Analysis Of the Principle, Applications and Defects Of Z-Buffer Algorithm

Muhan Xu

Raffles Institution, Singapore, Singapore

25yxumu772g@student.ri.edu.sg

Abstract. Z-buffering, commonly known as depth buffering, is a fundamental algorithm in the realm of computer graphics used to carry out hidden surface removal. This algorithm is classified as an image-space approach, where visibility is determined individually for each pixel on the view plane. Owing to its simplicity, efficiency, and accuracy in rendering complex scenes, the Z-buffer algorithm has been widely employed in real-time graphics renders, e.g., in video games, virtual reality (VR) and 3-dimensional modelling software. This study presents a detailed analysis of the underlying principles of the algorithm, namely, comparing depth values at each pixel to decide which polygon obscures the rest. The next component of the study is dedicated to presenting the state-of-art applications of the Z-buffer algorithm, justifying its immense value add to various industries, such as the video game and animation industries. According to the analysis, this study will conclude with an analysis of the limitations of Z-buffer algorithm, presentencing the challenges and imperfections of the algorithm that afflict its practitioners hitherto, which shed light on guiding further investigations.

Keywords: Enter key words or phrases in alphabetical order, separated by commas.

1. Introduction

In 3D computer graphics, a three-dimensional scene is rendered to produce a two-dimensional image, viewed from a certain perspective. However, not every portion of every 3D polygon would be visible from a particular camera angle. Hidden-surface elimination algorithms are the solutions to visibility problems [1]. Given a set of polygons in a Euclidean space, a polygon, or part thereof is said to be visible, if a line segment that joins it to the camera view does not intersect any other polygon. On the other hand, if a polygon is occluded by another, it is said to be hidden. Hidden-surface elimination is the key to rendering realistic 3D scenes. Yet, it is a challenge to execute accurately. Hidden surface elimination algorithms can be classified into two broad categories: object space method and image space method.

Object space algorithms operate on geometrical primitives. It is implemented on objects in the three-dimensional physical coordinate system, before geometrical transformations such as scaling, rotation or translation are applied. This method compares objects or parts of objects to each other within the scene definition in determining which surfaces are visible. Algorithms such as Back Face Detection and Elimination [2], Painter's Algorithm fall under the object space method. While the Object-Space Method surface elimination might work to eliminate hidden surfaces of simple, regular objects, it fails to process complex polygons and non-opaque objects.

Image Space method [3], on the other hand, operated on pixels. It is a comparatively more sophisticated method, capable of accurately determining visibility of surfaces in a more complex setting, such as one that involves irregular, transparent, or multiple objects in the same frame. This method refers to algorithms that are implemented on two-dimensional screen coordinate systems, where objects have gone through projection and rasterization. Image Space method, visibility is decided point by point, for each individual pixel position on the view planes. A popular algorithm that is classified under the image-space method category is the Z-buffer algorithm.

Z-buffering, also known as depth buffering, is an evolutionary algorithm in the field of computer graphics. Z-buffering involves the use of a Z-buffer, a two-dimensional array that stores the depth of each pixel in screen space. Various techniques had been explored before the invention of the Z-buffering algorithm, including Binary Space Partitioning (BSP) Trees, Warnock's Algorithm and

Painter's Algorithm. However, they faced limitations concerning inaccurate computation of results, inefficiency, and long processing times. In the 1970s, a radical new approach which generates the most accurate on-screen results thus far was proposed, i.e., Z-buffering. The algorithm was originally described Wolfgang Straßer in his PhD thesis on efficient methods of rendering occluded objects and subsequently, furthered by Edwin Catmull. The Z-buffer rasterization algorithm resolves visibility independently at each pixel. ensuring the accurate elimination of hidden surfaces, such that only visible surfaces contribute to the final image presented on-screen. Albeit the tremendous improvement in accuracy of 3D image rasterized using Z-buffer, the algorithm has several drawbacks. Z-buffer algorithms utilizes a significant amount of memory bandwidth to store the additional depth-buffer data. Time taken to process the depth of each polygon hinders the efficiency of the algorithm too.

This study is dedicated to analyzing four main areas relating to the Z-buffer algorithm. First, an analysis of the underlying principles of the Z-buffer algorithm as well as a description of its coding logic. Next, a discussion regarding the state-of-art applications of Z-buffering in computer graphics, computer games and medical imaging would be discussed. Lastly, an assessment of the limitations Z-buffer algorithm faced today will be demonstrated.

