Computer Graphics 9:
Clipping In 3D

Contents

In today's lecture we are going to have a look at some perspective view demos and investigate how clipping works in 3-D

- The clipping volume
- The zone labelling scheme
- 3-D clipping
  - Point clipping
  - Line clipping
  - Polygon clipping

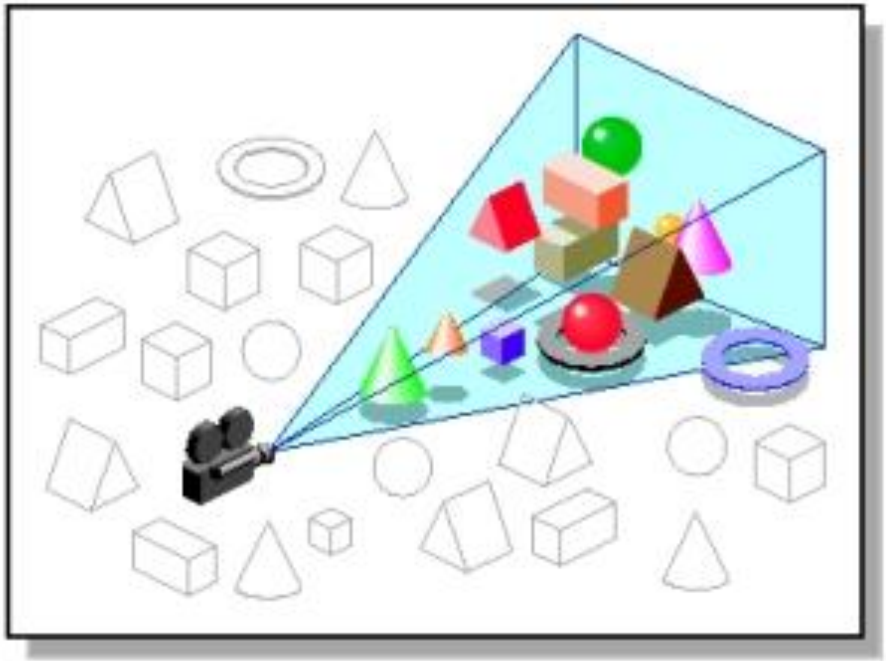
3-D Clipping

- Just like the case in two dimensions, clipping removes objects that will not be visible from the scene
- The point of this is to remove computational effort
- 3-D clipping is achieved in two basic steps
  – Discard objects that can't be viewed
    - i.e. objects that are behind the camera, outside the field of view, or too far away
  – Clip objects that intersect with any clipping plane

Discard Objects

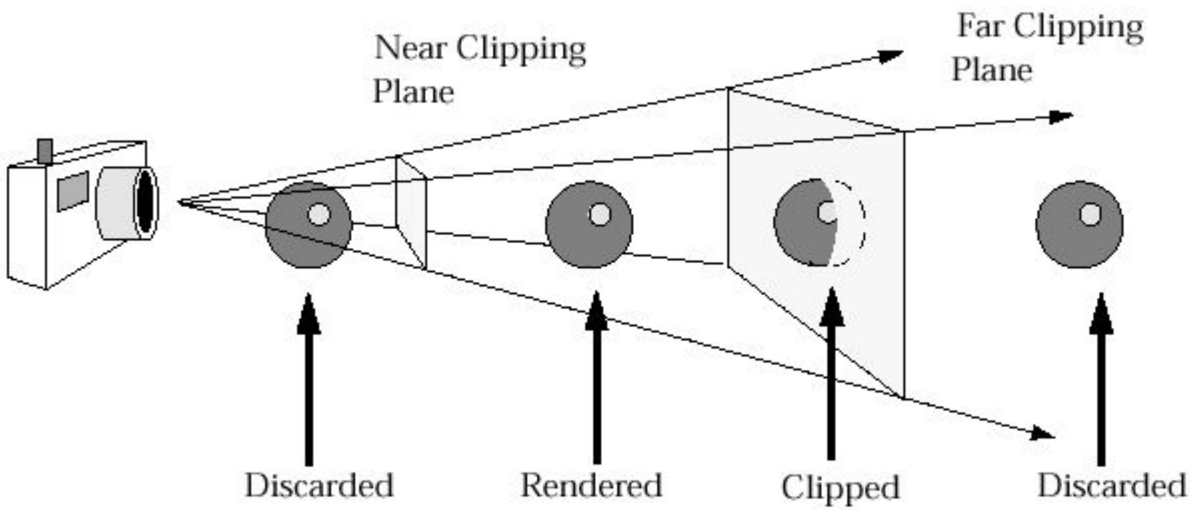
Discarding objects that cannot possibly be seen involves comparing an objects bounding box/sphere against the dimensions of the view volume

