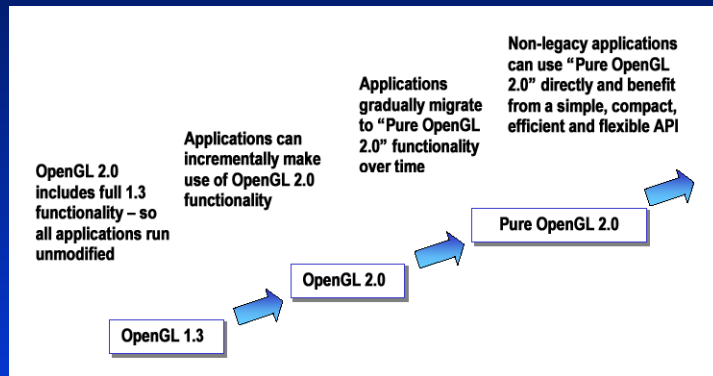


Stages of the OpenGL Processing Pipeline

Reproduced from the **OpenGL Reference Manual**, 3rd Ed. OpenGL V1.2 (Addison Wesley 2000)

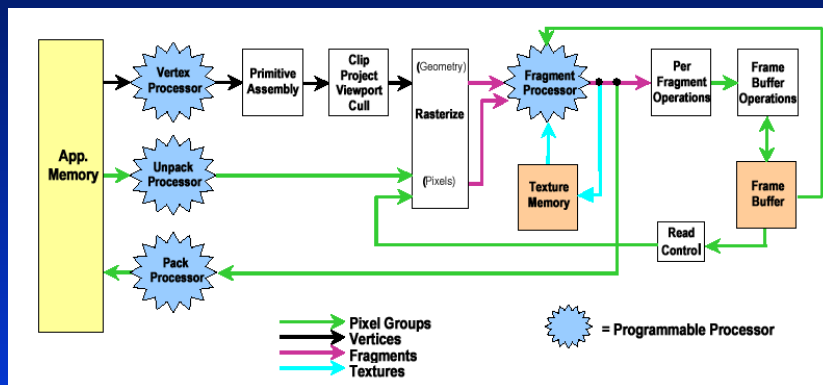
Copyright © Manuel M. Oliveira

OpenGL Transition Path



Copyright © Manuel M. Oliveira

OpenGL 2.0 Logical Diagram



Copyright © Manuel M. Oliveira

GLUT

- OpenGL Utility Toolkit, written by Mark Kilgard
 - Window Management
 - glutInit(), glutInitDisplayMode(), glutInitWindowPosition(), glutInitWindowSize(), glutCreateWindow()
 - Display Callback
 - glutDisplayFunc(), glutPostRedisplay()
 - Running the program
 - glutMainLoop()
 - Handling Input Events
 - glutReshapeFunc(), glutKeyboardFunc(), glutMouseFunc(), glutMotionFunc()
 - Managing a Background Process
 - glutIdleFunc()

Copyright © Manuel M. Oliveira

GLUT Example

- Simple Example

```
...  
int main (int argc, char** argv)  
{  
    glutInit(&argc, argv); // initialize the toolkit  
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);  
    glutInitWindowSize(200, 200);  
    glutInitWindowPosition(50, 50);  
    glutCreateWindow("Hello World");  
    your_init_function();  
    glutDisplayFunc(display_func);  
    glutMainLoop();  
}
```



Copyright © Manuel M. Oliveira