Hidden Surface Removal

When modelling a 3D world we have to take into account that some surfaces are non-visible because they are blocked, i.e. behind other surfaces. How do we determine what is visible and what is hidden?

Two approaches

- Solve the problem in world space. Try to work out geometrically what goes behind what using the (real) 3D world coordinates, and then draw the result. ← extremely hard
- Solve the problem in display space.
  During scan-version, whenever we generate a pixel P, we determine whether some other vertex (from the world coordinates) ⊗, closer to the eye/camera also maps to P. The closest one will be drawn, the others won't.

The Z-Buffer

- For every pixel in the display memory, there is a corresponding entry in the Z-buffer
- The Z-buffer is used to keep a record of the z-value / z-depth of each pixel
- Here's the algorithm:
  1. Initialise each pixel to desired background colour
  2. Initialise each Z-buffer entry to MAX_DEPTH
  3. For each pixel **P** generated during scan-conversion,
     if (z-coordinate of **P** < Z_BUFFER[P])
     Compute colour of **P**, store colour in **P**, update Z_BUFFER[P]
     else // z-coordinate of P = Z_BUFFER[P]
     Don't change **P** because something closer is already mapped to **P**

Z-fighting

Lack of precision in the Z-buffer leads to incorrect rendering of pixels with similar Z-values

Solution: glPolygonOffset()