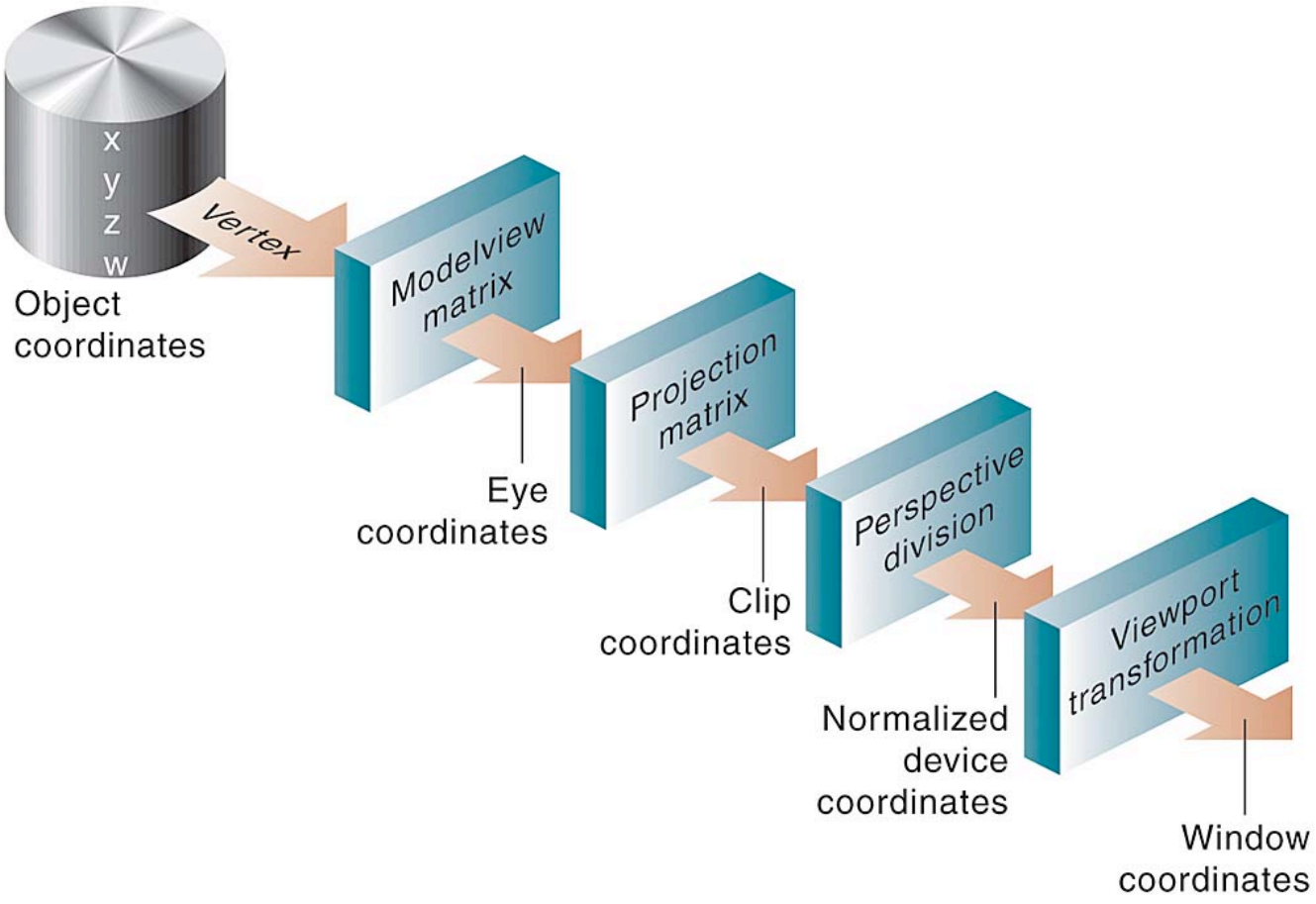
# Today's Agenda

> Intro to OpenGL programming (Java/Processing)

Vertex transformation
The ModelView matrix
The Projection matrix
Drawing
Texturing
Using OpenGL in Processing
 via Processing wrappers
 using directly calls (via the Java bindings)

# Vertex Transformation



# Vertex Transformation

The Modelview matrix encodes information about the camera and the current position of your "cursor" in space.

This "cursor" includes position information as well as orientation and zooming.

After multiplying a point by the Modelview matrix, your point is in "eye" coordinates. That is, in relation to the position of the camera.

The point is then multiplied by the Projection matrix, which puts the point in "clip" coordinates. This means that your point has been projected into a space defined by the properties of your "lens": including the depth of field, the aspect ration, and the field of view of camera. The Projection Matrix "clips" the infinite space of your 3D space into a finite box which contain what will actually be seen on the screen.

This coordinates of this box are then "normalized" into the range -1 through +1 along each axis. This is called the "device coordinates."

