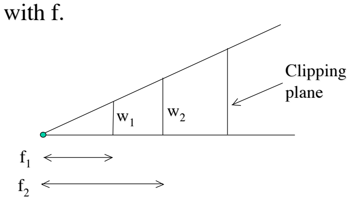
Further comments on the canonical frustum

u_{min}, u_{max}, v_{min}, v_{max}, are thought of as being in the camera coordinate system ==> units are that of world coordinate system

For assignment three, you need to choose u_{min}, u_{max}, v_{min}, v_{max}, and f.

I suggest simply setting u_{min}, u_{max}, v_{min}, v_{max}, to reflect your understanding of your screen window in world coordinates, and set f accordingly. (Best to keep the aspect ratio the same).

Note the approximate reciprocal relation of u_{min}, u_{max} and v_{min}, v_{max}, with f.



f_1 and w_1 give the same image as f_2 and w_2, but to see this in the math note that the camera center has shifted.

Because of this shift, the clipping plane values change, but f_1–B_1 == f_2–B_2

Determining the screen coordinates

Once you have (x,y) you need to map them back to the screen coordinates.

Use primes (‘) for mapped quantities (including projection) and carets (^) for screen quantities.

The canonical frustum gives the screen as a square that is 2f’ by 2f’. Note that f’ is between 0 and 1 (why?)

Our window on the screen has corners: $(\hat{u}_{\min}, \hat{u}_{\max})$ and $(\hat{v}_{\min}, \hat{v}_{\max})$ (Unless we want to distort things we assume the same aspect ratio as the camera window.)

Determining the screen coordinates

Our screen coordinates are then:

$$\hat{x} = \left(\frac{\hat{u}_{\max} + \hat{u}_{\min}}{2}\right) + \left(\frac{x'}{2f'}\right) \bullet \left(\hat{u}_{\max} - \hat{u}_{\min}\right)$$
$$\hat{y} = \left(\frac{\hat{v}_{\max} + \hat{v}_{\min}}{2}\right) + \left(\frac{y'}{2f'}\right) \bullet \left(\hat{v}_{\max} - \hat{v}_{\min}\right)$$

Or, equivalently:

$$\hat{x} = \hat{u}_{\min} + \left(\frac{x' + f'}{2f'}\right) \bullet \left(\hat{u}_{\max} - \hat{u}_{\min}\right)$$
$$\hat{y} = \hat{v}_{\min} + \left(\frac{y' + f'}{2f'}\right) \bullet \left(\hat{v}_{\max} - \hat{v}_{\min}\right)$$

Plan A: Clipping against the canonical frustum

2D algorithms are easily extended. For line clipping with Cohen Sutherland we use the following 6 out codes:

$y>-z \quad y<z \quad x>-z \quad x<z \quad z<-1 \quad z>z_{min}$

$(z_{min} = (f-F)/(B-f))$

Recall C.S for segments

Compute out codes for endpoints
While not trivial accept and not trivial reject:
 Clip against a problem edge (one point in, one out)
 Compute out codes again
Return appropriate data structure

Clipping against the canonical frustum

Clipping polygons in 3D against canonical frustum planes is simpler and more efficient than the general case.

Recall the S.H. gives four cases:

- Polygon edge crosses clip **plane** going from out to in
  - emit crossing, next vertex
- Polygon edge crosses clip **plane** going from in to out
  - emit crossing
- Polygon edge goes from out to out
  - emit nothing
- Polygon edge goes from in to in
  - emit next vertex

(The above is from before, just change “edge” to “plane”)

Object in world coordinates (after modeling transforms)

↓

Transform object from world coordinates to standard camera coordinates ✓

↓

Clip against canonical view frustum

↓

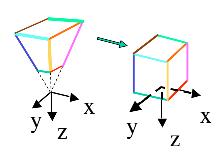
Project using standard camera model ✓

Plan A: Clip against canonical frustum ✓ (relatively easy—we chose the canonical frustum so that it would be easy!)

Plan B: Be even more clever. Further transform to cube and clip in homogenous coordinates.

