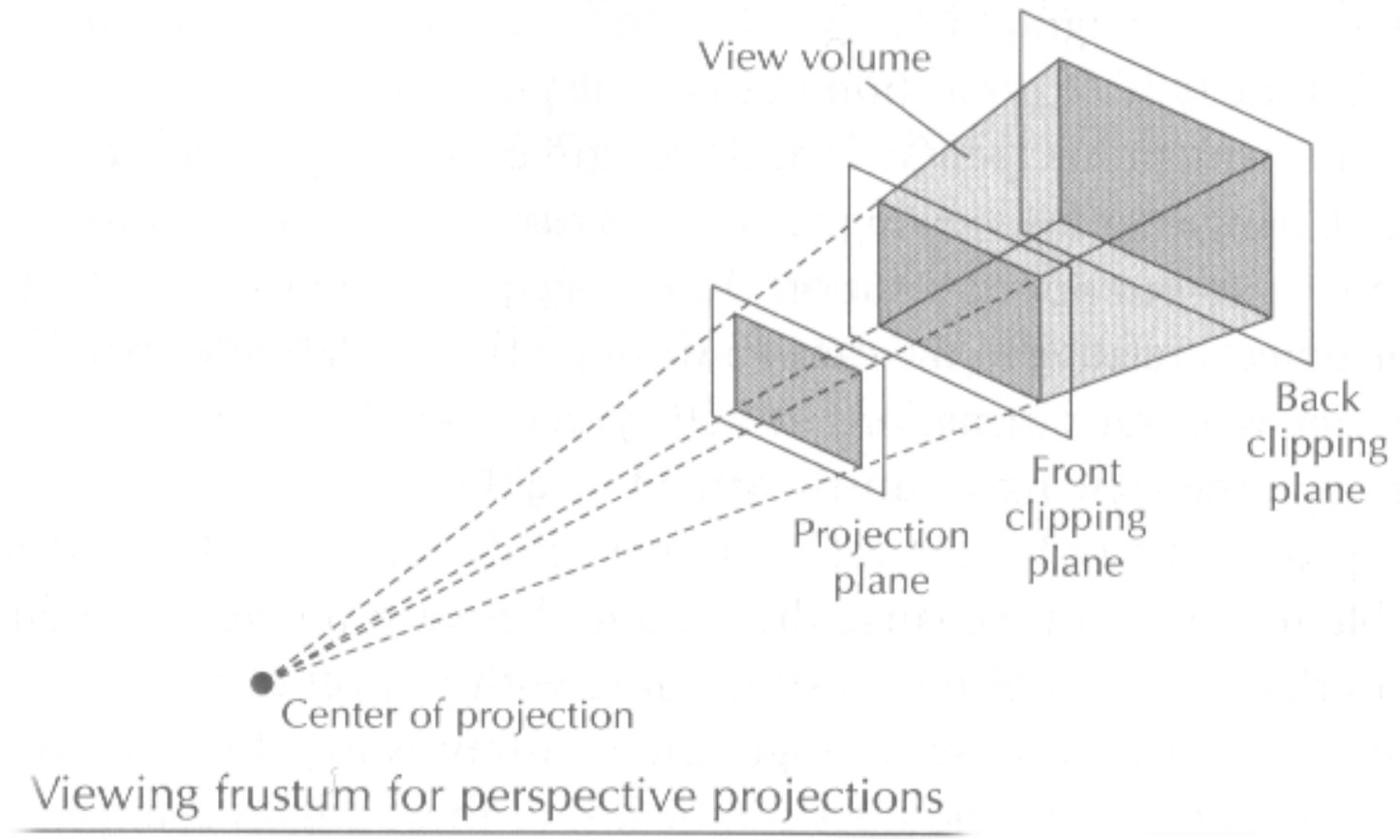


Viewing frustum for perspective projections

Monash University • Clayton's School of Information Technology

CSE3313 Computer Graphics

Lecture 18: Viewing, Perspective and Hidden Surface Removal in OpenGL

Viewing in OpenGL

Reference: Angel, Chapter 5

- In OpenGL the standard perspective transformation assumes:
  - (1) the **centre of projection** is at the origin of the viewing system;
  - the **viewing direction** is along the negative z axis;
  - the **view plane** is orthogonal to the z axis.
- Objects in world coordinates can be transformed relative to this viewing coordinate system by setting the matrix mode to `GL_MODELVIEW` and changing the **current transformation matrix.**
- OpenGL has a convenient way to specify the viewing transformation. We position a synthetic camera at a point $\mathbf{e}$ (called the **eyepoint**), in world coordinates.
- We specify a vector pointing from the eyepoint to a point, $\mathbf{a}$, called the **at point.**