– Can be done before or after projection



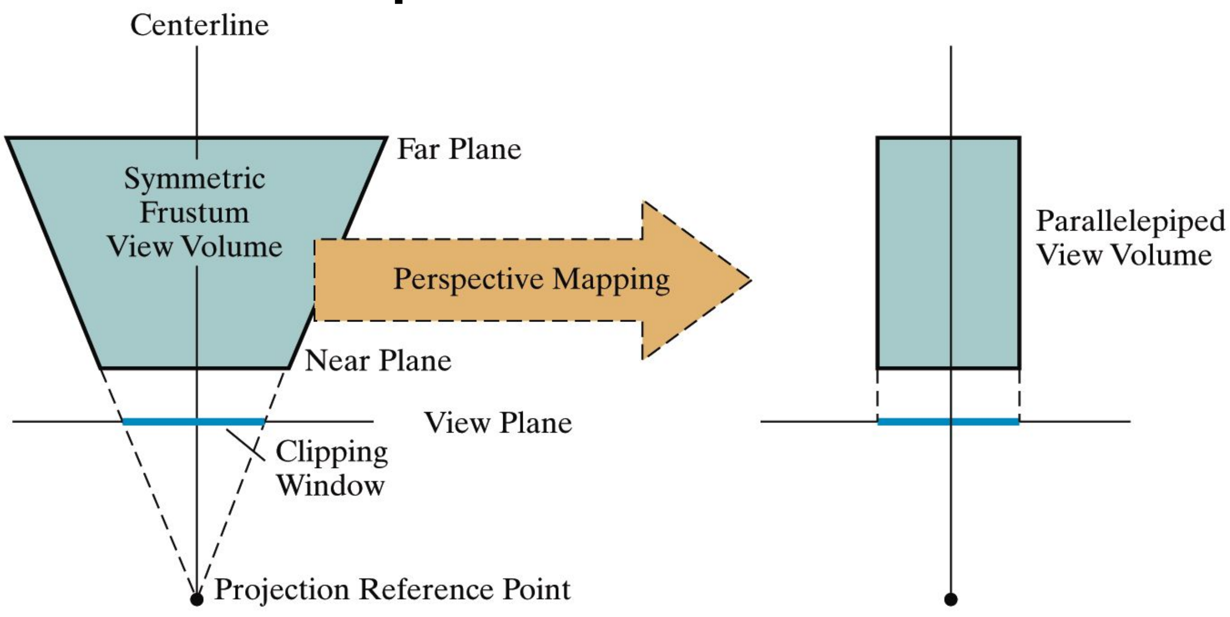
Clipping Objects

Objects that are partially within the viewing volume need to be clipped – just like the 2D case



