# Lighting and Rasterization - Visible Surface Determination

# Intended Learning Outcomes

- Understand the goal of visible surface determination
- Describe the method of back-face detection
- Describe the method of Z-buffer method
- Describe the method of ray casting
- Able to program visible surface determination techniques

# Visible Surface Detection

- Also called Hidden Surface Elimination
- Only visible surfaces should be rasterized
- The problem is not easy as has to handle partially visible scenarios –
  - Concave objects
  - one object partially in front of each other

# Three Methods

- Back-face detection (also called Culling)
- Z buffer (also called depth buffer)
- Ray Casting



- Back-face detection is always run as a preliminary test. It is fast and reduces about half of the workload before further processing.



- Other methods also exist: e.g. painter's algorithm, A buffer method, …

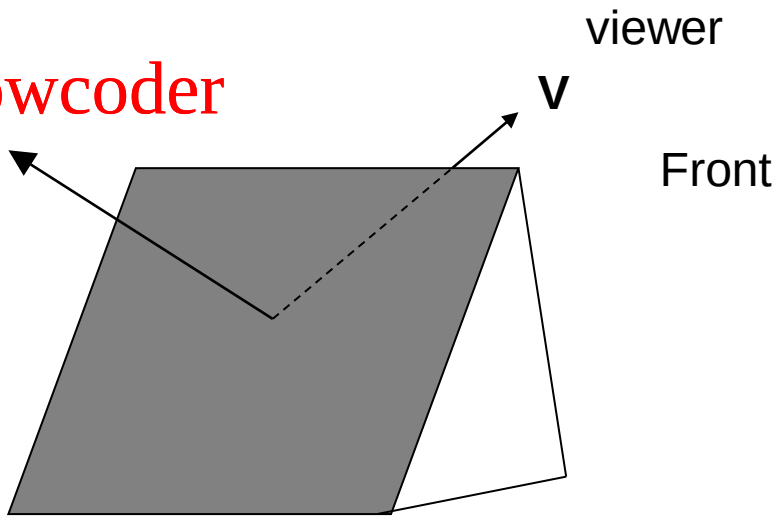
# Back-Face Detection /Culling

- Fast and simple
- Use as a preliminary step before more sophisticated visibility tests
- Eliminates ≈ 50% of faces from further consideration



$\mathbf{N}\cdot\mathbf{V} < 0 \Rightarrow$ back face

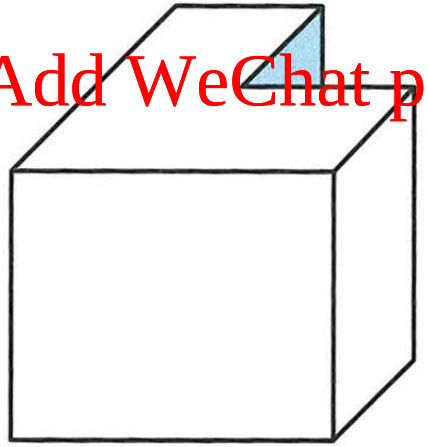
$\Rightarrow$ eliminate



- Sometimes, **v** is replaced by the **VPN** for faster approximate processing

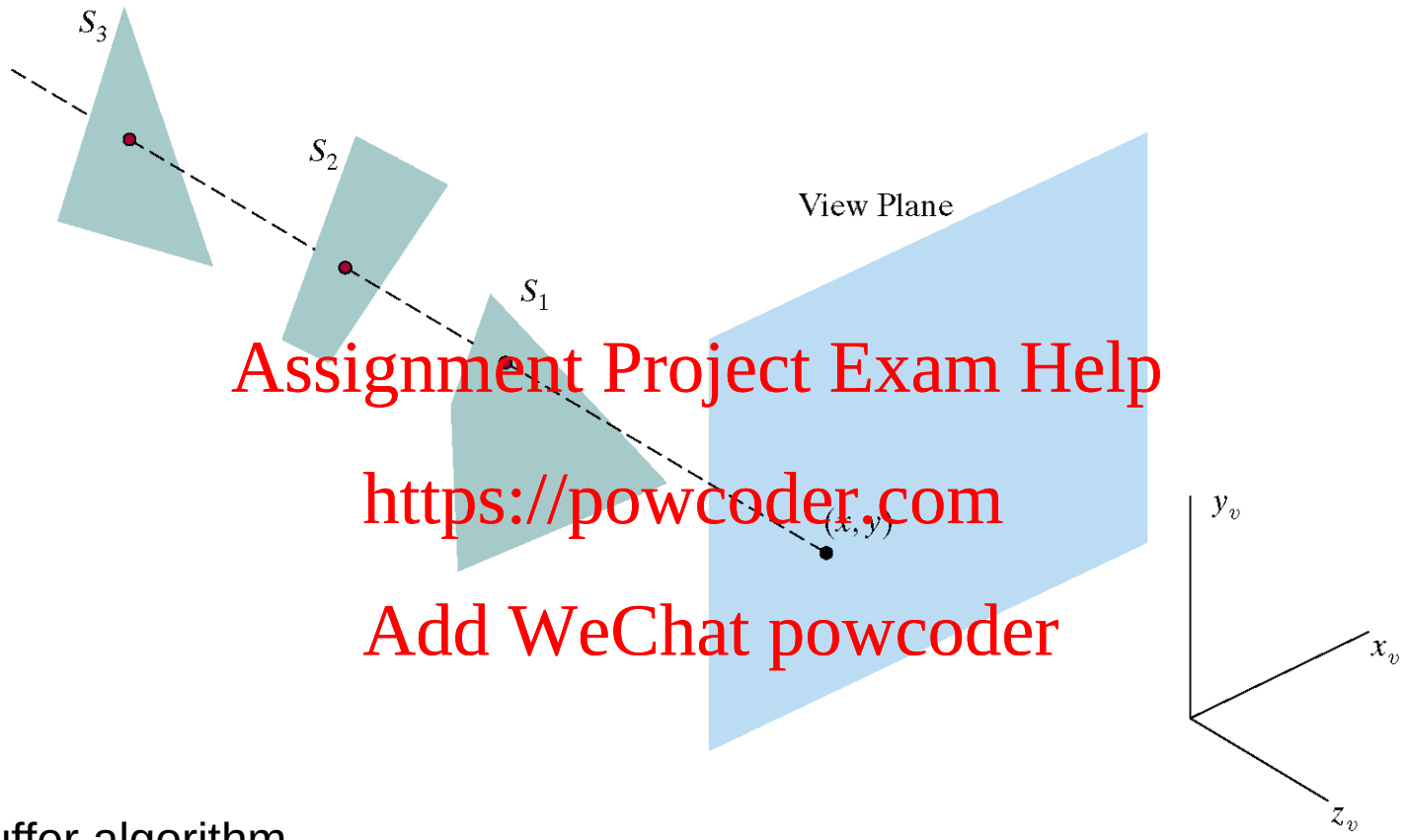
- Disadvantage: cannot handle concave object or partially overlapping object