Finally, these device coordinates are stretched to fit the screen or window pixels. Information about the z-axis is discarded. Although you can retrieve it if you need it for such things as selecting objects, and also it can be used for things like determining transparency (if blending is on), for culling algorithms, for making shadows, etc.

# Vertex Transformation

The Object or Local coordinate system is defined in terms of the Geometry itself. The origin is usually the center or the lower-left of the object.

The Model or World coordinate system defines the x, y, and z axes which serve as a basis for the 3D space. Where is the origin? Which way is up?

The Eye, Camera, or View coordinate system defines another set of x, y, and z axes which server as a different basis for the 3D space. The camera is always positioned at the origin of this coordinate system.

The Clip coordinate system describes the bounded view of the visible by the camera in terms of both the “lens” of the camera, its “depth of focus”, and the aspect ratio of the screen bounds.

The  Normalized Device coordinates is the same view normalized from -1 to +1 along each axis.

The Window coordinates are these x and y coordinates positioned within the screen bounds. The z is used for depth-testing and is bound between 0 and 1.

# Vertex Transformation

To transform our 3D point from object coordinates into 2D window coordinates we do the following operations:

Given a vertex $\mathbf{v}_o$ in object coordinates $(x_o, y_o, z_o, w_o)$, where $w_o$ is always 1.

Put the object point into eye coordinates by multiplying it by the MODELVIEW matrix **M** (which concatenates the transformation from object coordinates → world coordinates → eye coordinates)…

$$\mathbf{v}_e = \mathbf{M}\mathbf{v}_o$$

Put the vertex into clip coordinates by multiplying it by the PROJECTION matrix P

$$\mathbf{v}_c = \mathbf{P}\mathbf{v}_e$$

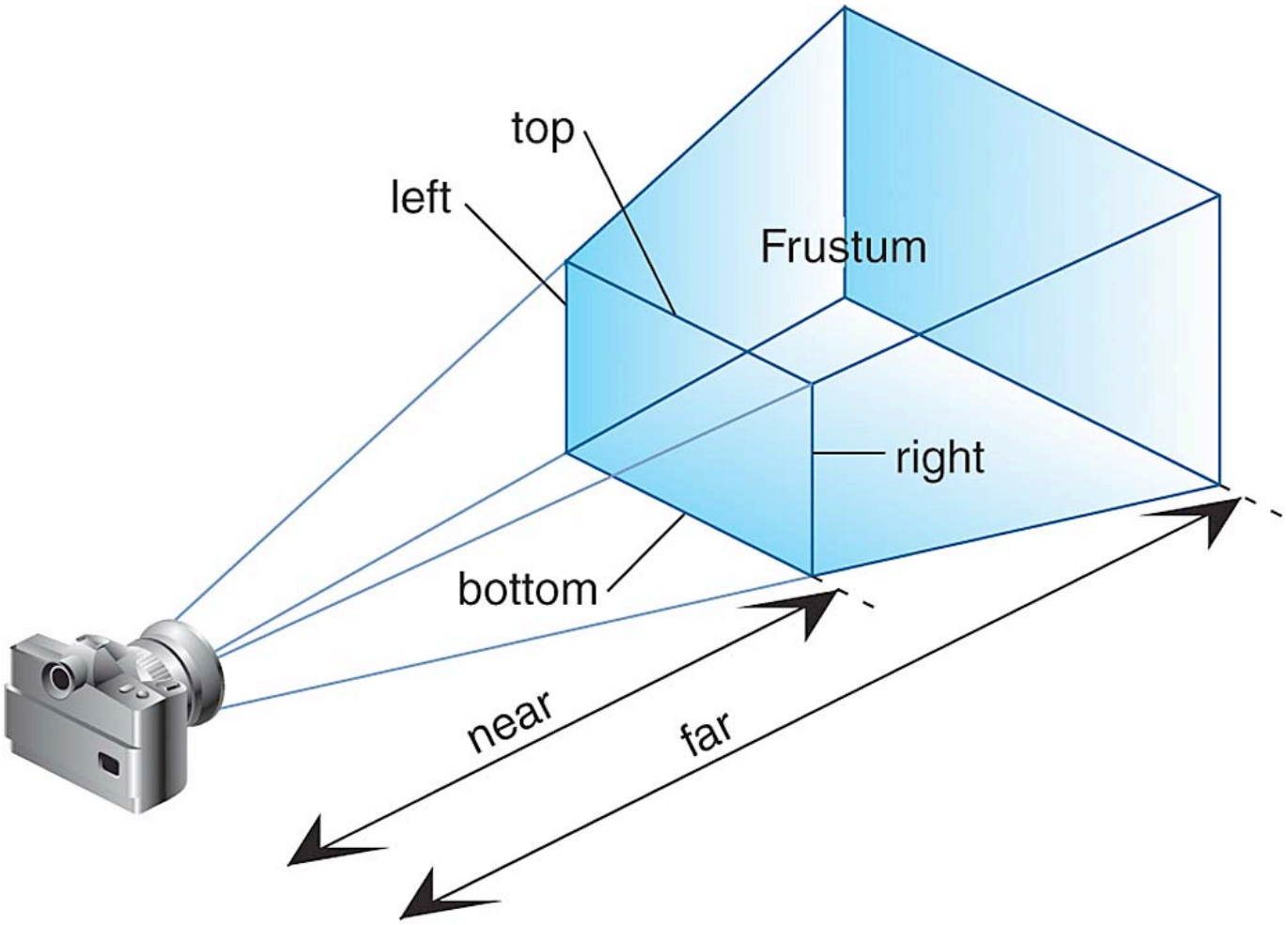
Put the vertex into normalized device coordinates by dividing by the $w_c$ value of $\mathbf{v}_c$.