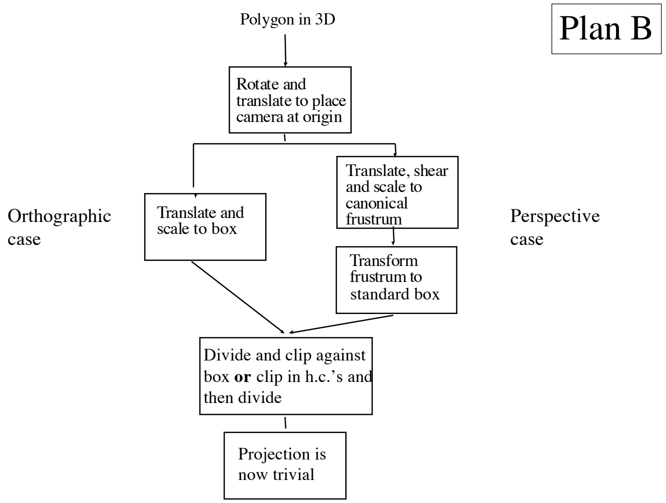
Plan B: Clipping in homogenous coords

- For any camera, can turn the view frustrum into a regular parallelepiped (box). We will use the box bounded by x = ±1, y = ± 1, z = -1, and z = 0.

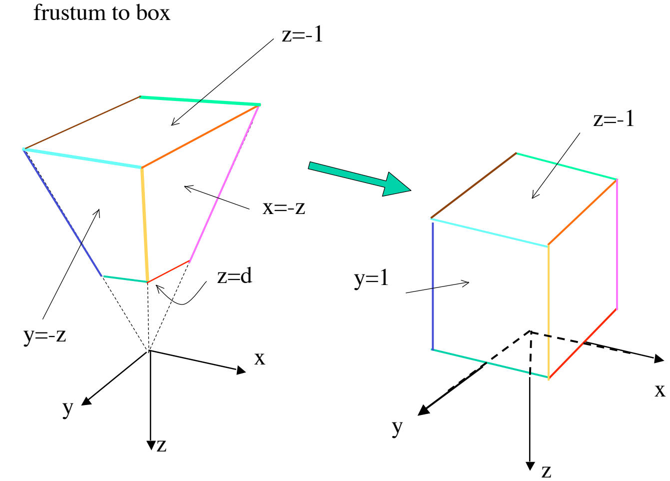


- Advantages
  - Simplified clipping in homogenous coordinates
  - Extends to cases where we use homogenous coordinates to represent additional information (and w could be negative).
  - Can simplify visibility algorithms.
- Approach: clever use of homogenous coordinates

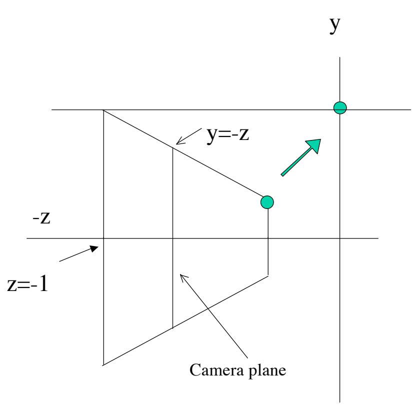
Plan B



Transforming canonical frustum to box

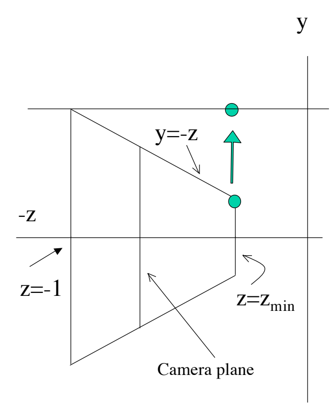


Transforming canonical frustum to box



Do this in two steps. One stretch in y (and x), and one stretch in z.

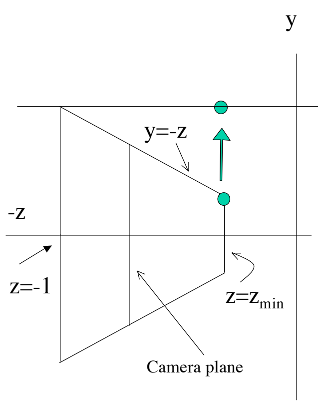
Transforming canonical frustum to box



The picture should suggest an appropriate scaling for y.

It is ?

Transforming canonical
frustum to box



On top, y—> 1, so scaling is (1/y)
Recall that y=-z there.

On bottom, y—> -1 so scaling is (–1/y). Recall that y=z there.

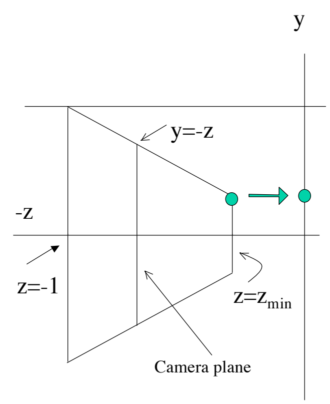
So scaling is y'= y/(-z)

Similarly, x' = x/(-z)

Transformation is **non-linear**, but in h.c., we can make w = (-z).

