When the object is rendered, the fill or edge colors are interpolated from
vertexes. You can change the current color between calls to
"COLOR". For example, "COLOR" current color assigns a color to all subsequent
previous commands.

The effect of each command is determined by modifications done by
Each command modifies part of the state.

OpenGL API style:
The state also includes 3 matrix stacks.

mapping.
OpenGL state includes 3 matrices: modelview, projection and texture.
Now the MODELVIEW matrix equals I.

LoadIdentity();

// Subsequent matrix ops apply to the MODELVIEW matrix.
ModelView(GL_MODELVIEW);

EC, to make the current modelview matrix be the identity matrix:
stage at each clock step. Stages all operate simultaneously; each command progresses from stage to stage in parallel.

Modern graphics (and CPU) hardware is pipelined: A series of separate modern graphics card hardware

OpenGL's access to its matrices is restricted because OpenGL drives

OpenGL's access to its matrices is restricted because OpenGL

So, it's your job to create modeling and viewing transformations to create your OpenGL graphics, and to combine them into OpenGL in the way OpenGL

... can use it for either modeling or viewing or BOTH OpenGL.

The OpenGL modeling matrix is called the model-view matrix because

Confused?

Read book p.104: "Viewing and modeling transformations are intimately related in OpenGL and are in fact combined into a single modeling matrix."
coordinates.

to given rectangle in the screen or 3D graphics window, expressed in pixel
coordinates within the view volume.

primaries outside the view volume.

coordinates within the view volume to [0, 1] [0, 1] range and clips

(normalizes)

projection when z coordinate is ignored)

projection (produces a perspective or other

(sometimes called viewing coordinates)

standard camera. The numbers now describing the primaries are

Viewing Transformation which moves the world in front of the

Modeling Transformation to produce an instance (in the world).

Viewport Transformation

6. $\text{Viewport} \rightarrow$ Viewing

4. $\text{Projection} \rightarrow$ Viewing

5. $\text{Normalization Translation and Clipping (normalizes)}$

3. $\text{Viewing} \rightarrow$ Modeling

2. $\text{Modeling} \rightarrow$ Viewing

1. Input of Master or Template model

Study HB 6-1 to 6-4, 7-1, 7-2, 7-3; 2 and 3-D Graphics Piplines:

University at Albany Computer Science Dept.
matrix that translates \((0,0,0)\) to \((20.0,30.0,40.0)\).

\[ \mathbf{E}_{\text{TRANSLATE}}(20.0,30.0,40.0) \]

\[ \mathbf{M} \mathbf{W} = \mathbf{M} \cdot \mathbf{L} \]

\[ \mathbf{E}_{\text{MULTIPLY}} \]

\[ \mathbf{E}_{\text{MULTIPLY}} \]

\[ \mathbf{E}_{\text{MULTIPLY}} \]

The current \texttt{OPENGL} \texttt{MODELVIEW} matrix \(\mathbf{M} \mathbf{W} \mathbf{M} \mathbf{V} \mathbf{T}\) transforms coordinates of

University at Albany Computer Science Dept.
(don't worry—it makes it easy to code viewing transformations)

coordinate describe fixed objects.

Think: Transformations change the system by which numeric draws an image of the world as if the camera were centered at point

draw-world()

translate((-100, 0), (100, 0));

2) Using the translation (-100, 100, 0) for viewing:

using a fixed coordinate system.

Think: Transformations move or change objects all described

draw an instance of wheel and bolts at position (20, 30, 40).

draw-wheel-and-bolts();

translate(20, 30, 40, 0, 0);

the redbook

1) Using the (20, 30, 40) translation for modeling (see example 3-4 in
(20, 30, 40).

2. and SECOND, translating (the rotated model) so its origin is at

1. FIRST, rotating the model around x axis by 45°,

draw wheel and bolts AFTER

draw wheel and bolts ()

getrotatet(45, 0, 1, 0, 0, 0, 0)

gettranspose(20, 0, 30, 0, 40, 0);

REMEMBER the OpenGL right matrix multiply rule:
// RESTORE saved MY matrix.
// uses MODELING coordinates.
// draw-model() ... ;
// rotate the model.
// ... rotate( ) ;
// where I want my rotated model to Go.
// ... translate( ) ;
// SAVE current MY matrix.
// pushModelMatrix( ) ;

pushModelMatrix( ) ;

// primitive, outline models, or your own model-drawing functions:
FIRST. Initialize the modelview matrix with the VIEWING
SECOND. Use the handy viewFrustum() ;
transform to apply to your temporary transformation when
you temporarily want a MODELING transformation to apply to
you own model.

// display:

// draw-model() ... ;
// draw-model() ... ;
// draw-model() ... ;

// draw-model() ... ;
3. Put the moving robot on the planet of the RedBook Ch. 3 solar system.

animate control, your choice.

2. Make those fingers move independently (via keyboard, mouse or

1. Add fingers (at least two) to the RedBook Ch. 3 robot.

Project 3:

dot/cross products.

HW 3: 2-d rectangular transformations practice with angles in space, and

(And Modelling)

readings: finish Ch. 5, start Ch. 6 and 7, read Redbook Ch. 3 (viewing

Current assignments: