https://dx.doi.org/10.22161/ijaers.4.2.29 *ISSN: 2349-6495(P) | 2456-1908(O)*

Visible Surface Detection Algorithms: A Review

Nisha

Assistant Professor, Department of computer science, University of Delhi, India

Abstract— *In computer graphics, there are many surface detection algorithms. When we view a picture containing non-transparent objects and surfaces, then we cannot see those objects from view which is behind from objects closer to eye. We must remove these hidden surfaces to get a realistic screen image. The identification and removal of these surfaces is called Hidden-surface problem. In this paper I review some visible surface detection algorithms like Z buffer method, Area subdivision method, Scan line method etc.*

Keywords— Visible surface detection algorithms, Z buffer, Area subdivision, Scan line method etc.

I. INTRODUCTION

In computer graphics there are multiple visible surface detection algorithms or Hidden surface algorithms. A major part of rendering (making images more realistic) is the visible surface problem, i.e. only display those surfaces which should be visible. Hidden surface Algorithms are usually image space or a combination of object and image space. There are two approaches for removing hidden surface problems − **Object-Space method** and **Image-space method**. The Object-space method is implemented in physical coordinate system and image-space method is implemented in screen coordinate system. When we want to display 3D object on a 2D screen, we need to identify those parts of a screen that are visible from a chosen viewing position. Algorithms used in image space for hidden surface removal are much more efficient than object space algorithms. But object space algorithms for hidden surface removal are much more functional than image space algorithms for the same. The combination of these two algorithms gives the best output. There are many visible surface detection algorithms like Z buffer, Painter's algorithm, Area subdivision algorithm etc. [2]

II. Z- BUFFER OR DEPTH BUFFER ALGORITHM

This method is developed by Cutmull. It is an image-space approach. The basic idea is to test the Z-depth of each surface to determine the closest (visible) surface. In this method 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. It is applied very efficiently on surfaces of polygon. Surfaces can be processed in any order. To override the closer polygons from the far ones, two buffers named **frame buffer** and **depth buffer,** are used. **Depth buffer** is used to store depth values for (x, y) position, as surfaces are processed ($0 \leq$ depth ≤ 1). The **frame buffer** is used to store the intensity value of color value at each position (x, y). The z-coordinates are usually normalized to the range [0, 1]. The 0 value for z-coordinate indicates back clipping pane and 1 value for z-coordinates indicates front clipping pane. [3]

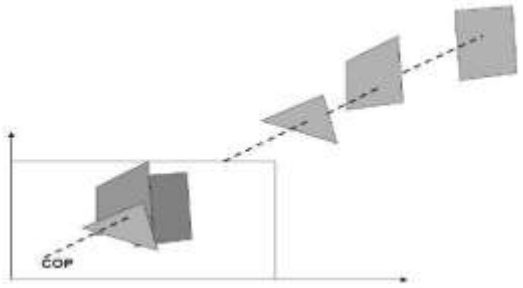

Fig.1: Z buffer Method

This algorithm has some advantages like

- It is easy to implement.
- It reduces the speed problem if implemented in hardware.
- It processes one object at a time.

It has some disadvantages also like

- It requires large memory.
- It is time consuming process.

III. SCAN-LINE METHOD

It is an image-space method to identify visible surface. This method has depth information for only single scan-line. In order to require one scan-line of depth values, we must group and process all polygons intersecting a given scan-line at the same time before processing the next scan-line. Two important tables, **edge table** and **polygon table,** are maintained for this. **The Edge Table** − It contains coordinate endpoints of each line in the scene, the inverse slope of each line, and pointers into the polygon table to

https://dx.doi.org/10.22161/ijaers.4.2.29 *ISSN: 2349-6495(P) | 2456-1908(O)*

connect edges to surfaces. **The Polygon Table** − It contains the plane coefficients, surface material properties, other surface data, and may be pointers to the edge table.