$$\mathbf{v}_d = (x_c / w_c, y_c / w_c, z_c / w_c)$$

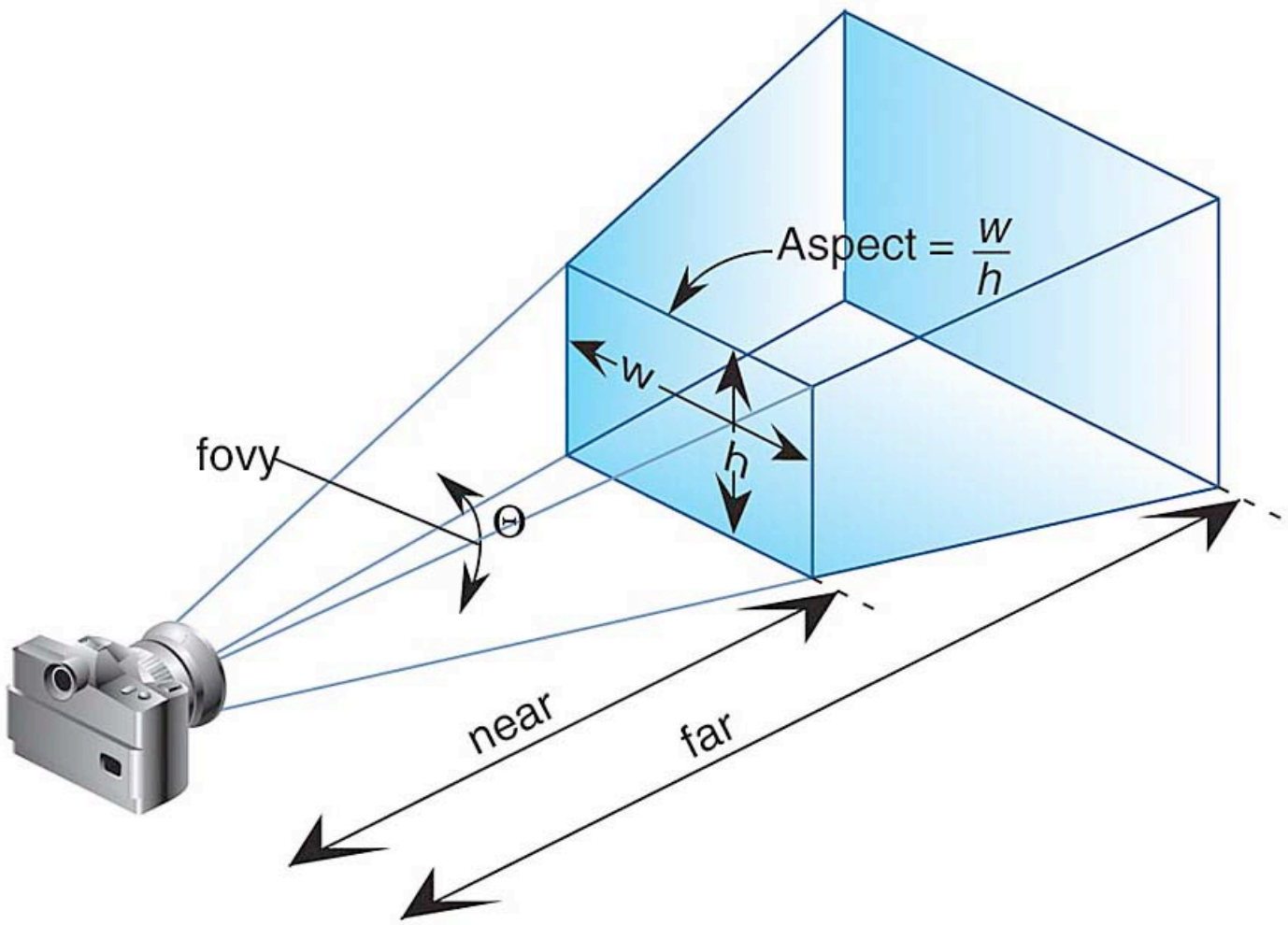
Put the vertex into screen space by scaling $x_c$ and $y_c$ by the width and height of the screen.

$$\mathbf{v}_p = (x_d * \text{width}/2, y_d * \text{height}/2)$$

# Clip Coordinates



# Clip Coordinates



# Code...