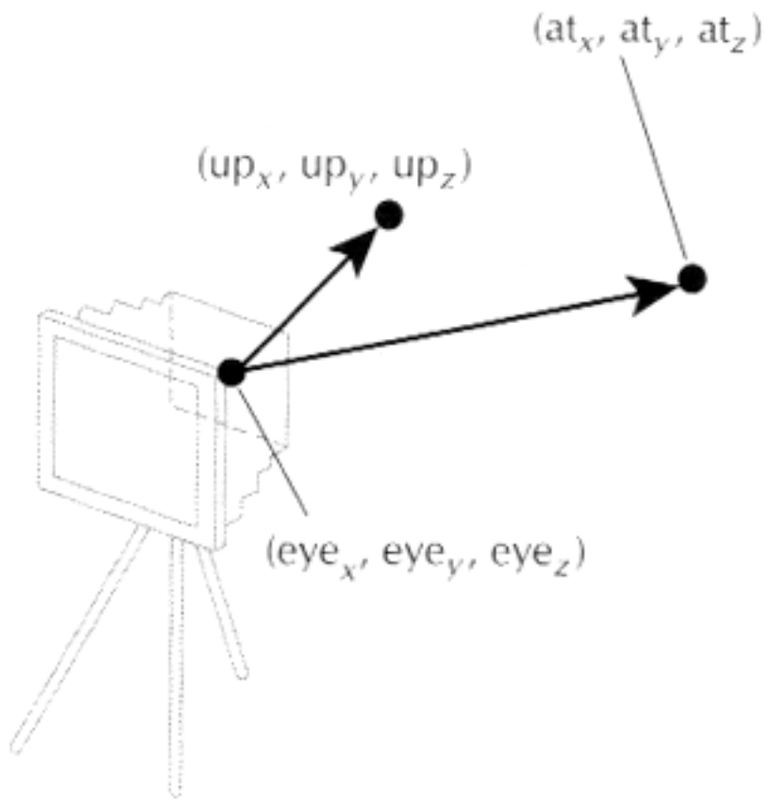
Viewing in OpenGL (cont.)

- Since the camera can be rotated around the viewing direction, we need to specify a **view up vector**.

- The OpenGL utility function

```
gluLookAt( eyex, eyey, eyez, atx, aty, atz,
           upx, upy, upz );
```

alters the model-view matrix for a camera pointed along this line.