(So, in h.c., x and y transforms are trivially the identity)

Transforming canonical
frustum to box



For z, we translate near plane to origin (shift everything by - z_{min}).

But now box is too small. Specifically it has z dimension (1 + z_{min}) (recall z_{min} is negative)

So we have an extra scale factor 1 / (1 + z_{min}) and thus
z'=(z - z_{min}) / (1 + z_{min})

But we want x and y to work nicely in h.c., with w=-z, so we use

z' = ((z - z_{min}) / (1 + z_{min})) / (-z)

(Thus in our box, depth transforms **non-linearly**)

In h.c.,

x=>x

y=>y

z=>(z - z_{min}) / (1 + z_{min})

1=>-z

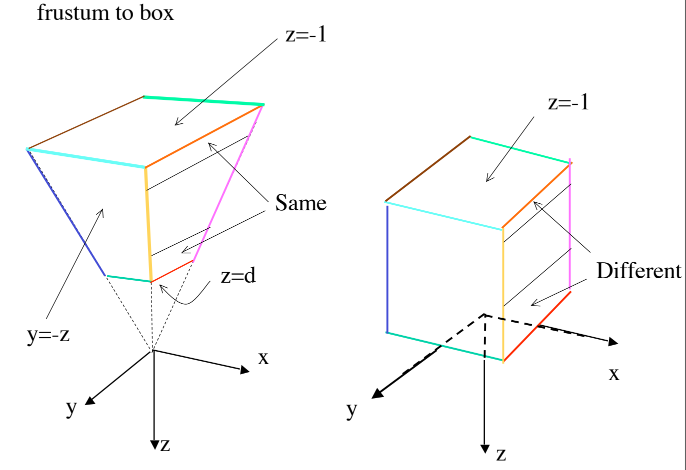
So, the matrix is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{1}{1+z_{min}} & \dfrac{-z_{min}}{1+z_{min}} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Mapping to standard view volume (additional comments)

- The mapping from [z_{min}, -1] to [0,-1] is non-linear. (Of course, there exists a linear mapping, but not if we want everything else to work out nicely in h.c.).
- So a change in depth of $\triangle$ D at the near plane maps to a larger depth difference in screen coordinates than the same $\triangle$ D at the far plane.

Transforming canonical frustum to box



Mapping to standard view volume (additional comments)

- But depth order is preserved (important!); the function is monotonic (proof?).
- And lines are still lines (proof?) and planes are still planes (important!).

Clipping in homogeneous coordinates

- We have a cube in (x,y,z), but it is **not** a cube in homogeneous coordinates, so we must divide if we want to take advantage of this particularly nice clipping situation.
- However, dividing before clipping might be inefficient if many points are excluded, so we often clip in homogeneous coordinates.

Clipping in homogeneous coord.'s

- Write h.c.'s in caps, ordinary coords in lowercase.
- Consider case of clipping stuff where x>1, x<-1
- Rearrange clipping inequalities:

$$\left(\frac{X}{W}\right)>1 \qquad \left(\frac{X}{W}\right)<-1$$

becomes

$$X>W, \quad X<-W, \quad W>0$$

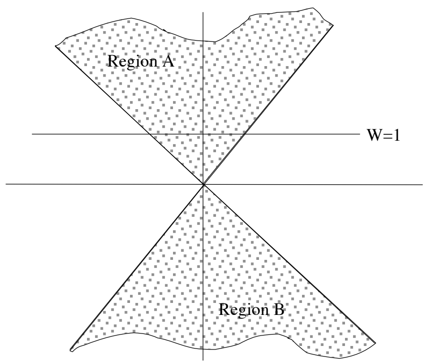
AND

$$X<W, \quad X>-W, \quad W<0$$

(So far W has been positive, but negatives occur if we further overload the use of h.c.'s)

Clipping in homogeneous coord.'s

The clipping volume in cross section



Clipping in homogeneous coord.'s

- If we know that W is positive (the case so far!), simply clip against region A
- If we are using the h.c. for additional deferred division, then W can be negative.
- If W is negative, then we use region B. The clipping can be done by negating the point, and clipping against A, due to the nature of A and B.
- Case where object has both positive and negative W is a little more complex.
- Notice that the actual clipping computations are not that different from the case in Plan A---no free lunch!