Xi'an Jiaotong-Liverpool University
西交利物浦大学

CPT205 Computer Graphics

Viewing and Projection

Lecture 05
2022-23

Yong Yue

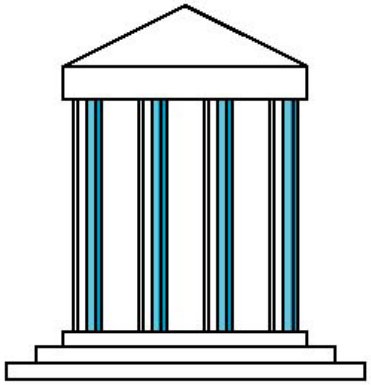
Topics for today

- Concepts of viewing and projection
- Types of projection and their advantages / disadvantages
- Co-ordinate parameters for 3D viewing
- Orthogonal projection
- Perspective projection / Frustum projection
- OpenGL functions
- Sample code

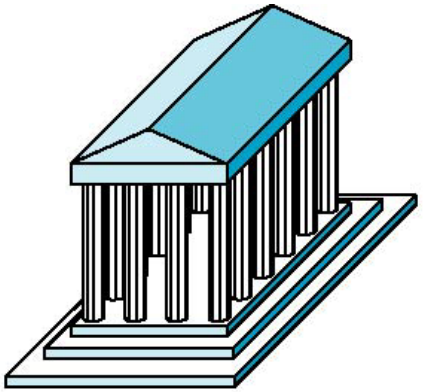
Classic viewing

- Viewing requires three basic elements
  - One or more objects
  - A viewer with a projection surface
  - Projectors that go from the object(s) to the projection surface
- Classical views are based on the relationship among these elements
  - The viewer picks up the object and orients the object in a way that it is to be seen.
- Each object is constructed from flat *principal faces*
  - Buildings, polyhedra, manufactured objects, etc.

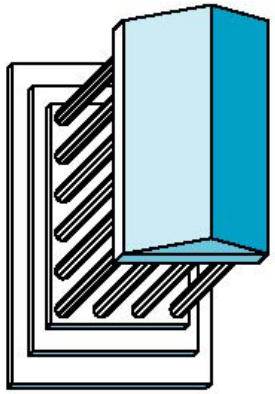
Classic projection



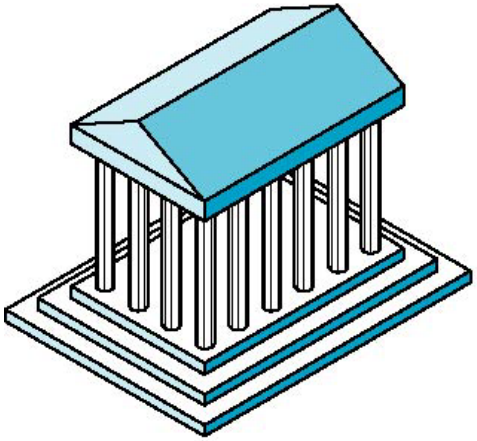
Front elevation



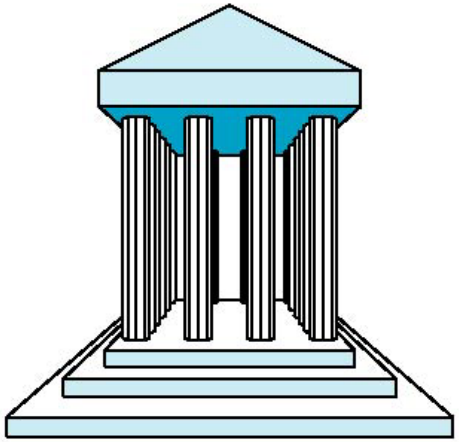
Elevation oblique



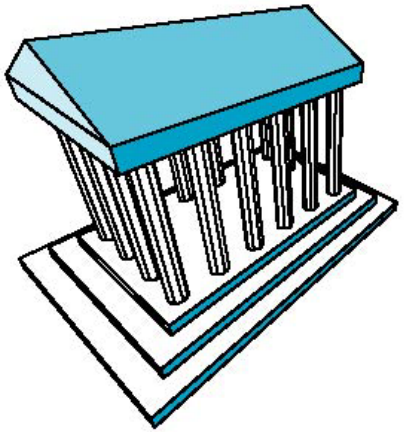
Plan oblique



Isometric



One-point perspective

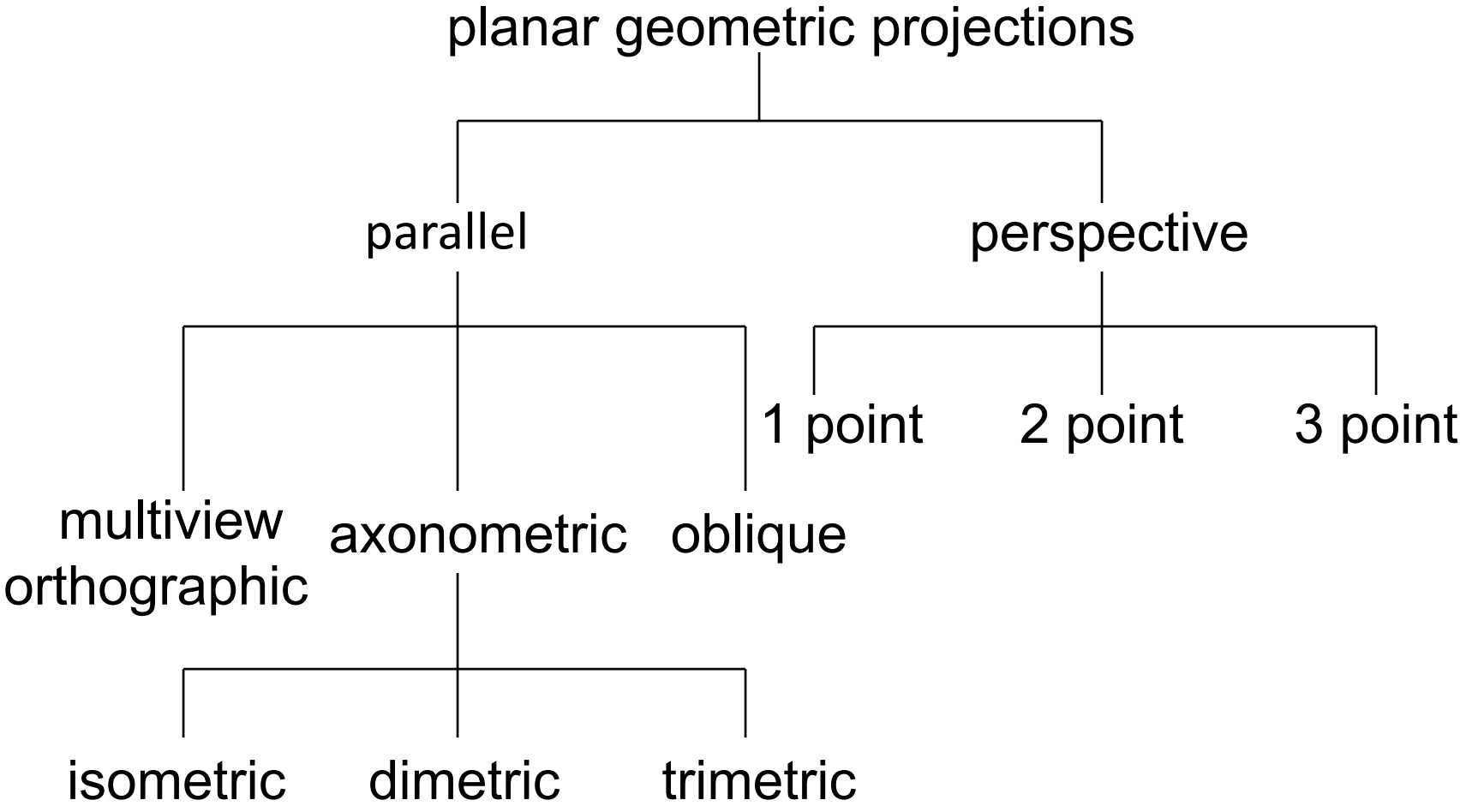


Three-point perspective

Planar geometric projection

- Standard projections project onto a plane.
- Projectors are lines that either converge at a centre of projection or are parallel.
- Such projections preserve lines but not necessarily angles.
- Non-planar projections are needed for applications such as map construction.