2. Principle

To begin, a range of depth values, often denoted as ***z***, is defined to take on a value between a near and far value of ***z***. After a perspective transformation, a nonlinear projection where three-dimensional objects are projected onto a picture plane, the new value range of z in camera space is denoted as ***z'***. In this paper, this study will normalise the resulting values of ***z'*** to be between the values of ***0*** and ***1***, where ***0*** represents the near clipping plane, and ***1*** represents the far clipping plane.

$$z' = \frac{zfar+znear}{zfar-znear} + \frac{1}{z}\left(\frac{-2\cdot zfar\cdot znear}{zfar-znear}\right) \quad (1)$$

After which, on a view plane, an additional 2D array, called the Z-buffer is used alongside the regular colour buffer. The new 2D array stores the depth (Z-coordinate) of the nearest object for each pixel in the screen, effectively creating a depth map, where 1 represents infinite distance and 0 represents zero distance.

To begin the derivation of the depth map, the Z-buffer array, depthBuff is initialized to 1, which represents an infinite distance from front of view. The frameBuff array is initialized to the background color. Subsequently, the Depth Value z of each pixel in the polygon's projection is reviewed and compared against the depthBuff array of the screen. If z< depthBuff [x][y], indicating that the new polygon has a lesser depth, is nearer to the camera view and therefore, obscures the background as well as any polygon that had been previously reviewed. In this situation, the depth depthBuff [x][y] is to be updated to depth of polygon z[i][j], and the color of the pixel frameBuff [x][y] is set to the color of the polygon being reviewed, c[i][j]. The running time complexity for Z-buffer, and other Image-space method algorithms alike, is O (Number of pixels * Number of objects). The space complexity for Z-buffering is 2*x*y (where x is the number of rows of pixels and y is the number of columns of pixels) because two arrays are required, one for frame buffer frameBuff, and the other for depth buffer depthBuff.

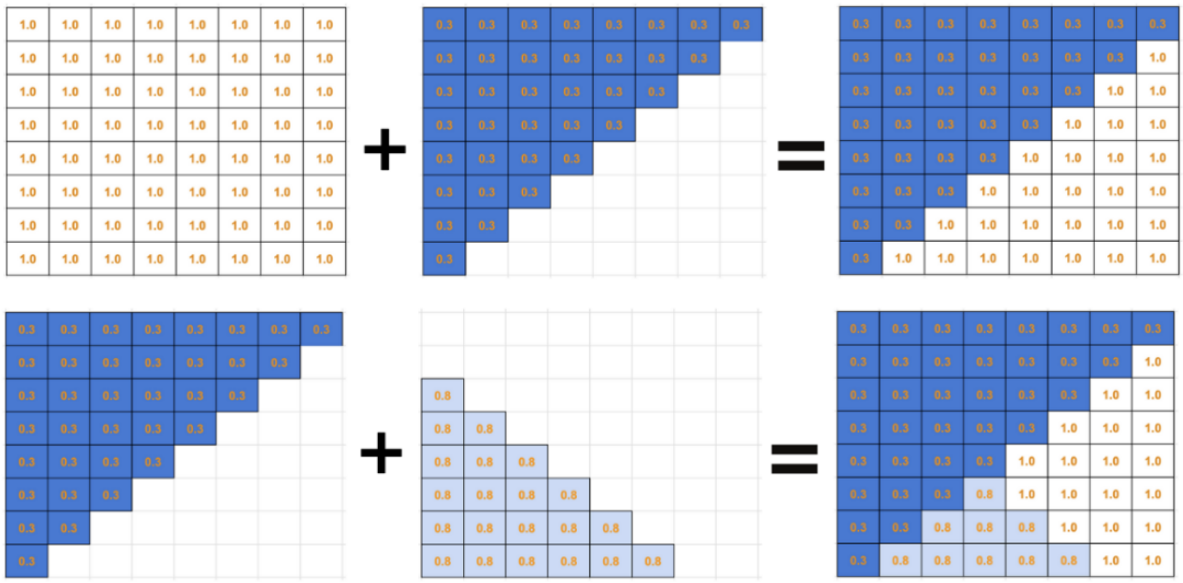

Figure 1. Visualisation of depth buffer array (Picture credit: Original).