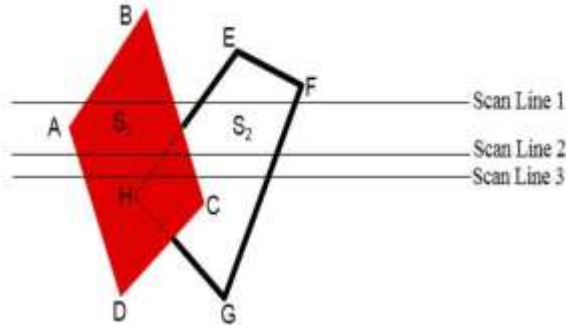

Fig.2: Scan line algorithm

To facilitate the search for surfaces crossing a given scan-line, an active list of edges is formed. The active list stores only those edges that cross the scan-line in order of increasing x. Also a flag is set for each surface to indicate whether a position along a scan-line is either inside or outside the surface. Pixel positions across each scan-line are processed from left to right. At the left intersection with a surface, the surface flag is turned on and at the right, the flag is turned off. You only need to perform depth calculations when multiple surfaces have their flags turned on at a certain scan-line position. [3]

IV. AREA-SUBDIVISION METHOD

The area-subdivision method takes advantage by locating those view areas that represent part of a single surface. Divide the total viewing area into smaller and smaller rectangles until each small area is the projection of part of a single visible surface or no surface at all. Continue this process until the subdivisions are easily analyzed as belonging to a single surface or until they are reduced to the size of a single pixel. An easy way to do this is to successively divide the area into four equal parts at each step. There are four possible relationships that a surface can have with a specified area boundary.

- **Surrounding surface** − One that completely encloses the area.
- **Overlapping surface** − One that is partly inside and partly outside the area.
- **Inside surface** − One that is completely inside the area.
- **Outside surface** − One that is completely outside the area.

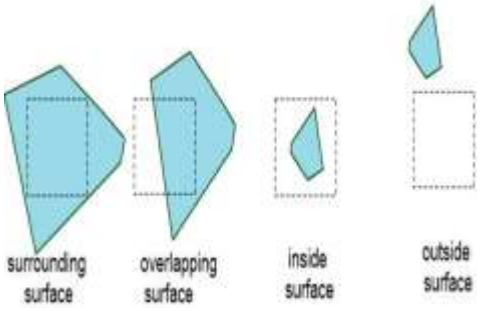

Fig.3: Area subdivision method

The tests for determining surface visibility within an area can be stated in terms of these four classifications. No further subdivisions of a specified area are needed if one of the following conditions is true: [1]

- All surfaces are outside surfaces with respect to the area.
- Only one inside, overlapping or surrounding surface is in the area.
- A surrounding surface obscures all other surfaces within the area boundaries.

V. DEPTH SORTING METHOD

Depth sorting method uses both image space and object-space operations. The depth-sorting method performs two basic functions:

- First, the surfaces are sorted in order of decreasing depth.
- Second, the surfaces are scan-converted in order, starting with the surface of greatest depth.

The scan conversion of the polygon surfaces is performed in image space. This method for solving the hidden-surface problem is often referred to as the **painter's algorithm**. The following figure shows the effect of depth sorting

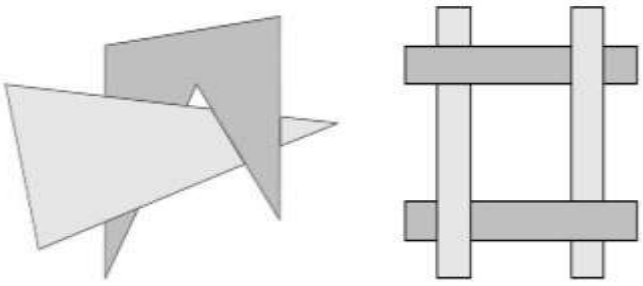

Fig.4: Depth sorting method

The algorithm begins by sorting by depth. For example, the initial “depth” estimate of a polygon may be taken to be the closest z value of any vertex of the polygon.