Taxonomy of planar geometric projection



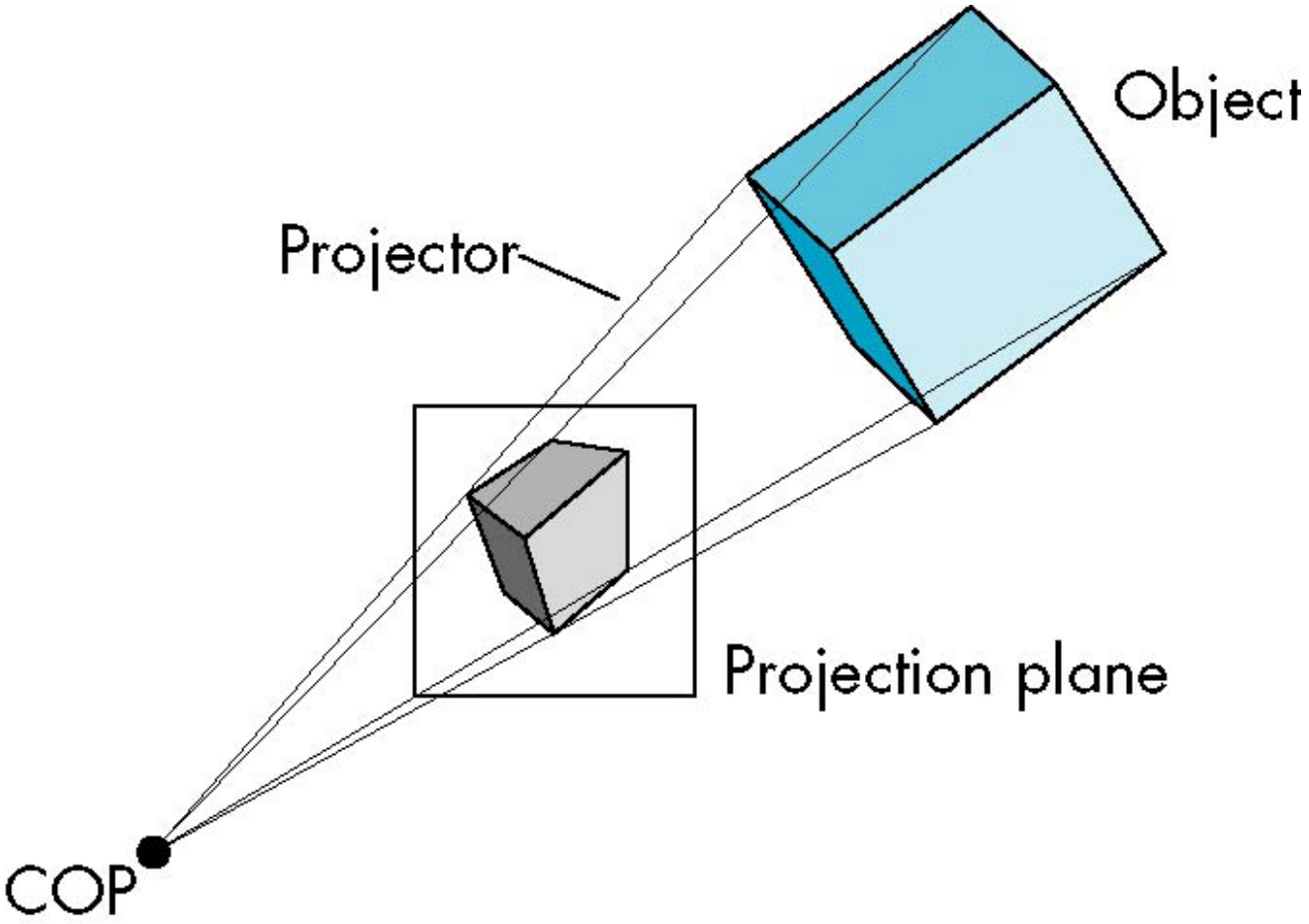
Perspective vs parallel

- Computer graphics treats all projections in the same way and implements them with a single pipeline.
- Classical viewing has developed different techniques for drawing each type of projection.
- The fundamental distinction is between parallel and perspective viewing even though mathematically parallel viewing is the limit of perspective viewing.

Perspective projection

- Perspective projection generates a view of 3-dimensional scene by projecting points to the view plane along converging paths, causing the objects farther from the viewing position to be displayed smaller than the objects of the same size that are nearer to the viewing position.
- A scene generated using perspective projection appears more realistic since this is the way that human eyes and cameras form images.

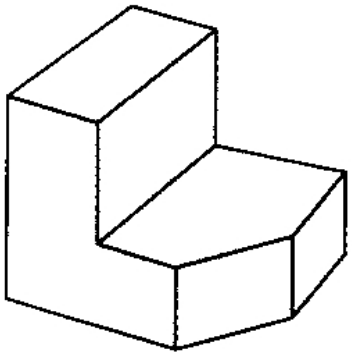
Perspective projection

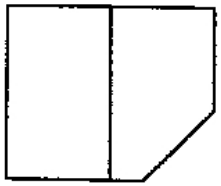


(Centre Of Projection)

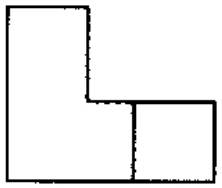
Parallel projection

- This method projects points on the object surface along parallel lines.
- It is usually used in engineering and architecture drawings to represent an object with a set of views showing accurate dimensions.

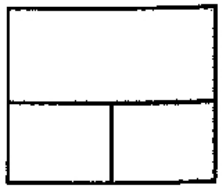




Top

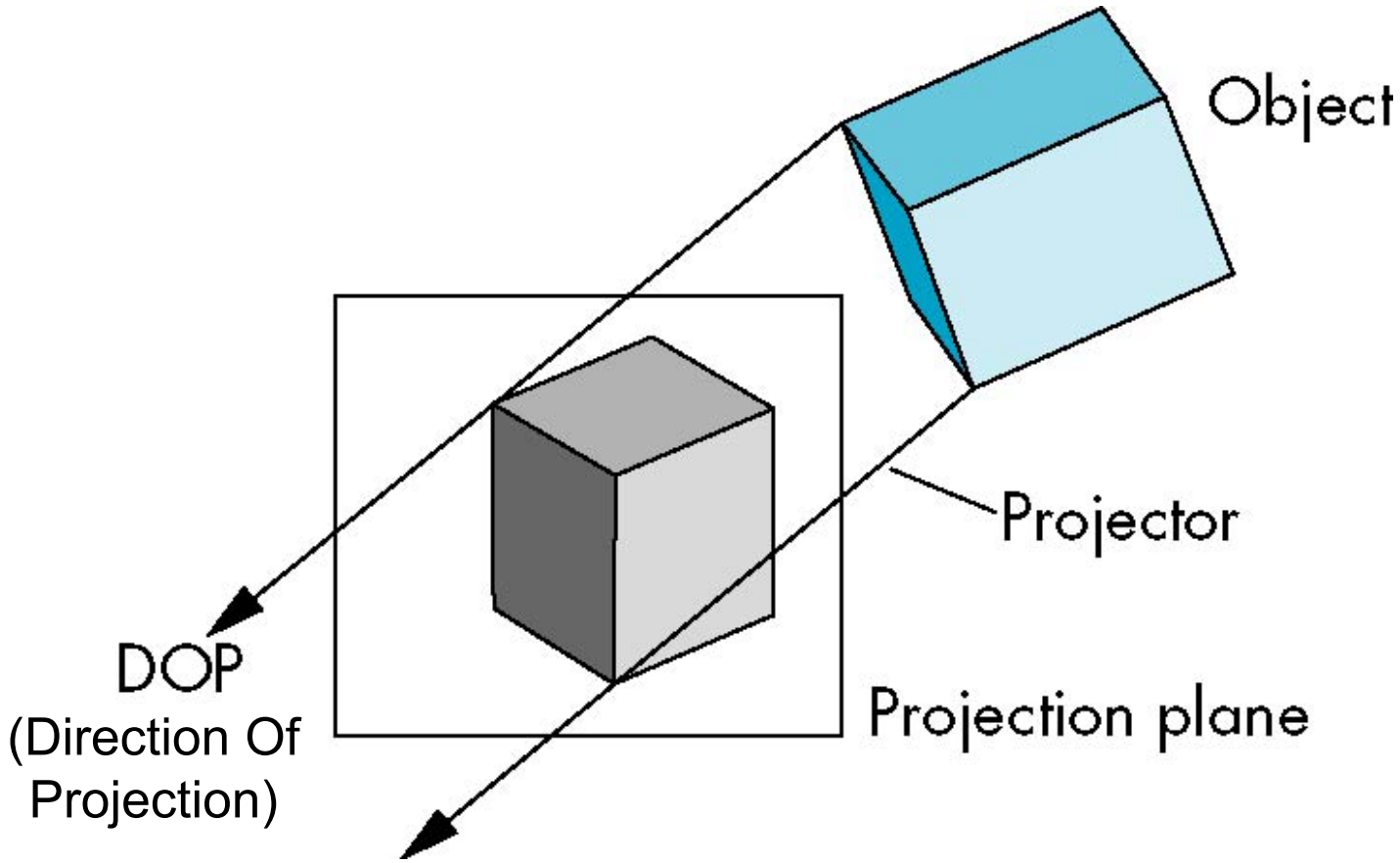


Side



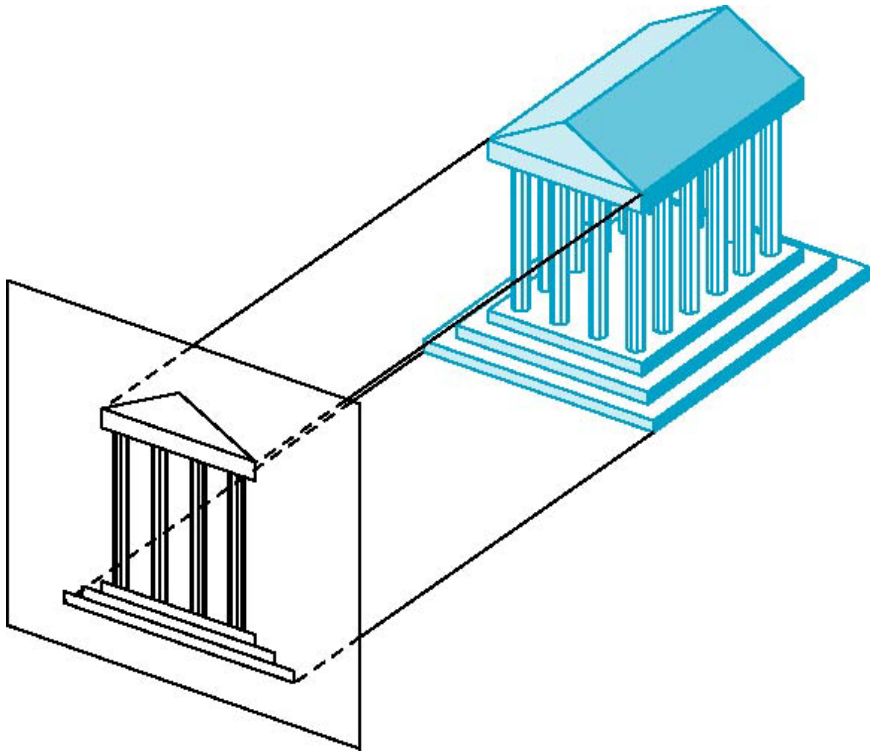
Front

Parallel projection



Orthographic projection

The projectors are orthogonal to projection surface.