3. Applications

3.1. Depth of field

Depth of field refers to the optical phenomenon where objects within a specific distance range in a scene appear sharp and in focus, while objects closer or farther away from this range appear blurred or out of focus. Depth of field proves effective in directing the viewer's attention within the rendered scene, giving them a better sense of depth within a scene. To illustrate the depth of field concept, this study provides an equation to calculating the circle of confusion, an optical “region” caused by a cone of light, instead of a single point, due to rays not coming to a perfect focus when imaging a point source [4-6].

One of the most common uses of the Z-buffer is to create this depth of field effect, aptly applied in rendering scenes in Toy Story 4 [7], as seen in Figure 2. To accomplish this, the depth information of each pixel is used to determine the circle of confusion size for each pixel. The circle of confusion for world-space distance is determined by camera parameters – free to be decided by animators. The formula is as follows:

$$CoC = abs(aperature \cdot \frac{focalLength \cdot (objDist - planeInFocus)}{objDist \cdot (planeInFocus - focalLength)}) \tag{2}$$

Here, CoC refers to the diameter of the circle of confusion; aperture determines how collimated the right rays are set to be in relation to one another [6]. Aperture, focal length, and plane in focus are all camera parameters and remains constant. Object distance on the other hand, is a variable that depends on the distance of the polygon from the camera view. It can be calculated from the Z-buffer values.

$$objDist = \frac{-zfar \cdot znear}{zDepthBuff \cdot (zfar - znear) - zfar} \tag{3}$$

Depth of field is an integral part of cinematography and movie animations. It directs the viewer’s gaze to the object that is focused on, while reducing the distraction caused by surrounding, irrelevant objects. Many cinematographers would agree that depth of field technique can serve as a valuable and nuanced technique during moments of drama, allowing a character to visually detach themselves from a situation or providing a means to deeply immerse the audience into the character's psyche during a critical moment [8].

Figure 2. Depth of field effect in Pixar animation Toy Story 4 [7].

3.2. Creating gog

Depth buffers values serve as a useful tool in rendering graphical environments that employ “fog” gradients. Distance fog, also known as “fog”, gradient is a technique used in 3D computer graphics to enhance the perception of distance through the varied shading of objects, depending on their distance from camera view. It stimulates the effect of light scattering, which causes more distant objects to appear lower in contrast [9]. This is a useful technique in rendering more realistic, real-world scenes. distant objects differently.

With the depth buffer value of each pixel, the distance between a viewpoint and a fragment is computed. This distance value is then utilised to blend the polygon’s colour with a fog colour (seen from Fig. 3) [5, 10]. The further the polygon from the camera view, the greater the magnitude of the depthBuff value of the pixel, the greater the effect of the fog density. Volumetric Fogging effects are employed in numerous PC games, such as Genshin, Silent Hill, Minecraft, to name a few. It creates an ambient effect and increases the realism of rendered scenes. Especially in underground empty hangars depicted in the image above, or inside deep forests, including fog and moisture contributes to the immersive experience of the game.

Figure 3. Fog effects on action-adventure video game, Control [5].

4. Defects and Limitations

4.1. Z-Fighting

Z-fighting, also known as stitching, is a phenomenon that occurs where two or more primitives have extremely similar Z-buffer values, which keeps track of depth. During rendering of a single pixel, there is uncertainty as to which one of the primitives are drawn in that pixel, because the difference between their depth values is indistinguishable. The consequence of Z-fighting is visual flickering, where noisy rasterization of two polygons causes artificing of visuals presented on screen as shown in Fig. 4. This limitation is often a consequence of limited sub-pixel precision. Often, the first solution that comes to mind would be to use a higher resolution depth buffer which is able to distinguish minimal differences in primitive depth. However, an increase in resolution comes at the cost of the amount of memory required.