Let us take the polygon P at the end of the list. Consider all polygons Q whose z-extents overlap P’s. Before drawing P,

https://dx.doi.org/10.22161/ijaers.4.2.29 *ISSN: 2349-6495(P) | 2456-1908(O)*

we make the following tests. If any of the following tests is positive, then we can assume P can be drawn before Q.

- Do the x-extents not overlap?
- Do the y-extents not overlap?
- Is P entirely on the opposite side of Q's plane from the viewpoint?
- Is Q entirely on the same side of P's plane as the viewpoint?
- Do the projections of the polygons not overlap?

If all the tests fail, then we split either P or Q using the plane of the other. The new cut polygons are inserting into the depth order and the process continues. Theoretically, this partitioning could generate $O(n^2)$ individual polygons, but in practice, the number of polygons is much smaller. [1]

VI. BINARY SPACE PARTITION (BSP) TREES

Binary space partitioning is used to calculate visibility. To build the BSP trees, one should start with polygons and label all the edges. Dealing with only one edge at a time, extend each edge so that it splits the plane in two. Place the first edge in the tree as root. Add subsequent edges based on whether they are inside or outside. Edges that span the extension of an edge that is already in the tree are split into two and both are added to the tree. [1]

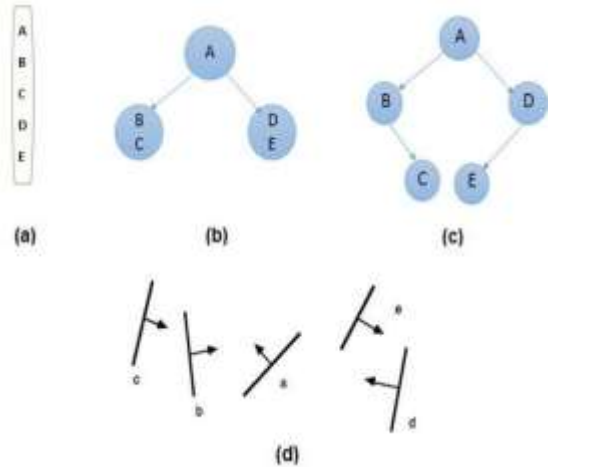

Fig.5: BSP tree method

- From the above figure, first take **A** as a root.
- Make a list of all nodes in figure (a).
- Put all the nodes that are in front of root **A** to the left side of node **A** and put all those nodes that are behind the root **A** to the right side as shown in figure (b).
- Process all the front nodes first and then the nodes at the back.
- As shown in figure (c), we will first process the node **B**. As there is nothing in front of the node **B**, we have put NIL. However, we have node **C** at back of node **B**, so node **C** will go to the right side of node **B**.
- Repeat the same process for the node **D**.

VII. CONCLUSION

In this paper I reviewed different visible surface detection algorithms. Every algorithm has some advantages and disadvantages like Z buffer method, it is easy to implement and processes one object at a time but it requires large memory and time. Scan line method is simpler and easy to implement. In area subdivision four relationships of surface and region is used. Depth sorting method uses both image and objects based approach but in this some test are used to determine the visible surface. Every algorithm has its own advantages and disadvantages but scan line method is very simple and easy to implement.

REFERENCES

[1] https://www.tutorialspoint.com/computer_graphics/visible_surface_detection.htm
[2] http://www.cse.iitm.ac.in/vplab/courses/CG/PDF/VIS_SURF_DET.pdf
[3] Donald Hearn, and M. Pauline Baker, "Computer Graphics, C Version", 3 edition,, December 2004
[4] http://ocw.metu.edu.tr/pluginfile.php/1021/mod_resource/content/0/documents/lecturenotes_2007/week13_VisibleSurfaceDetection.pdf
[5] Rogers DF. "Procedural elements for computer graphics". New York: McGraw-Hill, 1985.