Multiview orthographic projection

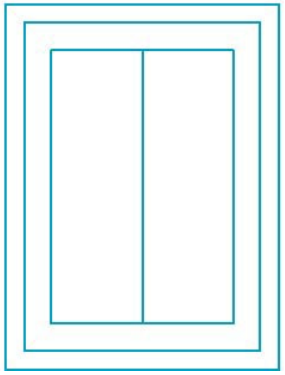
- The projection plane is parallel to the principal face.
- Usually form front, top and side views.

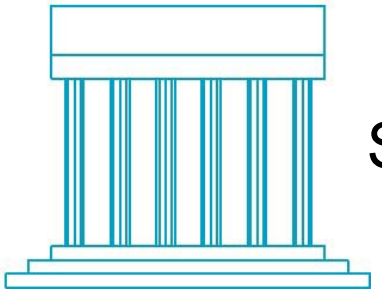
Isometric (not multiview orthographic view)

Front

In CAD and architecture, we often display three multiviews plus isometric

Top





Side

Advantages and disadvantages

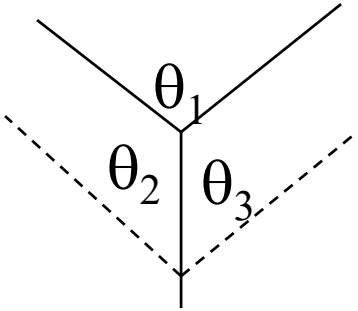
- Preserves both distances and angles
  - Shapes preserved
  - Can be used for measurements
    - Building plans
    - Manuals
- Cannot see what object really looks like because many surfaces are hidden from the view
  - Often the isometric view is added

Axonometric projections

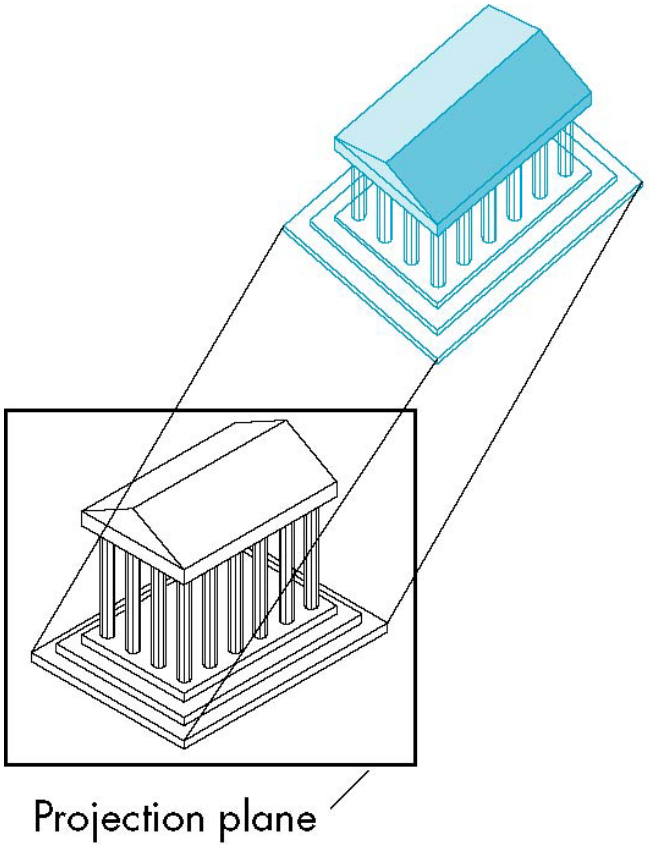
Allow projection plane to move relative to the object.

Classified by the number of angles of a corner of a projected cube

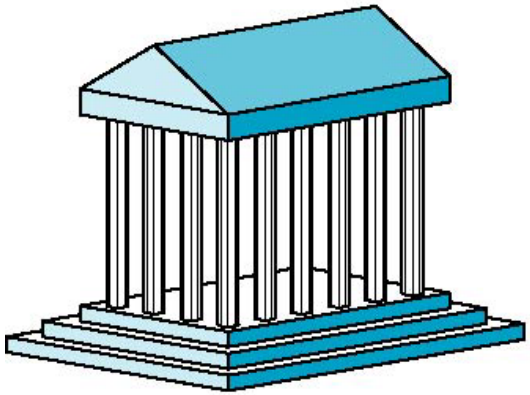
One: trimetric
Two: dimetric
Three: isometric



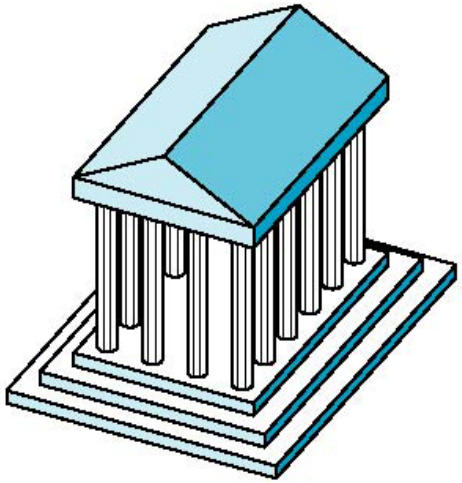
<u>Refer the textbook for more detail.</u>