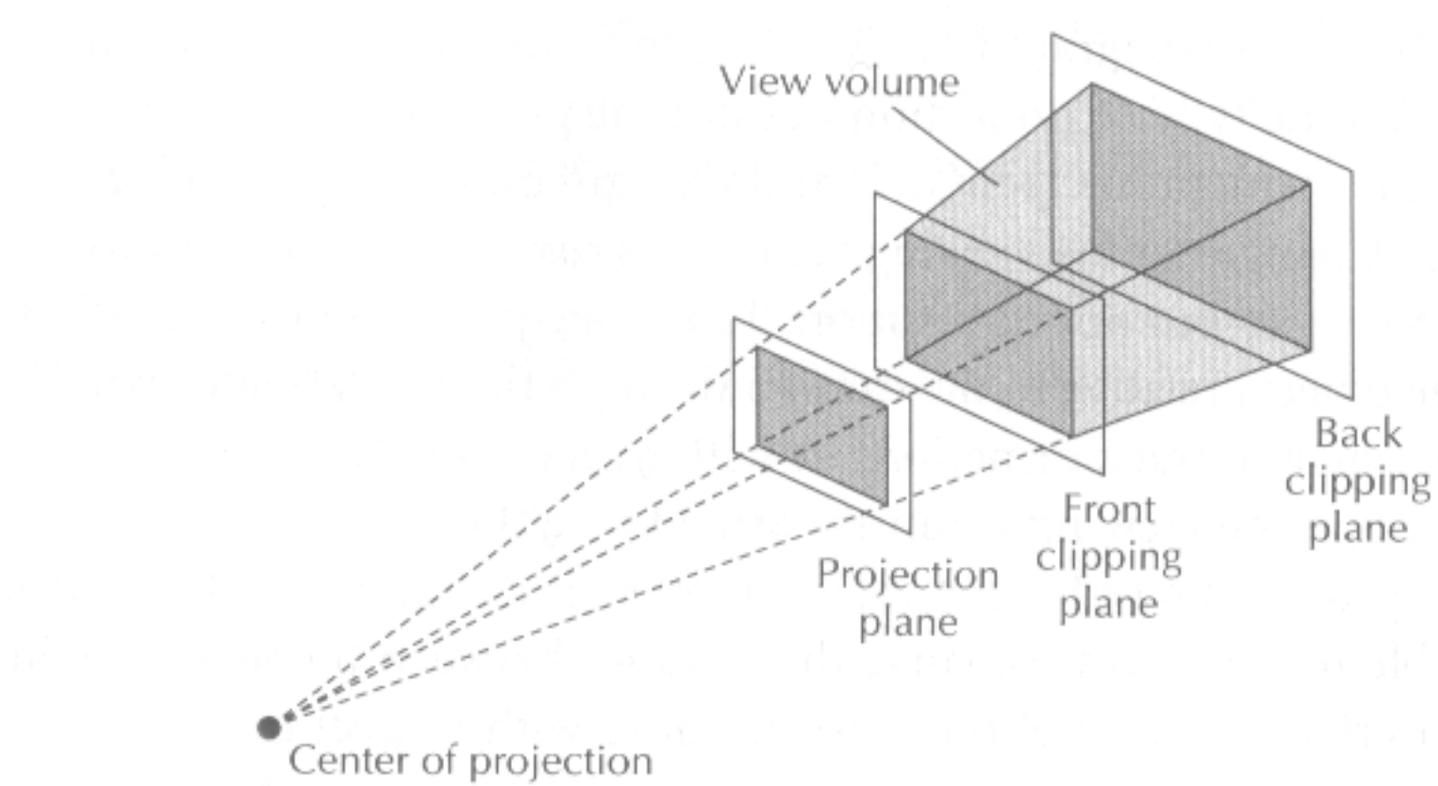
Perspective in OpenGL

- OpenGL provides two functions for specifying perspective views and one for parallel views.

- The function

```
glFrustrum(left, right, bottom, top, near, far)
```

specifies a **view volume** that is a **viewing frustum**.



Viewing frustum for perspective projections

glFrustum

- near and far define the front and back clipping planes, measured from the COP. Both must be positive, with `far > near`.

- `(left, bottom,near)` defines the bottom left corner of the front clipping window. `(right, top, near)` the top right coordinates.

- Viewing parameters are measured in camera or viewing coordinates.

- A typical code sequence is:

```
glMatrixMode( GL_PROJECTION );
glLoadIdentity( );
glFrustum(left, right, top, bottom, near, far);
glMatrixMode( GL_MODELVIEW );
```

- Note: the frustum does not have to be symmetric with respect to the z axis.