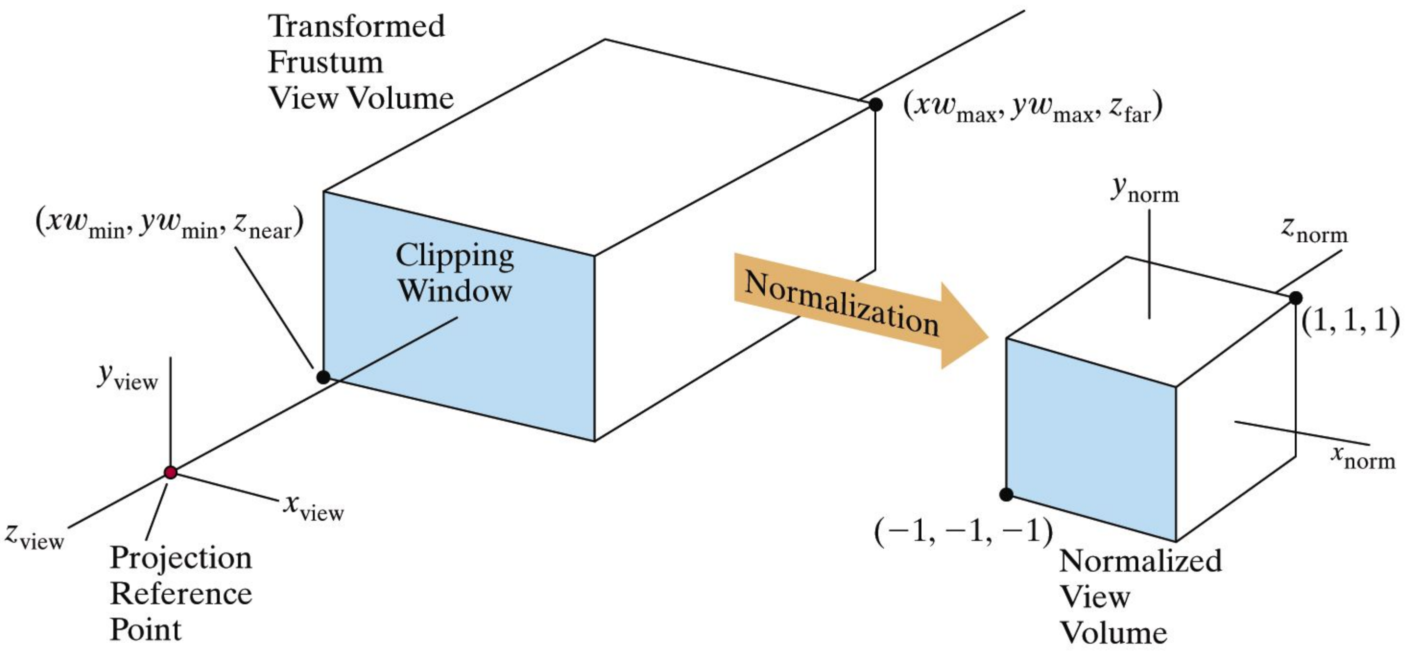
The Clipping Volume

After the perspective transformation is complete the frustum shaped viewing volume has been converted to a parallelopiped - remember we preserved all z coordinate depth information



Normalisation

The transformed volume is then *normalised* around position (0, 0, 0) and the z axis is reversed



When Do We Clip?

We perform clipping after the projection transformation and normalisation are complete

So, we have the following:

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = M \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

We apply all clipping to these homogeneous coordinates

Dividing Up The World

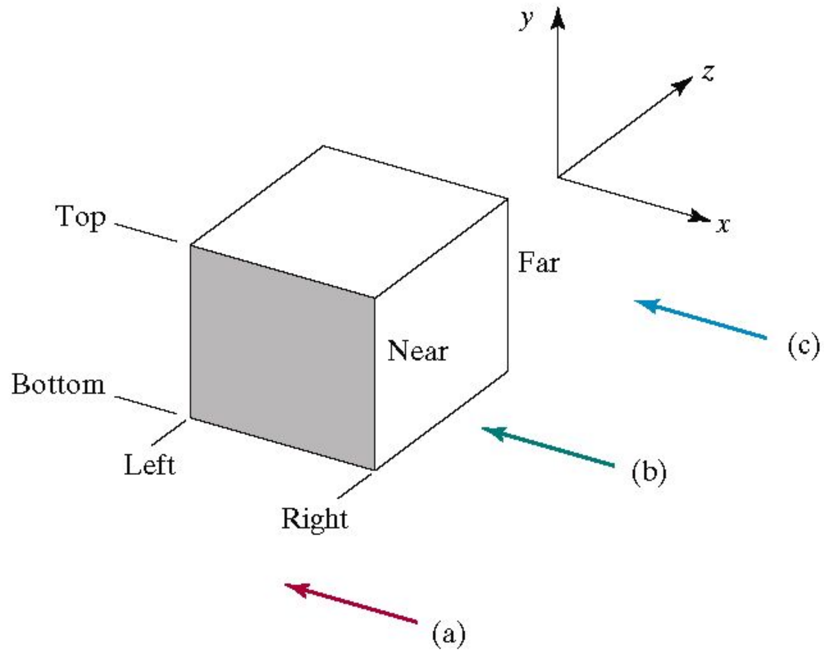
Similar to the case in two dimensions, we divide the world into regions