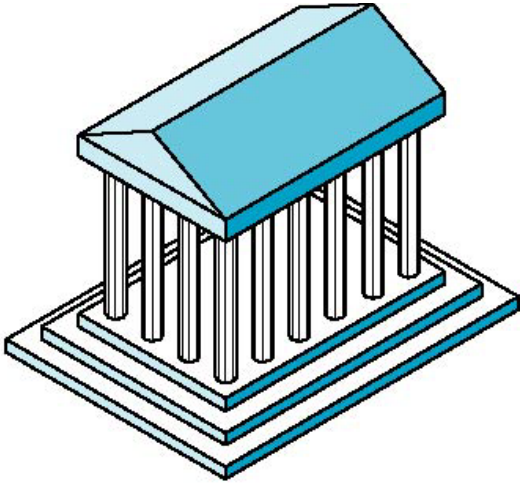
Axonometric projections



Dimetric



Trimetric



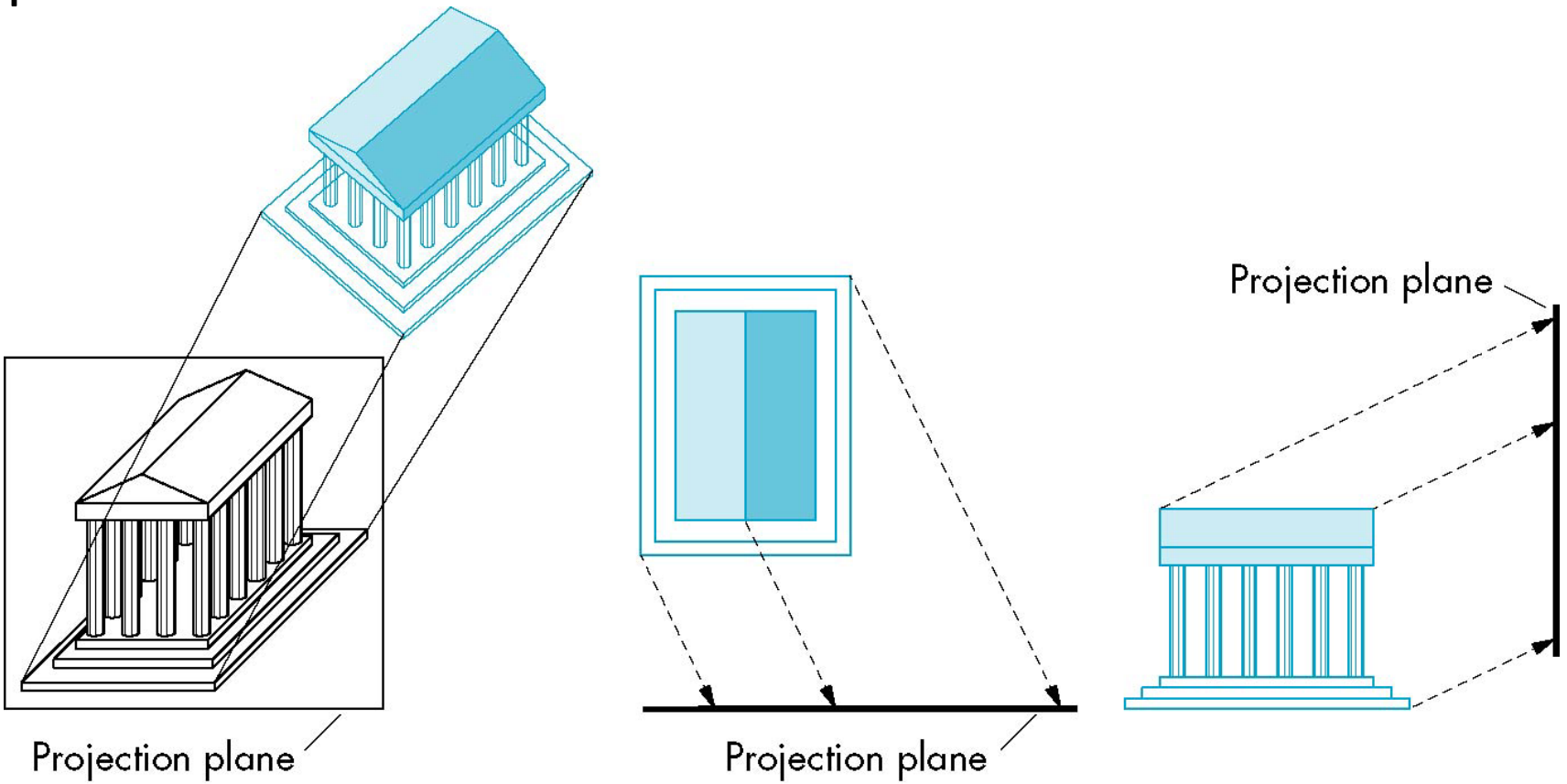
Isometric

Advantages and disadvantages

- Lines are scaled (*foreshortened*) but can find scaling factors
- Lines preserved but angles are not
  - Projection of a circle in a plane not parallel to the projection plane is an ellipse
- Can see three principal faces of a box-like object
- Some optical illusions possible
  - Parallel lines appear to diverge
- Does not look real because far objects are scaled the same as near objects
- Used in CAD applications

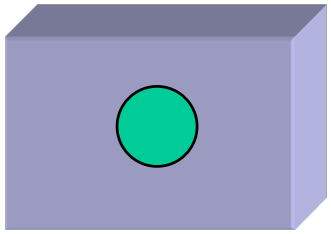
Oblique projection

Arbitrary relationship between projectors and projection plane.



Advantages and disadvantages

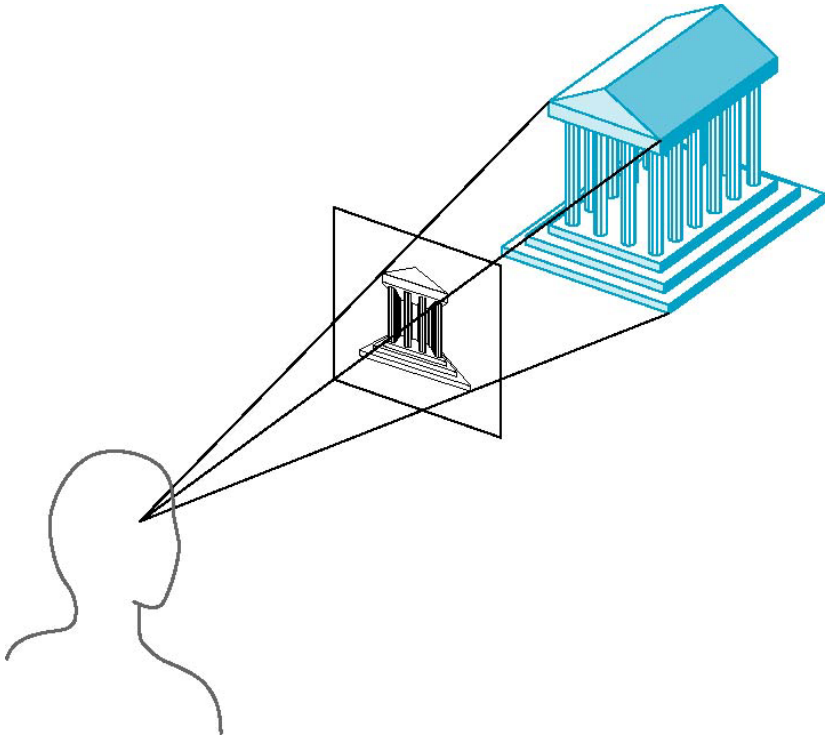
- Can pick the angles to emphasise a particular face
  - Architecture: plan oblique, elevation oblique
- Angles in faces parallel to the projection plane are preserved while we can still see “around” side.



- In the physical world, we cannot create oblique projections with a simple camera; possible with bellows camera or special lens (architectural).

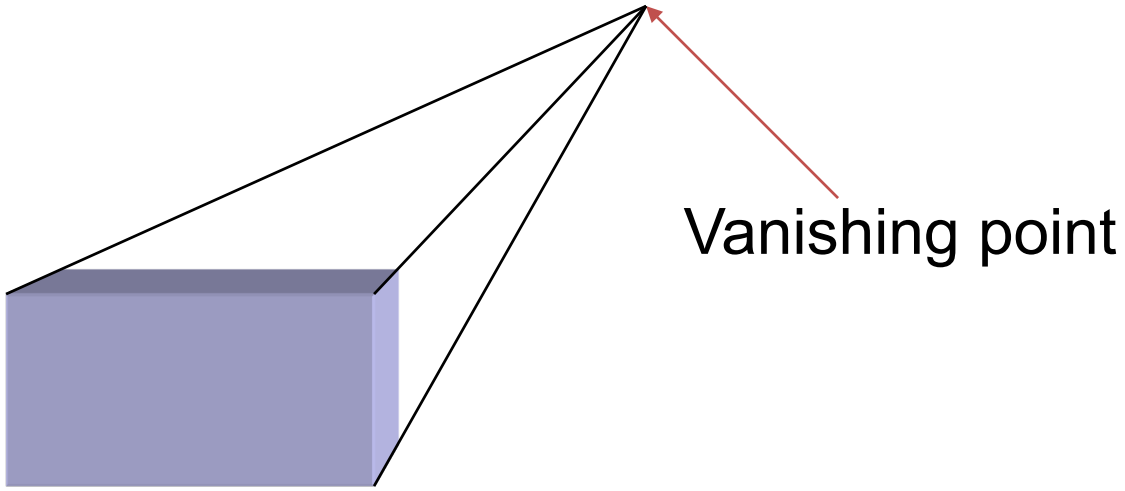
Perspective projection

Projectors converge at the centre of projection (CoP).



Vanishing points

- Parallel lines (not those parallel to the projection plane) on the object converge at a single point in the projection (the *vanishing point*).
- Drawing simple perspectives by hand uses the vanishing point(s).



Three-point perspective

- No principal face parallel to the projection plane
- Three vanishing points for the cube

Two-point perspective

- One principal direction parallel to the projection plane
- Two vanishing points for the cube

One-point perspective

- One principal face parallel to the projection plane
- One vanishing point for the cube

Advantages and disadvantages

- Objects further from the viewer are projected smaller than the same sized objects closer to the viewer (*diminution*)
  - Looks realistic
- Equal distances along a line are not projected into equal distances (*non-uniform foreshortening*).
- Angles preserved only in planes parallel to the projection plane.
- More difficult to construct by hand than parallel projections (but not more difficult by computer).

Computer viewing and projection

There are three aspects of the viewing process, all of which are implemented in the pipeline:

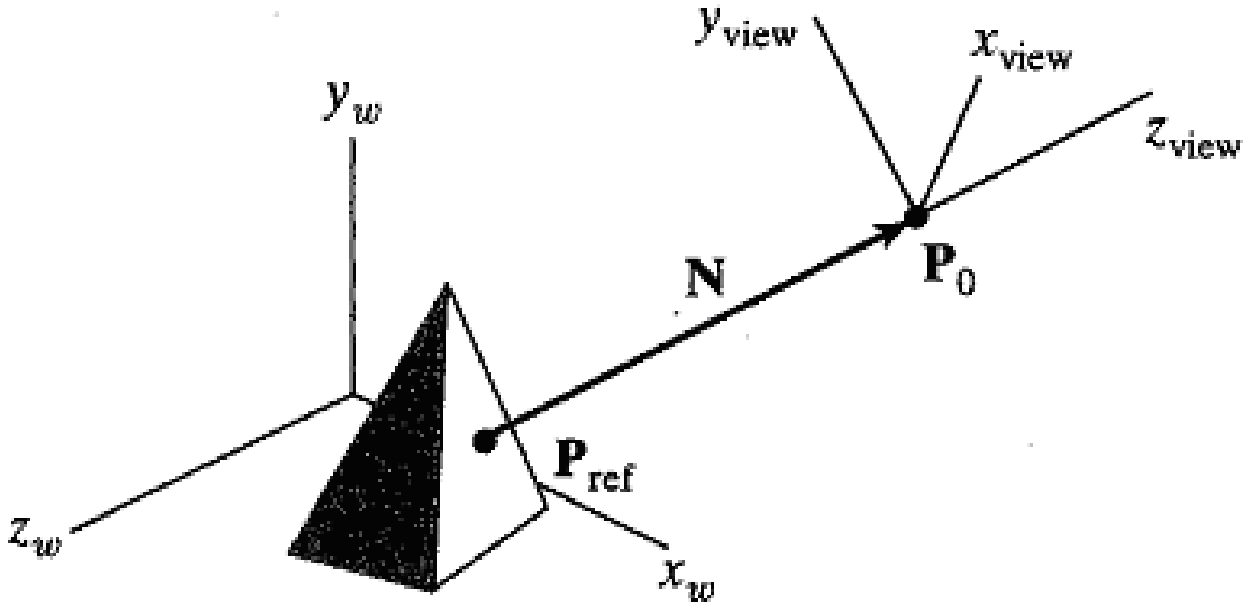
- Positioning the camera
  Setting the model-view matrix
- Selecting a lens
  Setting the projection matrix
- Clipping
  Setting the view volume

3D viewing co-ordinate parameters (1)

- A world co-ordinate position $P_0(x_0, y_0, z_0)$ is selected as the viewing origin (called view point, viewing position, eye position or camera position).
- The view plane can then be defined with a normal vector, which is the viewing direction, usually the negative z_{view} axis.
- The viewing plane is thus parallel to the x_{view}-y_{view} plane.