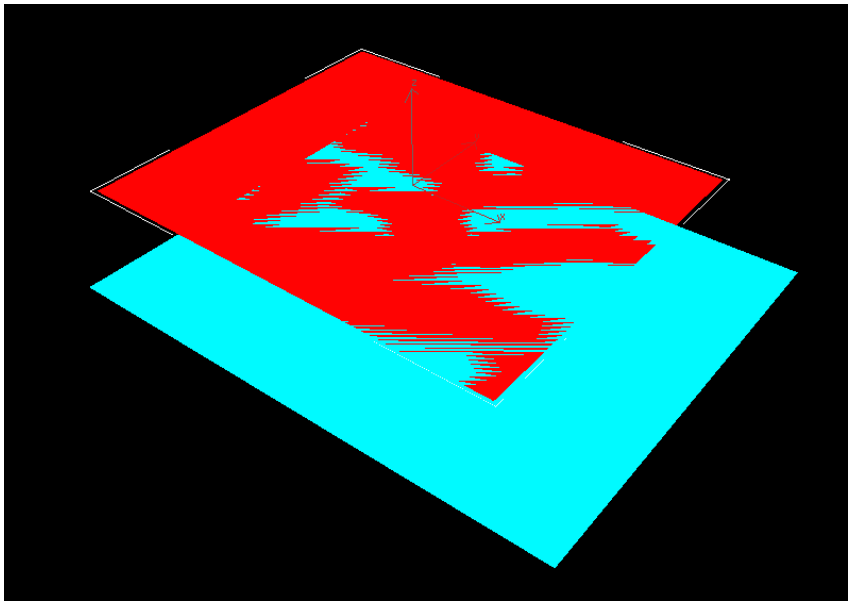

Figure 4. Illustration of Z-fighting between blue and red surfaces (Picture credit: Original)

4.2. Highly Memory Intensive

The second limitation of the Z-buffering algorithm is that it is highly memory intensive, utilizing a significant portion of memory bandwidth, also known as the amount of information, that can be transferred to and from memory per unit time. A Z-buffer requires a non-trivial amount of memory. A typical value in a Z-buffer is a 32-bit floating-point value; a rendered image that is 1280×800 pixels require approximately 3 MB of memory to store its Z-buffer.

4.3. Inaccurate rendering of scenes involving translucent objects

Furthermore, the traditional Z-buffering technique faces challenges when dealing with transparent or translucent objects. The algorithm replaces the frame buffer entirely with the colour value of the primitive with the closest distance to camera view, resulting in the loss of transparent effect of certain materials/objects in the rendered scene. Additional techniques like alpha blending or order-independent transparency must be employed to address this issue [4].

5. Conclusion

To sum up, this study delved into the realm of 3D computer graphics, endeavouring to analyse the underlying principles, coding logic, state-of-art applications, and limitations of the Z-buffer algorithm. A discussion on hidden-surface elimination algorithms, which can be categorised into object space and image space method is provided, with a focus on the Z-buffer algorithm. The Z-buffer algorithm,

a cornerstone in computer graphics, employs a two-dimensional array to store pixel depths, facilitating accurate visibility determination independently at each pixel. Despite the tremendous improvement in accuracy of the Z-buffer algorithm, as compared to previous hidden-surface elimination techniques, it nevertheless faces challenges such as Z-fighting, excessive memory bandwidth usage and processing inefficiencies. Moving forward, prospects for Z-buffer algorithms include the exploration of advanced memory management techniques to mitigate memory bandwidth usage and optimize performance. Additionally, research efforts may focus on developing novel rendering algorithms that address the inherent limitations of Z-buffering, such as Z-fighting or Z-buffer's inabilities to accurately render scenes with translucent objects.

References

[1] Sutherland E E, Sproull R F and Schumacker R A 1974 ACM Computing Surveys vol 6(1) pp 1–55

[2] Jorick van der H 2008 Computing Laboratory University of Oxford pp 18-30.

[3] Paul G A 2018 Hidden Surface Determination School of Computer Science and Engineering vol 2.

[4] Steve Baker 2023 Alpha-blending and the Z-buffer. Retrieved from: https://www.sjbaker.org/steve/omniv/alpha_sorting.html.

[5] Zhou T, Chen J X and Smith P 2007 Proceedings of HCI pp 165-174.

[6] Joe D 2007 Depth of Field: A Survey of Techniques NVIDIA Developer vol 23.

[7] Renée V 2019 How Pixar’s “Real” Fake Cameras Make Their Movies Look So Realistic No Film School p 15.

[8] Cinefade A 2018 How to vary depth of field in film p 1.

[9] Dumont E, Hautière N and Gallen R 2010 Atmospheric Turbulence Meteorological Modelling and Aerodynamics vol 5.

[10] Bontchev B, Vassileva D and Dobrin I 2018 In 16th International Conderence e-society IADIS Press pp 193-200.