This time we use a 6-bit region code to give us 27 different region codes

The bits in these regions codes are as follows:

bit 6 Far	bit 5 Near	bit 4 Top	bit 3 Bottom	bit 2 Right	bit 1 Left

Region Codes



011001	011000	011010
010001	010000	010010
010101	010100	010110

Region Codes
In Front of Near Plane
(a)

001001	001000	001010
000001	000000	000010
000101	000100	000110

Region Codes
Between Near and Far Planes
(b)

101001	101000	101010
100001	100000	100010
100101	100100	100110

Region Codes
Behind Far Plane
(c)

Point Clipping

Point clipping is trivial so we won’t spend any time on it

Line Clipping

To clip lines we first label all end points with the appropriate region codes

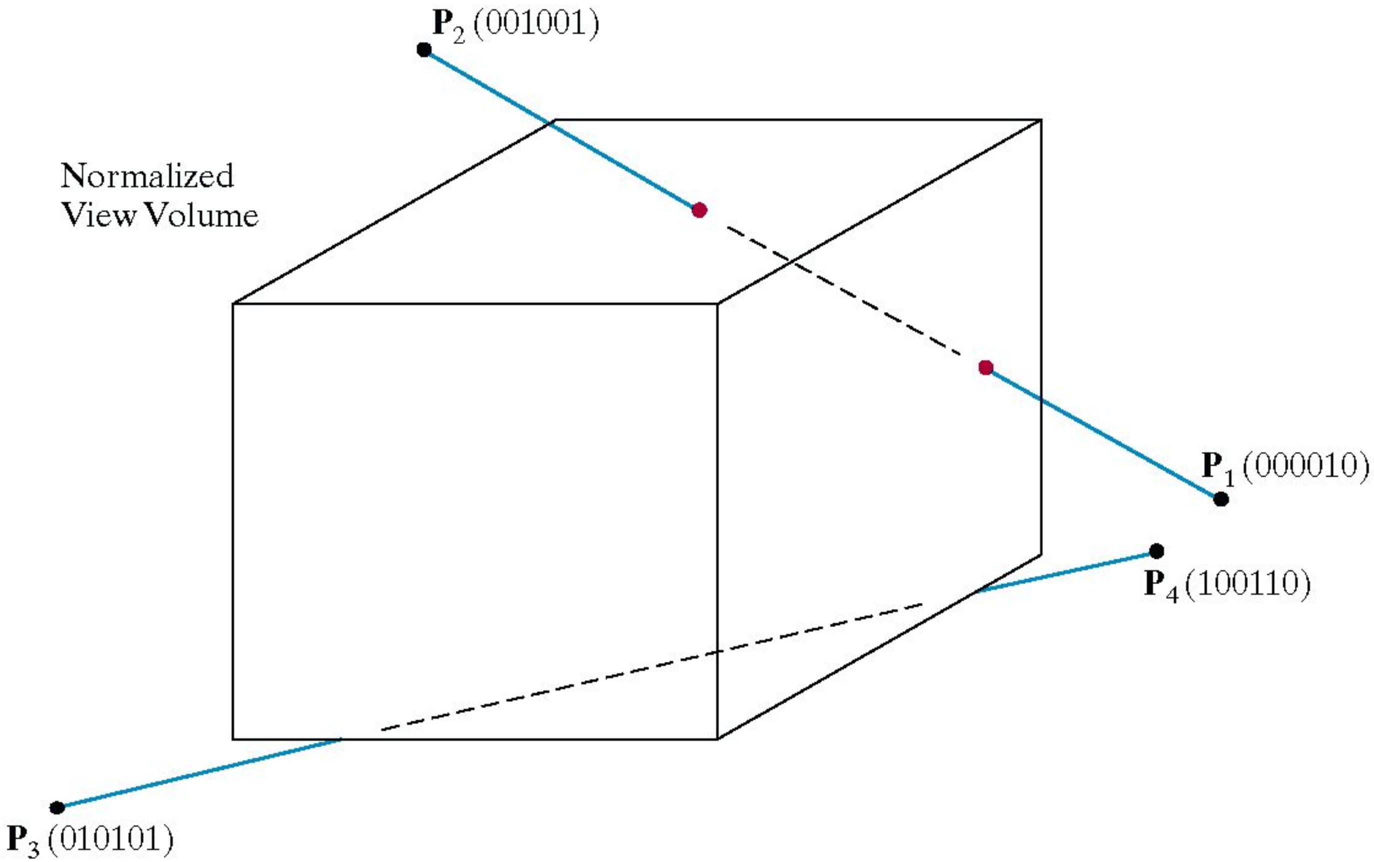
We can trivially accept all lines with both end-points in the [000000] region