3D viewing co-ordinate parameters (2)

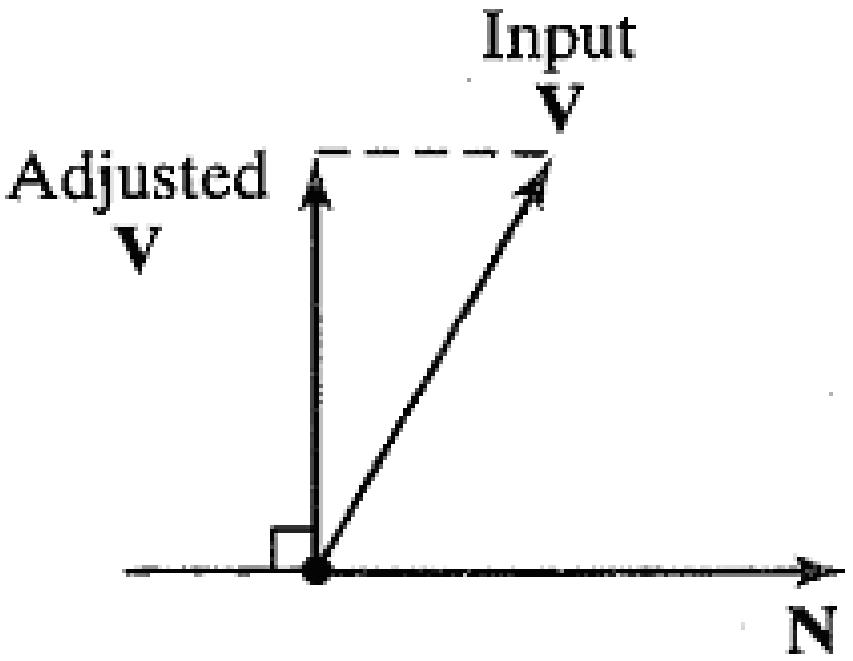
- The direction from a reference point to the viewing point can be taken as the viewing direction (vector), and the reference point (x_{ref}, y_{ref}, z_{ref}) is termed the look-at point.



3D viewing co-ordinate parameters (3)

- Once the viewing plane normal vector **N** is defined, the direction for view-up vector **V** can be set to establish the positive x_{view} axis. Since **N** defines the direction for z_{view}, **V** must be perpendicular to **N** (i.e. parallel to the x_{view}-y_{view} plane).
- But it is generally difficult to determine **V** precisely, and viewing routines typically adjust the user defined value of **V** (e.g. projecting it onto a plane perpendicular to **N**).
- Any direction can be used for **V** as long as it is not parallel to **N**.

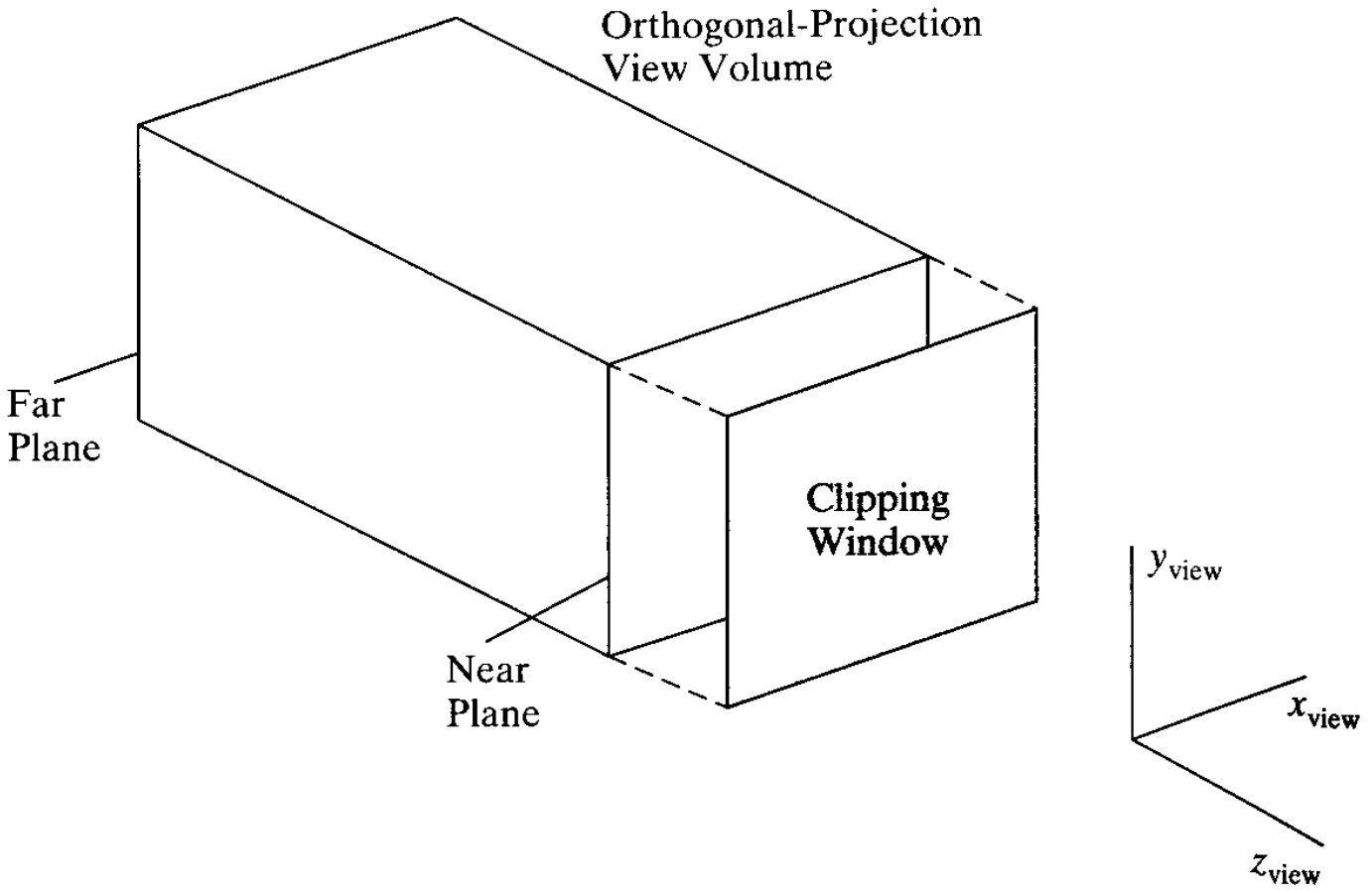
3D viewing co-ordinate parameters (4)



