VISIBLE SURFACE DETECTION

Amrita Kaur, Assistant Professor
Sushma Jain, Associate Professor
Anupam Garg, Assistant Professor
Santarpal Singh, Assistant Professor

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Outline

- Basics of Visible Surface detection
- Image-Space method vs. object-Space method
- Depth Cueing
- Depth buffer Algorithm

Visible Surface Detection

- Visible surface detection or hidden surface removal.
- Realistic scenes: closer objects occludes the others.

Classification:

- Object space methods
- Image space methods

Image-Space Method vs. Object-Space Method

Image-Space Method

- Depth-Buffer Method
- A-Buffer Method
- Scan-Line Method
- *Area-Subdivision Method*

Object-Space Method

- Back-Face Detection
- BSP-Tree Method
- Octree Methods
- Ray-Casting Method

Object Space Methods

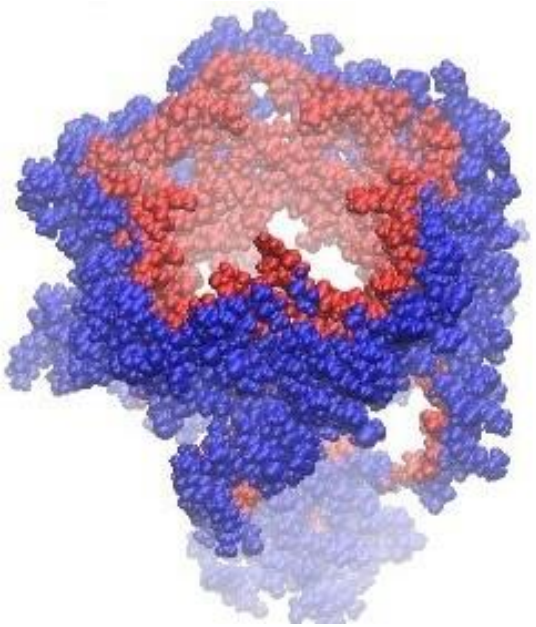
- Algorithms to determine which parts of the shapes are to be rendered in 3D coordinates.
- Methods based on comparison of objects for their 3D positions and dimensions with respect to a viewing position.
- For *N* objects, may require *N***N* comparision operations.
- Efficient for small number of objects but difficult to implement.
- Depth sorting, area subdivision methods.

Image Space Methods

- Based on the pixels to be drawn on 2D. Try to determine which object should contribute to that pixel.
- Running time complexity is the number of pixels times number of objects.
- Space complexity is two times the number of pixels:
  - One array of pixels for the frame buffer One array of
  - pixels for the depth buffer
- Coherence properties of surfaces can be used.
- Depth-buffer and ray casting methods.

Depth Cueing

- Hidden surfaces are not removed but displayed with different effects such as intensity, color, or shadow for giving hint for third dimension of the object.
- Simplest solution: use different colors-intensities based on the dimensions of the shapes.

