OpenGL: Open Graphics Library
standardized 3D graphics library (API)
available on many platforms, often with hardware support
derivatives of the Silicon Graphics' GL library
all function calls have the 
prefix, e.g.: glscale3fv()

Two GL toolkits:
GLU OpenGL Utility Library
contains reusable code for common objects (such as spheres)
support for MIBIRS surfaces, quadric surfaces, etc.
all function calls have the 
prefix, e.g.: glu
after opening an OpenGL implementation, no separate installation required
typically involves no hardware acceleration

GLUT: OpenGL Utility Toolkit
simplifies window handling and interaction
handles windows system dependencies
OpenGL objects, scene windows, keyboard, mouse events
implies full OpenGL
all function calls have the 
prefix, e.g.: glutMainLoop()

OpenGL Extensions:
allows new hardware innovations to be accessible through OpenGL

The Mesa 3D Graphics Library
http://www.mesa3d.org/
Mesa is a 3-D graphics library with an API which is very similar to that of
OpenGL. To the extent that Mesa utilizes the OpenGL command syntax
or state machine, it is being used with authorization from Silicon
Graphics, Inc. (SGI).
Mesa is OpenGL in all respects except the name.
Distributed under the XFree86 license, Mesa is not a licensed
implementation of OpenGL and has not been tested by the OpenGL
conformance tests.
But all that is just legalese: you can consider Mesa to be OpenGL; you can
use OpenGL documentation, for example.
Mesa Licensing Program
SGI is in the process of modifying its licensing programs to adapt to the
release of the OpenGL Sample Implementation (S.I.) under an open
source license.

Non-Goals
Make graphics programming easy
- OpenGL is a power tool
Integrate digital media and 3D graphics
- This is really hard

Goals for OpenGL
Industry-wide acceptance
Consistent implementations
Innovative implementations
Innovative and differentiated applications
Long life
High quality

Why/why not OpenGL?
+ Supporting O.S.
+ Open (kind of)
+ Free versions (Mesa)
- Can be hard to use in "structured"
programming. State machine. ("OpenGL is the assembler of computer graphics")
The Camera Model

- Position
- Direction (look at)
- View up vector
- Projection Matrix (orthographic, perspective, etc.)

Two matrix stacks

- **GL_PROJECTION**
  - Camera (projection) matrix and window to viewpoint
- **GL_MODELVIEW**
  - Model and View matrix

```plaintext
gLMatrixMode(GL_PROJECTION);
glLoadIdentity();
glTranslatef(0.0, 5.0, 6.0);
glRotatef(45.0, 1.0, 2.0, 3.0);
glTranslatef(-4.0, -5.0, -6.0);
```

- The rule in OpenGL is:
  - The transformation specified most recently is the one applied first.

```
C = T(4.0,5.0,6.0)R(45)T(-4.0,-5.0,-6.0)
qu=Cs
```

Defining primitives in OpenGL

- OpenGL is using a right hand system
- Vertices are defined using `glVertex*()`
- Primitive defining statements all start with `glBegin<primitive_type>;
  - `...` and ends with `gEnd();`

Some primitive types

- `GL_POINTS`
- `GL_LINES`
- `GL_LINESTRIPS`
- `GL_TRIANGLES`
- `GL_QUADS`
- `GL_POLYGON`

OpenGL (glut) Callback Functions

- Used for Input and Interaction
- The user submits a pointer to a function which OpenGL calls when the corresponding event occurs.

```plaintext
glutMouseFunc(mouse_func)
glutReshapeFunc(reshape_func)
glutKeyboardFunc(keyboard_func)
glutIdleFunc(idle_func)
glutDisplayFunc(display_func)
```
Display lists
Immediate mode vs. retained mode
- Command caching
- May improve performance
- Start command list using glNewList(list_id, operation)
- End with glEndList()
- glGenLists(n) can be used to obtain free list IDs
- Use glCallList(id) to invoke the list commands

Output from programs 3 & 6