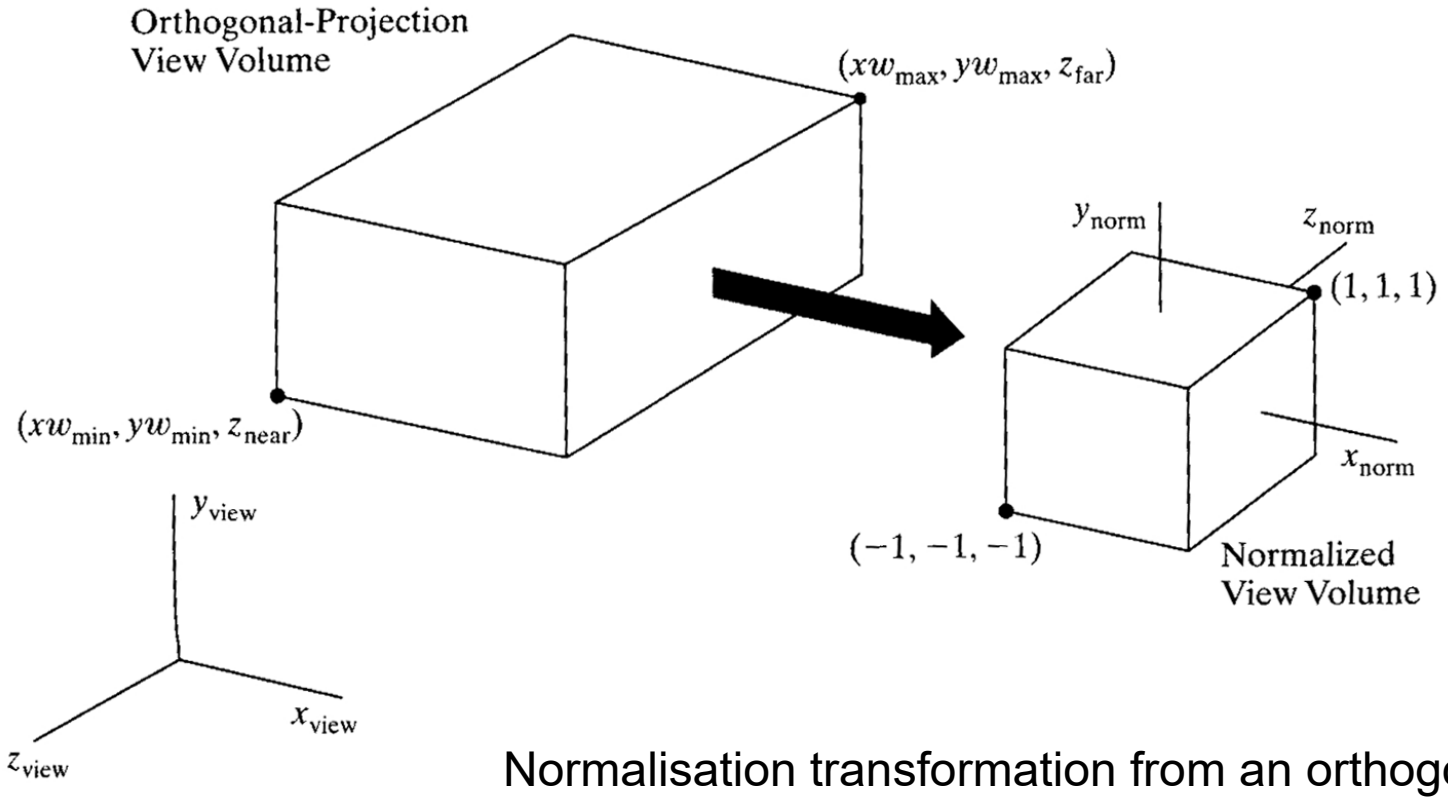
Orthogonal projection (1)

- Orthogonal (orthographic) projection is a transformation of object descriptions to a view plane along lines parallel to the view-plane normal vector **N**.
- It is often used to produce the front, side and top views of an object.
- Engineering and architectural drawings commonly employ these orthographic projections since the lengths and angles are accurately depicted and can be measured from the drawings.

Orthogonal projection (2)

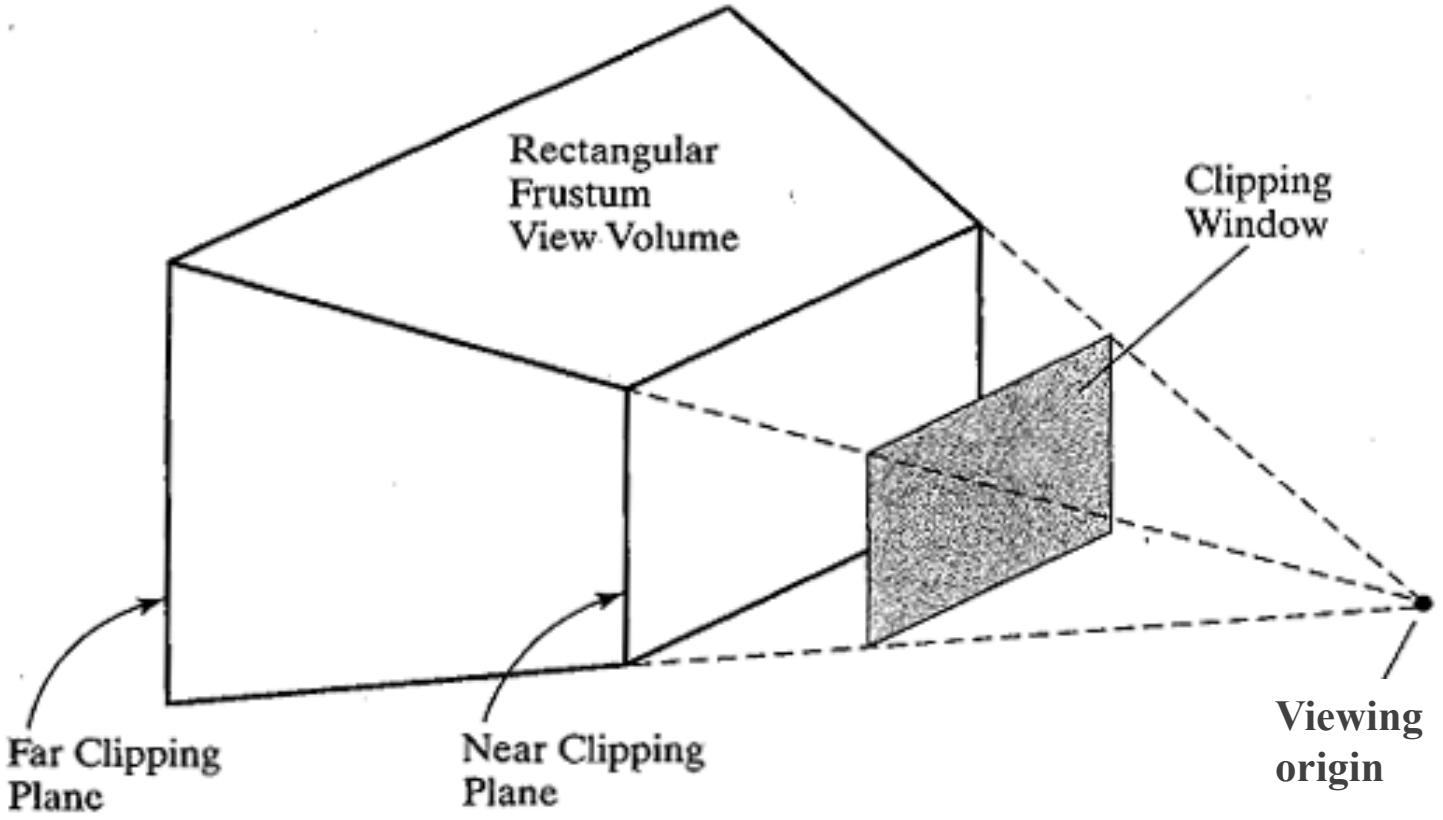


Orthogonal projection (3)



Normalisation transformation from an orthogonal-projection volume to the systematic normalisation cube within a reference frame

Frustum perspective projection (1)



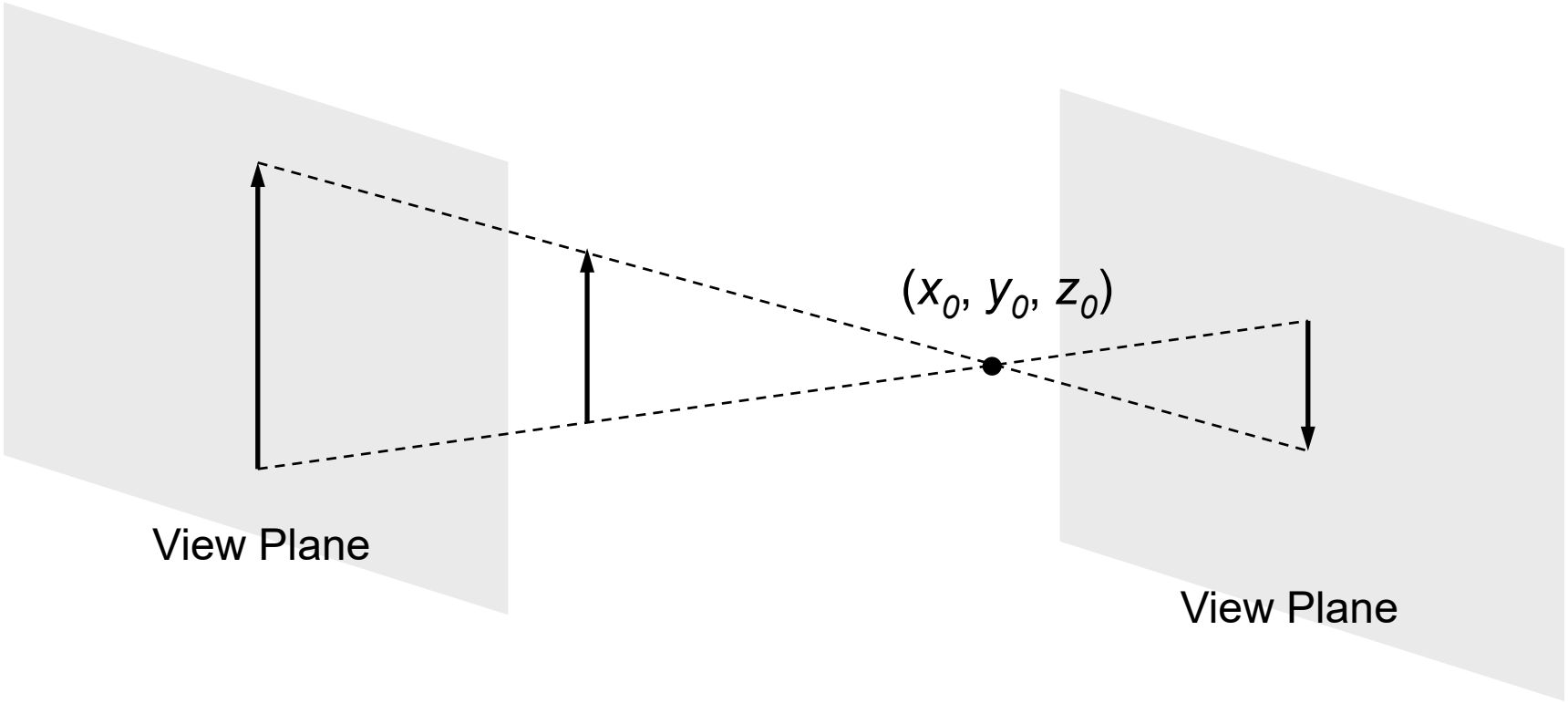
Frustum perspective projection (2)

- By adding near and far clipping planes that are parallel to the viewing plane, parts of the infinite perspective view volume are chopped off to form a truncated pyramid or frustum. These clipping planes can be optional for some systems.
- The near and far clipping planes can be used simply to enclose objects to be displayed. The near clipping plane can be used to take out large objects close to the viewing plane, which could be projected into unrecognisable shapes in the clipping window. Likewise, the far clipping plane can cut out objects that may be projected to small blots.