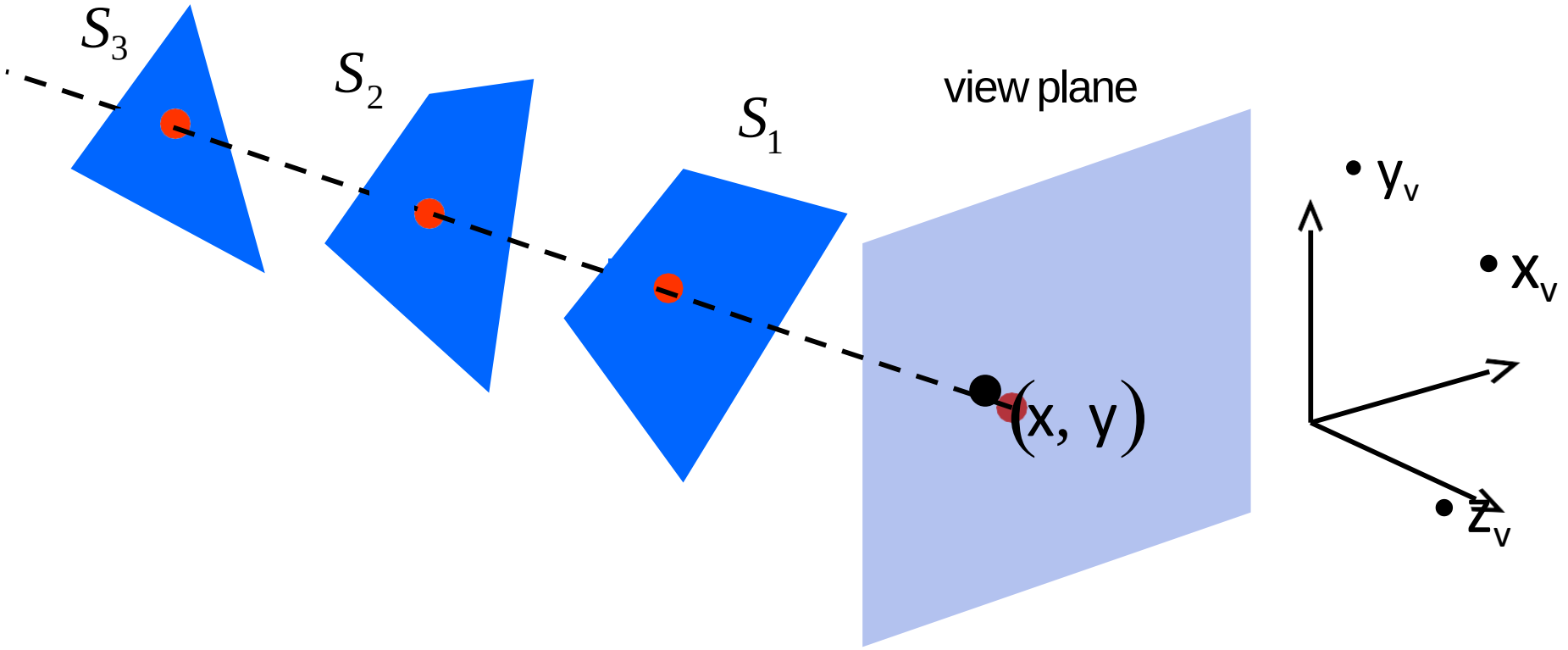


DEPTH-BUFFER ALGORITHM

- Also known as z-buffer algorithm. It is an image space approach.
- Each surface is processed separately one pixel position at a time across the surface.
- The depth values for a pixel are compared and the closest (smallest z) surface determines the color to be displayed in the frame buffer.
- Applied very efficiently on polygon surfaces

  - Surface are processed in any order

DEPTH BUFFER ALORITHM



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

DEPTH BUFFER ALGORITHM

- Two buffers are used
  - Frame Buffer
  - Depth Buffer
- The z-coordinates (depth values) are usually normalized to the range [0,1]

Continue..

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

  depthBuff (x,y) = 0, frameBuff (x,y) = $I_{background}$

  Process each polygon in a scene, one at a time

-
  - For each projected (*x*,*y*) pixel position of a polygon, calculate the depth *z*.
  - If *z* > depthBuff (x,y), compute the surface color at that position and set

  depthBuff (x,y) = z, frameBuff (x,y) = $I_{surfCol}$ (x,y)

Calculating depth values efficiently

- We know the depth values at the vertices. How can we calculate the depth at any other point on the surface of the polygon.
- Using the polygon surface equation:

$$z = \frac{-Ax - By - D}{C}$$

Continue..

- For any scan line adjacent horizontal x positions or vertical y positions differ by 1 unit.
- The depth value of the next position (x+1,y) on the scan line can be obtained using

$$Z' = \frac{-A(x+1) - By - D}{C}$$

$$= z - \frac{A}{C}$$

Continue..

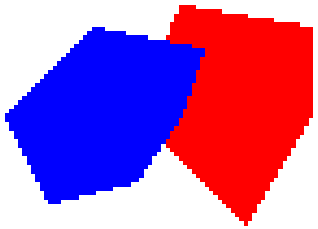
- For adjacent scan-lines we can compute the x value using the slope of the projected line and the previous x value.

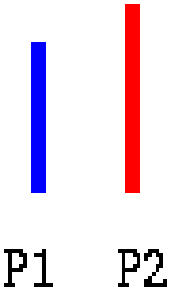
$$x' = x - \frac{1}{m}$$

$$\Rightarrow z' = z + \frac{A/m + B}{C}$$

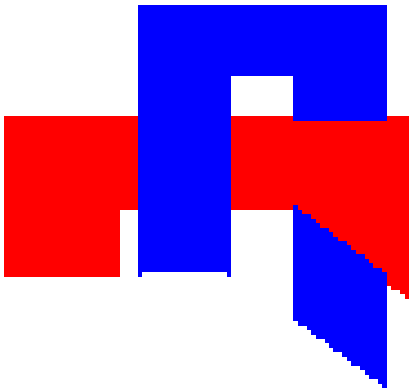
Depth buffer method

- Is able to handle cases such as





View from the Right-side



These polygons are both in front of and behind one another.

THANKING YOU