We can trivially reject all lines whose end points share a common bit in any position

– This is just like the 2 dimensional case as these lines can never cross the viewing volume

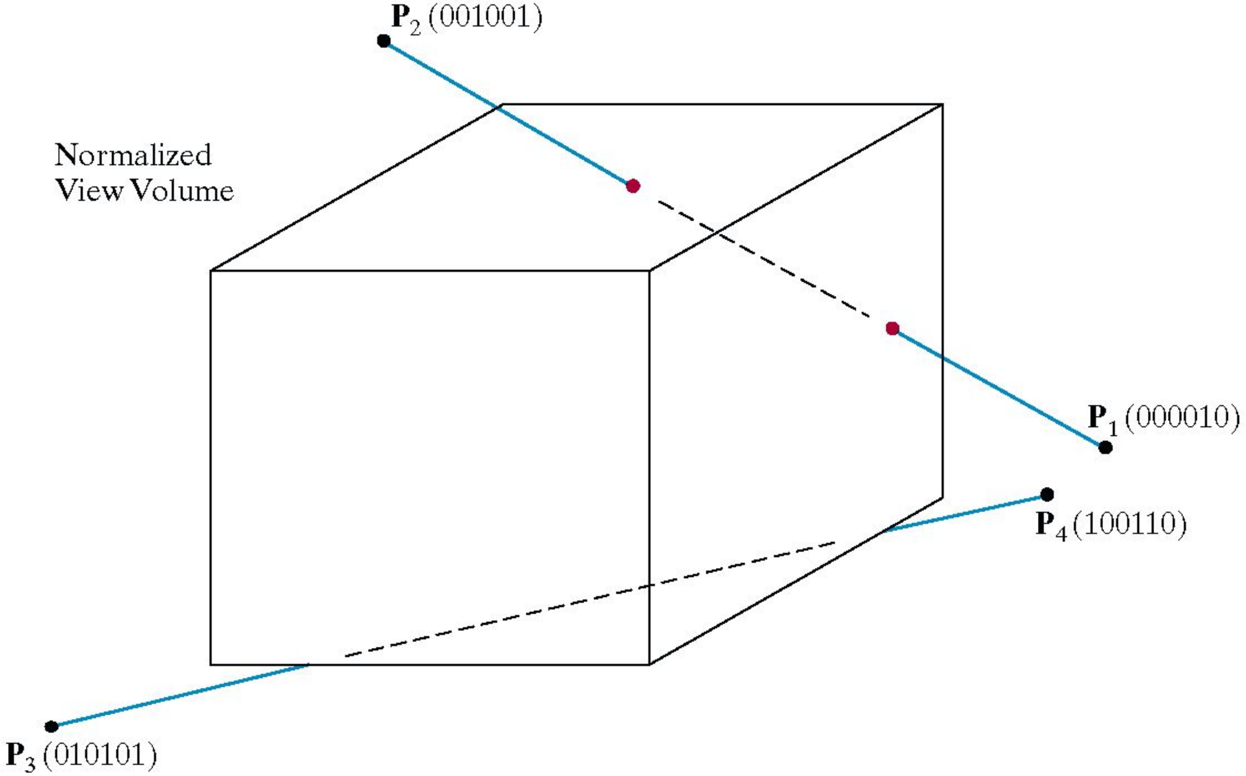
– In the example that follows the line from P_3[010101] to P_4[100110] can be rejected