Frustum perspective projection (3)

- Some systems restrict the placement of the viewing plane relative to the near and far planes, and other systems allow it to be placed anywhere except at the position of the viewing origin (view point, viewing position, eye position or camera position).
- If the viewing plane is behind the projection reference point, objects are inverted on the view plane.
- If the viewing plane is behind the objects, the objects are simply enlarged as they are projected away.
- When the projection reference point is very far away from the view plane, a perspective projection approaches to a parallel projection.

Frustum perspective projection (4)

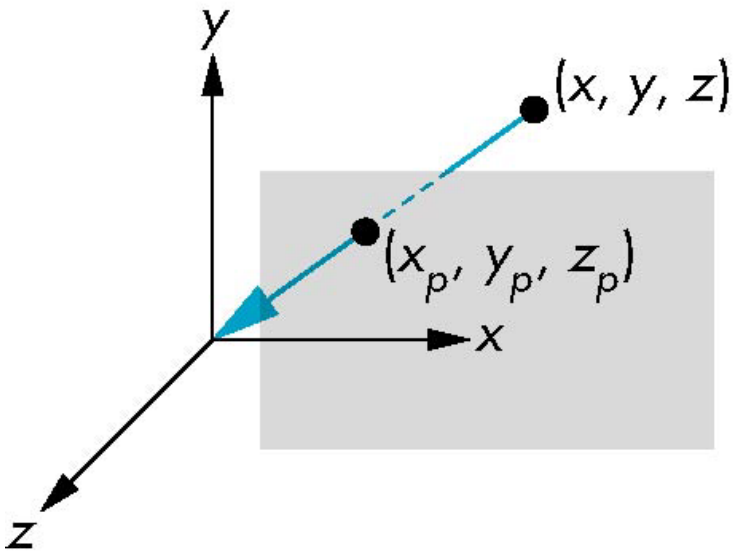


Objects are enlarged if the viewing plane is behind the objects.

Objects are inverted if the viewing plane is behind the projection reference point.

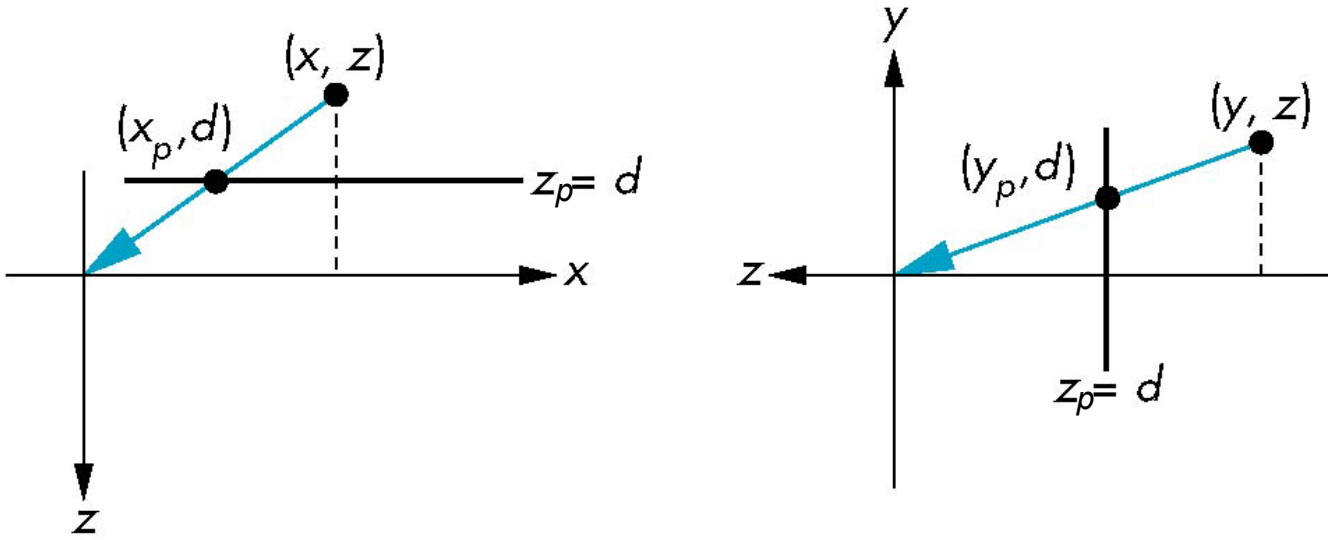
Simple perspective projection (1)

- Centre of projection at the origin
- Projection plane $z = d, d < 0$



Simple perspective projection (2)

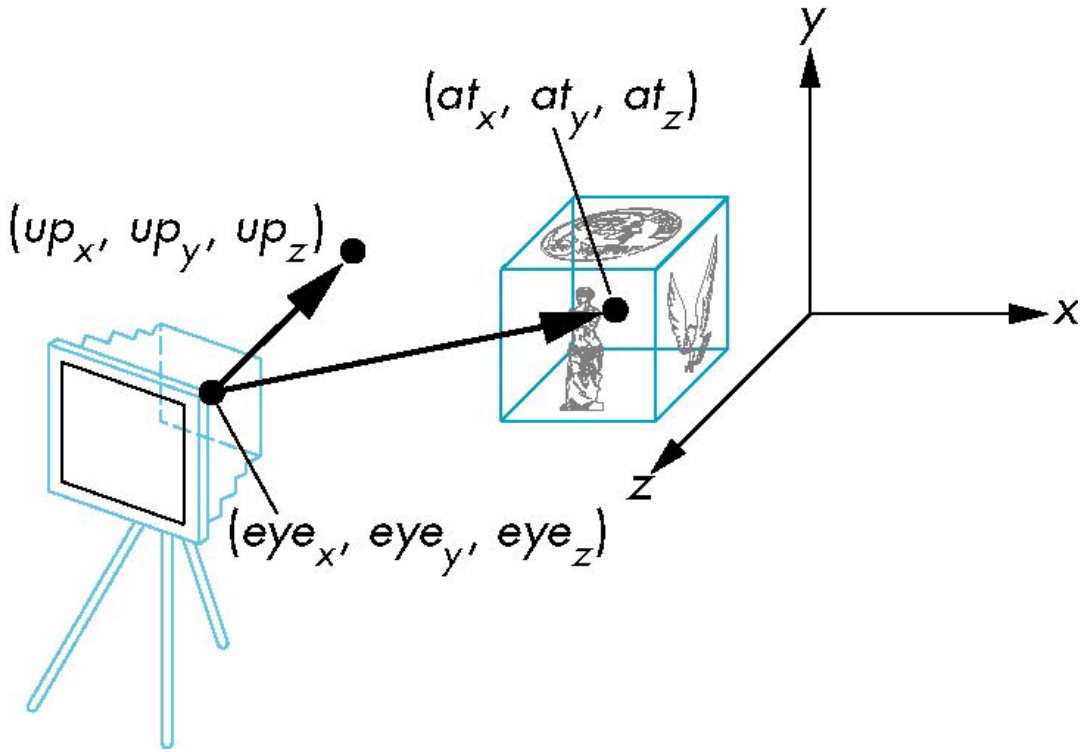
Consider top and side views



$$x_p = \frac{x}{z/d} \qquad y_p = \frac{y}{z/d} \qquad z_p = d$$

OpenGL functions (1)

- **`gluLookAt(eye_position, look_at, look_up)`**
  Specify three-dimensional viewing parameters.

