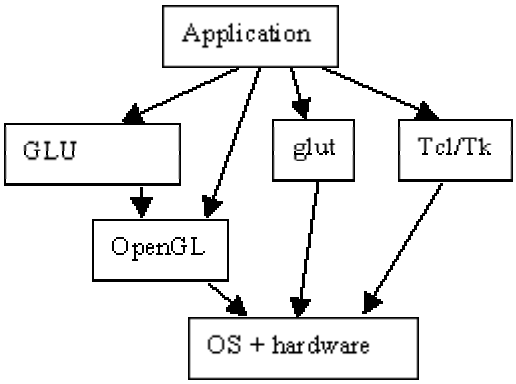
Introduction to the software libraries

This course makes use of several popular libraries to help build sophisticated portable graphics applications with minimal effort.
The following diagram gives an overview of the packages and how they interact. For the purposes of this course, one can think of the GLU and glut libraries as being part of the OpenGL library or the OpenGL *API* (application programmer's interface), eventhough this is not really the case.



OpenGL	-provides a software interface to graphics hardware and implements most of the graphics functionality.
GLU	provides support for some additional operations and primitive types, and is implemented using OpenGL function calls
glut	designed specifically to be used with OpenGL and it takes care of things like opening windows, redraw events, and keyboard and mouse input. It effectively hides all the windowing system dependencies for OpenGL.

OpenGL: Open Graphics Library

- standardized 3D graphics library (API)
- available on many platforms, often with hardware support
- derivative of the Silicon Graphics' GL library
- all function calls have the `gl` prefix, e.g.: `glScale3fv()`
- OpenGL resources on the Internet**

GLU: OpenGL Utility Library

- support for NURBS surfaces, quadric surfaces, surface trimming, ...
- all function calls have the `glu` prefix, e.g.: `gluOrtho2D()`
- comes with all OpenGL implementations, including Windows and Mesa, no separate installation required
- typically involves no hardware acceleration
- `glu32.dll` on Win95/Win98/WinNT, look in same directory as `Opengl32.dll`

glut: OpenGL Utility Toolkit

- hides windowing system dependencies

    - open, refresh, resize window
    - swap front/back drawing buffers for animation
    - trap keyboard, mouse events
- implements pull-down menus
- all function calls have the `glut` prefix, e.g.: `glutMainLoop()`
- versions for X-windows, MacOS, Win95/98, WinNT
- glut resources on the Internet**

2d transforms: OpenGL implementation

- OpenGL is immediate mode: graphics operations are applied 'instantly'
- in terms of transformations, the user gives a rotate, translate, or scale command, and the matrix multiplication represented by that transform is immediately applied to a global transformation matrix.
- In other words, a 4 by 4 matrix of floating point values is maintained. It changes each time a single transformation is done. However, all the individual transformations used to derive these numeric values are not retained.

2d transforms: OpenGL

- OpenGL transformation commands set up a 4 by 4 transformation matrix.
- Can use glGet(GL_MODELVIEW_MATRIX) to retrieve this matrix, and glLoadMatrix to replace it with your own matrix.
- **glRotate{fd} (TYPE angle, TYPE x, TYPE y, TYPE z)**
    - TYPE is f or d
    - rotates the current transformation matrix counterclockwise 'angle' degrees about a ray from the origin through the point (x, y, z)
    - eg. glRotatef(45.0, 0.0, 0.0, 1.0) rotates 45 degrees about the z-axis.
- **glTranslate {fd} (TYPE x, y, z)**
    - translates by the amounts x, y, z
    - note: glTranslatef(): empty argument is the identity matrix
- **glScale {fd} (TYPE x, y, z)**
    - applies x, y, z scaling factors.
    - eg. glScalef(2.0, -0.5, 1.0)