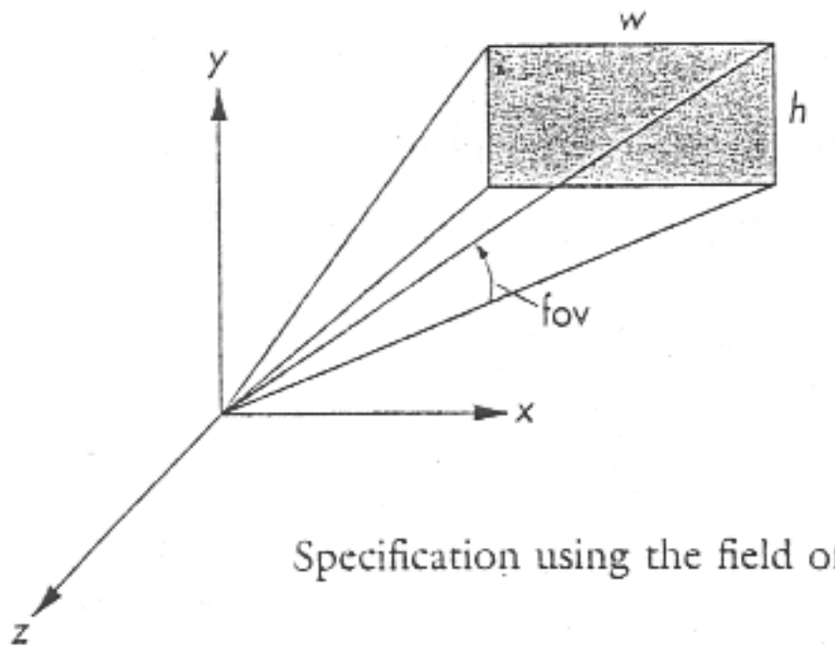
gluPerspective

- In many applications we may want to specify an angle or *field of view*. If the projection window is rectangular then there is a different angle for the horizontal and vertical components of the view.

- The OpenGL utility function

```
gluPerspective( fovy, aspect, near, far);
```

`fovy` specifies the angle of view in the up (y) direction, `aspect` specifies the aspect ratio – width/height. In this case the frustum is symmetric about the z axis.
This matrix postmultiplies the CTM.



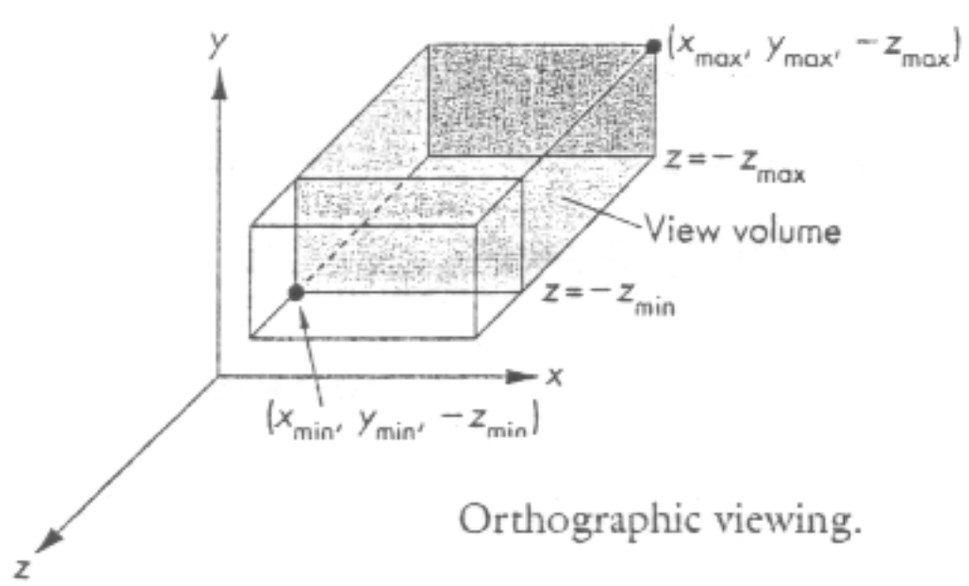
Specification using the field of view.

Parallel Viewing in OpenGL

- The only parallel viewing function provided by OpenGL is

```
glOrtho(left, right, bottom, top, near, far);
```

`near` and `far` need not be non-negative but we require `far` > `near`.



Orthographic viewing.

OpenGL and Hidden Surface Removal

- OpenGL allows the application program to specify that hidden surface removal should be carried out using a **depth buffer**.

```
glutInitDisplayMode( GLUT_RGB | GLUT_DEPTH );
glEnable( GL_DEPTH_TEST );
```

- In order to render a new scene the buffer can be cleared by

```
glClear( GL_DEPTH_BUFFER_BIT );
```

- Here OpenGL uses a **z-buffer** or **depth buffer** algorithm to remove hidden surfaces.

- To animate with hidden surfaces, the depth buffer must be cleared before each frame and then

```
glutSwapBuffers()
```

called.