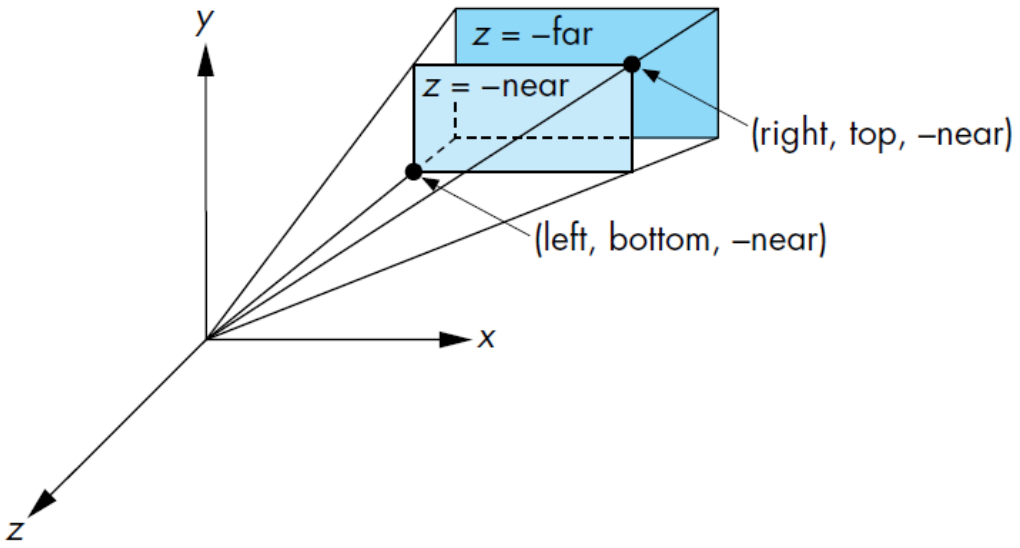


OpenGL functions (2)

- **`glOrtho(GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat near, GLfloat far)`**
  Specify parameters for a clipping window and the near and far clipping planes for an orthogonal projection.

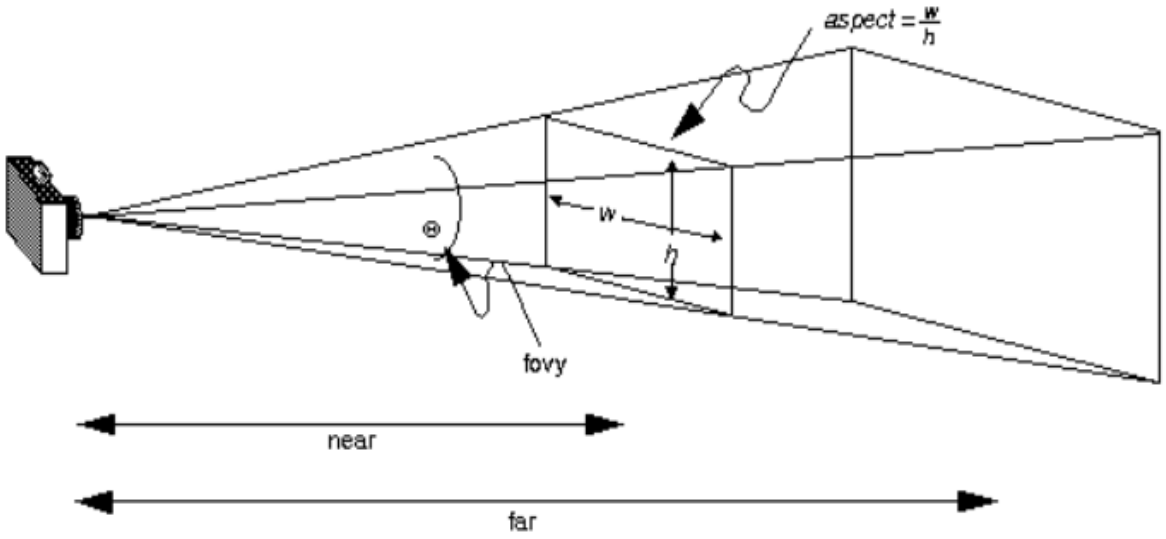
OpenGL functions (3)

- **`glFrustum(GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat near, GLfloat far)`**
  Specify parameters for a clipping window and the near and far clipping planes for a perspective projection (symmetric or oblique).



OpenGL functions (4)

- **`gluPerspective(GLfloat fov, GLfloat aspect, GLfloat near, GLfloat far)`**
  Specify field-of-view angle (**`fov`** which is a matrix) in the *y*-direction, the **`aspect`** ratio of **`near`** and **`far`** planes; It is less often used than **`glFrustum()`**.



An example of OpenGL program (1)

```
#define FREEGLUT_STATIC
#include <gl/freeglut.h>

GLint winWidth = 600, winHeight = 600;              // initial display window size

GLfloat x0 = 100.0, y0 = 50.0, z0 = 50.0;           // viewing co-ordinate origin
GLfloat xref = 50.0, yref = 50.0, zref = 0.0;       // look-at point
GLfloat Vx = 0.0, Vy = 1.0, Vz = 0.0;               // view-up vector

// co-ordinate limits for clipping window
GLfloat xwMin = -40.0, ywMin = -60.0, xwMax = 40.0, ywMax = 60.0;

// positions for near and far clipping planes
GLfloat dnear = 25.0, dfar = 125.0;

void init(void) {
     glClearColor(1.0, 1.0, 1.0, 0.0);

     // decides which matrix is to be affected by subsequent transform functions
     glMatrixMode(GL_MODELVIEW);
     gluLookAt(x0, y0, z0, xref, yref, zref, Vx, Vy, Vz);

     glMatrixMode(GL_PROJECTION);
     glFrustum(xwMin, xwMax, ywMin, ywMax, dnear, dfar);
}
```

An example of OpenGL program (2)

```
void displayFcn(void) {
      glClear(GL_COLOR_BUFFER_BIT);

      // parameters for a square fill area
      glColor3f(1.0, 0.0, 0.0);
      glPolygonMode(GL_FRONT, GL_FILL);
      glPolygonMode(GL_BACK, GL_LINE);

      // square in the x-y plane
      glBegin(GL_QUADS);
      glVertex3f(0.0, 0.0, 0.0);
      glVertex3f(100.0, 0.0, 0.0);
      glVertex3f(100.0, 100.0, 0.0);
      glVertex3f(0.0, 100.0, 0.0);
      glEnd();

      glFlush();
}

void reshapeFcn(GLint newWidth, GLint newHeight) {
      glViewport(0, 0, newWidth, newHeight); // Set viewport to window dimensions
}
```

An example of OpenGL program (3)

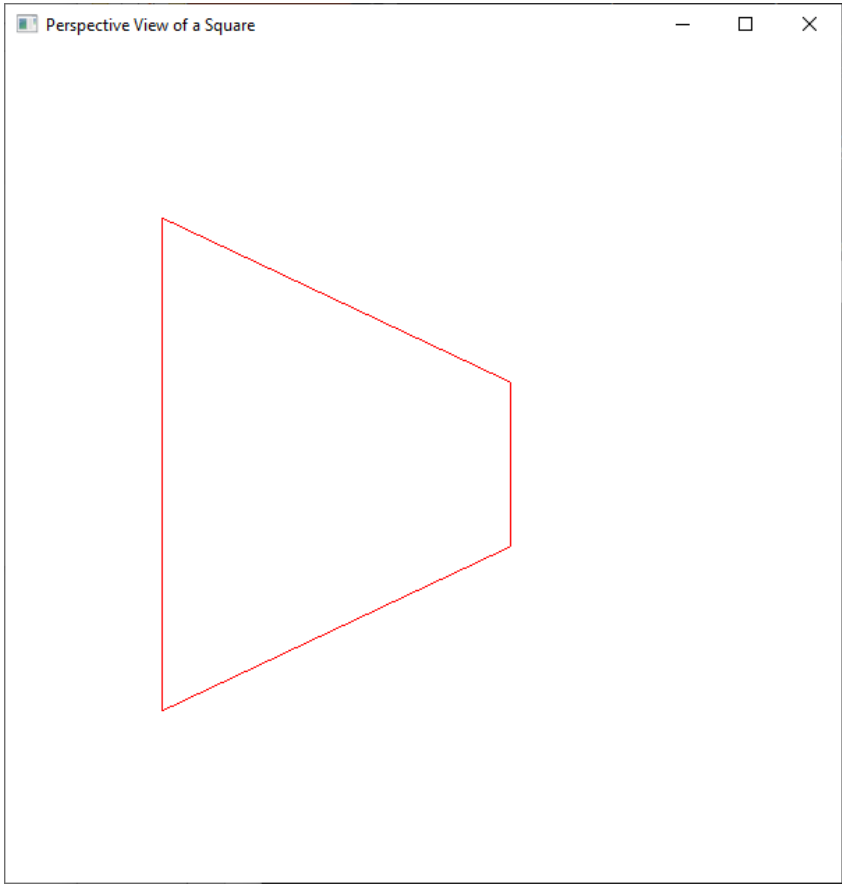
```
int main(int argc, char** argv) {
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
      glutInitWindowPosition(50, 50);
      glutInitWindowSize(winWidth, winHeight);
      glutCreateWindow("Perspective View of a Square");

      init();
      glutDisplayFunc(displayFcn);
      glutReshapeFunc(reshapeFcn);
      glutMainLoop();
}
```

An example of OpenGL program (4)

- This program displays a perspective view of a square which is defined in the *x*-*y* plane.
- A viewing co-ordinate origin is selected to view the front face at an angle. The perspective view is obtained using the **`glFrustum()`** function with the centre of the square as the look-at point.
- If the viewing origin is moved around to the other side of the square (e.g. z0 is initialised to -50), the back face will be displayed in wireframe (since the back face is constructed without filling in the area).

An example of OpenGL program (5)



Exercise

- Create an object of your choice using OpenGL geometric primitive functions;
- Display the object in the following views
  - Isometric
  - Side
  - Top
- Add more effects by applying continuous viewing and projection transformations (similar to last week's lab but not geometric transformations)