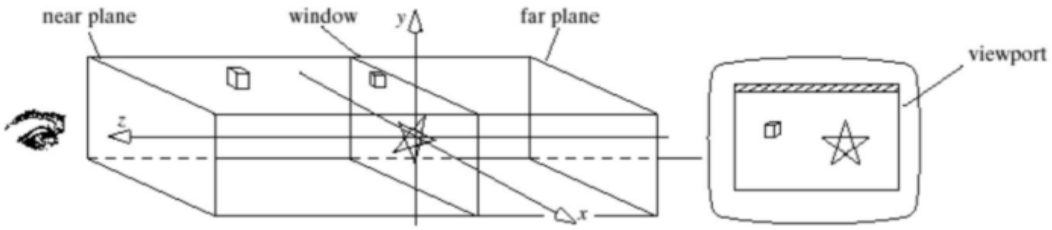
DRAWING 3D SCENE INTERACTIVELY

Drawing 3D Scenes in OpenGL

- We want to transform objects in order to orient and position them as desired in a 3D scene.
- OpenGL provides the necessary functions to build and use the required matrices.
- The matrix stacks maintained by OpenGL make it easy to set up a transformation for one object, and then return to a previous transformation, in preparation for transforming another object.

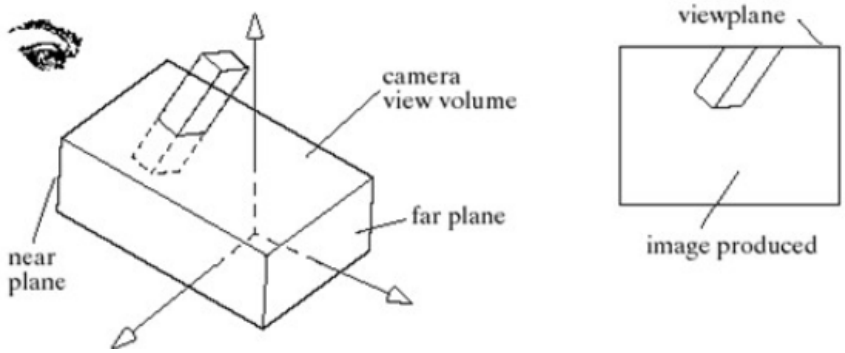
Graphics Pipeline

- The 2D drawing so far is a special case of 3D viewing, based on a simple parallel projection.
- The eye is looking along the z-axis at the world window, a rectangle in the *xy*-plane.

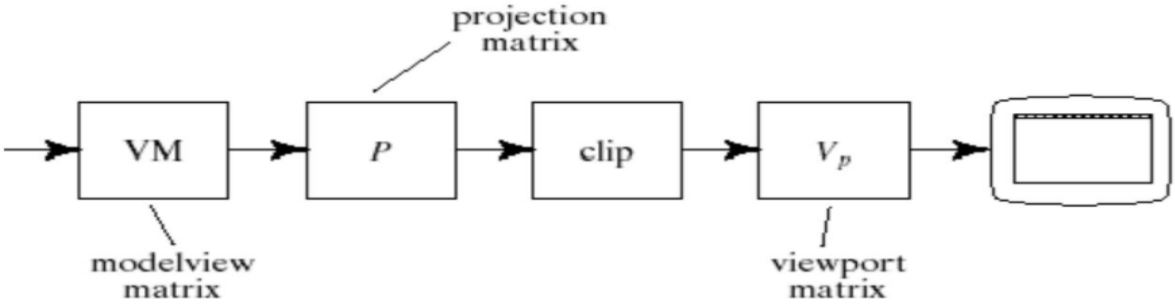


- *Eye* is simply a point in 3D space.
- The "orientation" of the eye ensures that the view volume is in front of the eye.
- Objects closer than *near* or farther than *far* are too blurred to see.

- In 3D, the only change we make is to allow the camera (eye) to have a more general position and orientation in the scene in order to produce better views of the scene.



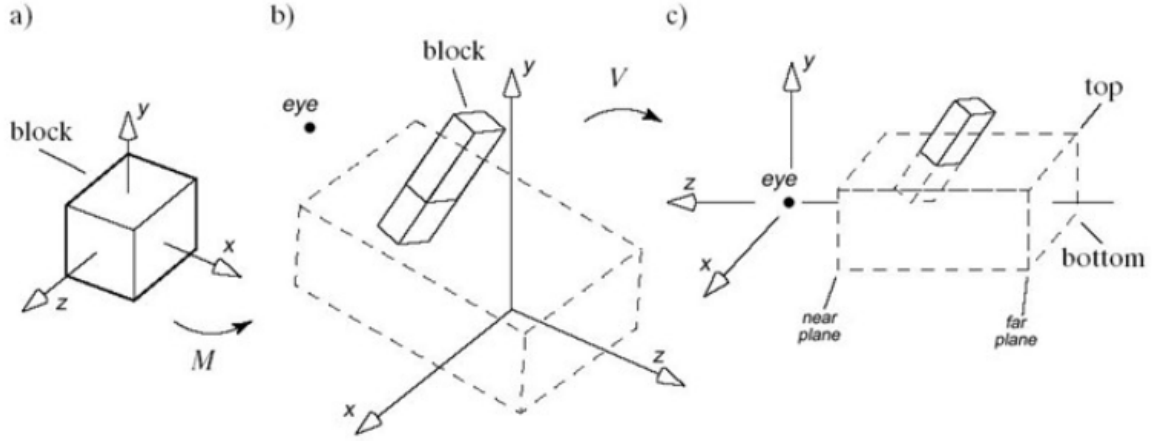
- OpenGL provides functions for defining the view volume and its position in the scene, using matrices in the graphics pipeline.



  - Each vertex of an object is passed through this pipeline using glVertex3d(x, y, z).
  - The vertex is multiplied by the various matrices, clipped if necessary, and if it survives, it is mapped onto the viewport.
  - Each vertex encounters three matrices:
    - The **modelview matrix;**
    - The **projection matrix;**
    - The **viewport matrix;**

The Modelview Matrix (M)

- A modeling transformation *M* scales, rotates, and translates the cube into the block.



The Projection Matrix

- The **projection matrix** scales and translates each vertex so that those inside the view volume will be inside a *standard cube* that extends from -1 to 1 in each dimension (Normalized Device Coordinates).

- Setting the Projection Matrix:
  - glMatrixMode(GL_PROJECTION);
  - glLoadIdentity (); // initialize projection matrix
  - glOrtho (left, right, bottom, top, near, far); // sets the view volume parellelpiped. (All arguments are glDouble ≥ 0.0.)
- left ≤ vv.x ≤ right, bottom ≤ vv.y ≤ top, and -near ≤ vv.z ≤ -far (camera at the origin looking along -z).

The Viewport Matrix

- The **viewport matrix** maps the standard cube into a 3D viewport whose *x* and *y* values extend across the viewport (in screen coordinates), and whose *z*-component extends from 0 to 1 (a measure of the depth of each point).
- This measure of depth makes hidden surface removal (do not draw surfaces hidden by objects closer to the eye) particularly efficient.