# Z Buffer

- Also called depth buffer method
- Two buffers
  - *Z* /Depth buffer : store depth values for each (x, y) position
  - *Frame* /*Refresh* buffer : store colour values for each (x,y) position
- Buffer stores the current visible surface information, values are updated as soon as new visible information found






Z buffer algorithm

Three surfaces overlapping pixel position $(x, y)$on the view plane. The visible surface, $S_1$, has the smallest depth value.

# Algorithm

1. Initialize the depth buffer and frame buffer so that for all buffer positions (x, y)

   depthBuff (x, y) = 1.0,  frameBuff (x, y) = backgndColor



2. Process each polygon in a scene, one at a time.
   a. for each projected (x, y) pixel position of a polygon, calculate the depth z (if not already known).
   b. If z < depthBuff(x, y), compute the surface colour at that position and set



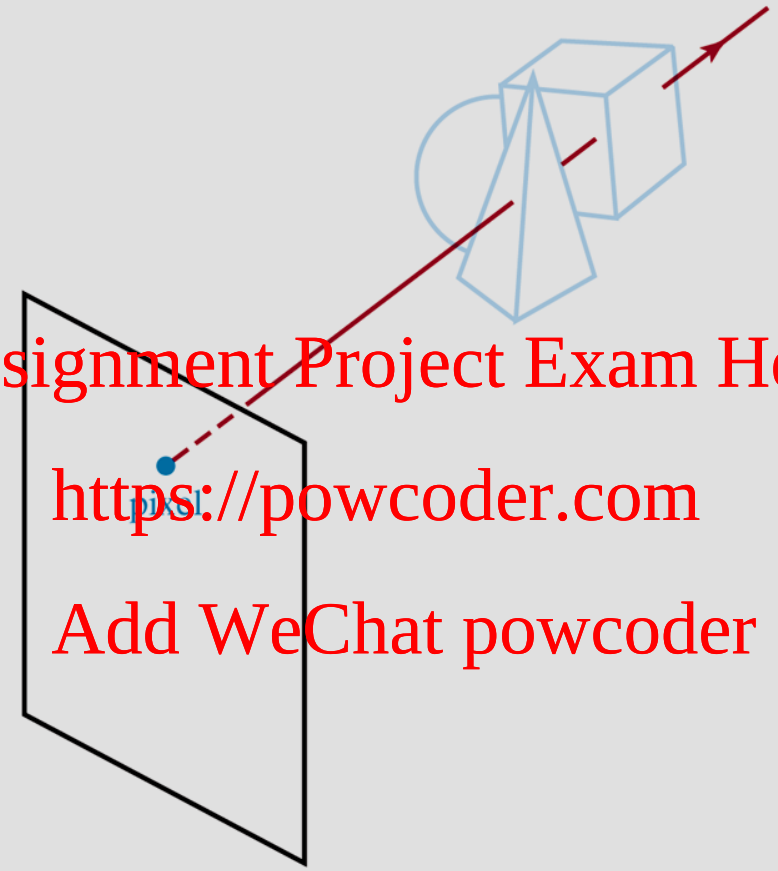


   depthBuff (x, y) = z,  frameBuff (x, y) = surfColor (x, y)

After all surfaces have been processed, the depth buffer contains depth values for the visible surfaces and the frame buffer contains the corresponding colour values for those surfaces.

# Ray Casting

- retrace the light paths of the rays that arrive at the pixel
- for each pixel, send a ray from PRP that goes through the pixel
- find all intersections of the ray with the surfaces
- the nearest intersections is the visible part of the surface for that pixel




Ray casting

A ray along the line of sight from a pixel position through a scene.

# Comparison of Z buffer and Ray Casting

| Method | Good for situations |
|---|---|
| Z buffer | Objects that cannot be adequately described by simple equations |
| Ray casting | Objects that can easily be described by simple equations |

# OpenGL Functions

- Back face removal

  *glEnable (GL_CULL_FACE);*

  *glCullFace (GL_BACK);*

- Z Buffer

  *glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);*

  *glClear (GL_DEPTH_BUFFER_BIT);*

  *glEnable (GL_DEPTH_TEST);*

# References

- Text: Ch. 16.1- 16.3, 16.10-11 for various visibility determination methods
- Text: Ch. 16.14 for OpenGL commands