Line Clipping Example



3D Line Clipping Example

Consider the line P_1[000010] to P_2[001001]
Because the lines have different values in bit 2 we know the line crosses the right boundary

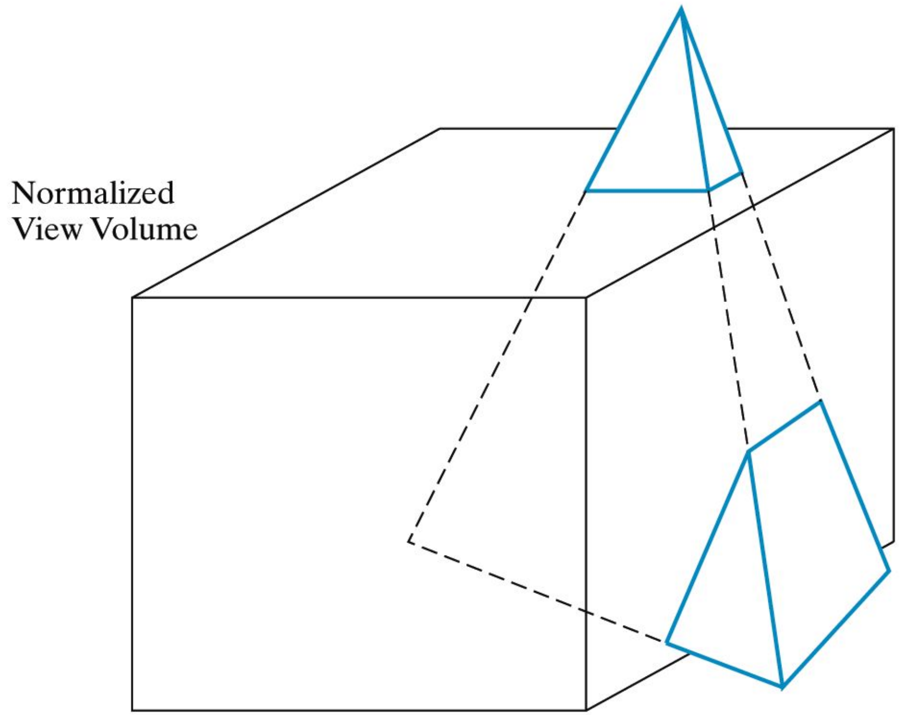


3D Line Clipping Example (cont…)

When then simply continue as per the two dimensional line clipping algorithm

3D Polygon Clipping

However the most common case in 3D clipping is that we are clipping graphics objects made up of polygons



3D Polygon Clipping (cont…)

In this case we first try to eliminate the entire object using its bounding volume

Next we perform clipping on the individual polygons using the Sutherland-Hodgman algorithm we studied previously

Cheating with Clipping Planes

For far clipping plane introduce something to obscure far away objects – fog

Make objects very near the camera transparent

Summary

In today's lecture we examined how clipping is achieved in 3-D