Al-Rafidain Engineering Journal (AREJ) Vol.26, No.2, October 2021, pp.300-308

Evaluation of Hidden Surface Removal Methods Using Open GL

Lubna Muzahim Saeed
lubnasaeed81@gmail.com

Fakhrulddin H. Ali
fhazaa@uomosul.edu.iq

* Remote Sensing Center, University of Mosul
** Computer Engineering Department, Collage of Engineering, University of Mosul

Received: 2/2/2021 Accepted: 14/3/2021

ABSTRACT

The research aims to address one of the complex problems that may be encountered when generating computer images for three-dimensional (3D) objects, which is the hidden surface detection and elimination. Several methods have been devised, through the last three decades, to solve this problem. However, the most popular methods used widely nowadays are depth or Z-Buffer (ZB) and Binary Space Partition Tree (BSPT). The first method (ZB) is very simple and more general but requires additional memory to store depth values in addition to intensity. On the other hand, (BSPT) method is more complex to implement but requires memory for pixel intensity only. Modelling graphical database in a binary tree make the dealing with parts of the database feasible. This is so important for clipping a part or more of the database when outside the viewing zone.

The focus of this paper is to study these two methods, design an algorithm to implement each of them. After that a reasonable procedure is to found and applied for testing the performance of each of the two methods using exactly the same graphic load. A step by step increase of this load should be possible so that the behavior of the execution time at different complexity level and on average bases can be reported. The outcome results of the two methods should be compared and a recommendation to use which one of them is to be concluded [1].

Since one of the most popular graphics library which is being used during the last ten years and nowadays is the Open Graphic Library(OpenGL) from(Silicon Graphic Incorporation (SGI)[2], the algorithms of this work are implemented and the related software is carried out using this library with visual C platform.

Keywords:

Hidden Surface. Back Face. Pixel. BSPT. ZB. OpenGL

https://rengj.mosuljournals.com

==========

1. INTRODUCTION

Computer has become a powerful tool for the rapid and economical production of pictures. There is virtually no area in which graphical displays cannot be used to some advantage, and so it is not surprising to find that the use of computer graphics is so widespread [3]. Although early applications in engineering and science had to rely on expensive and cumbersome equipment, advances in computer technology have made interactive computer graphics as a practical tool. Computer graphic have attracted the attention of many researchers in many fields like engineering, medicine , business, industry, government, art, entertainment, advertising , education and training [4] .

Several techniques are used to solve the hidden surface problem which are the painter s' algorithm, Z-buffer, scan line, ray casting, depth sort, and BSPT.

Among the most important of these technologies, BSPT and ZB are of the algorithms to rapidly generate realistic images of 3-D scenes composed of polygons. The BSP algorithm and z-buffer are used for solving the hidden surface problem [6].

OpenGL is a low-level graphic library specification, which makes available to the programmer a major of geometric primitives: points, lines, polygons, images, and bitmaps. It provides a set of commands that allow the specification of geometric objects using the

provided primitives together with commands that control how these objects are rendering into the frame buffer [6]. It provides a powerful primitive set of rendered commands and all higher-level drawing must be done in terms of these commands.

2. Literature Review

In 1980, F . I ., Parke presented the performance analysis and simulation of three multiprocessor ZB architectures. Three architectures have been proposed as approaches to applying many processors, working in parallel, to the task of rapidly creating shaded raster images [7].

In 1983, Michel Gangent introduced an algorithm that deals with hidden surface and line elimination that applies primarily to the generation of indoor perspective views of simple building models. The algorithm is recursive and operates in the objecty, space using extensively the coherence given by the adjacencies of the rooms within a building [8].

In 1993, Andreas Schilling and Wolfgan Straber introduced an algorithm that deals with the hidden surface elimination problem at pixel level. The implementation was divided into three stages in order to apply the pipeline technique to improve the performance to reach around 20M pixel per second [9].

In 1994, M. H. Aljammas introduced a method for modeling 3D solid objects using sub-division of a single cube. Octree is used for modeling and solving the hidden surface problem. The advantages and disadvantages, together with the impact on the major graphics operations, of this method are investigated [10].

In 2000, F. H.Ali and B.MK. Younis presented a simulation for the performance of the depth or ZB method and its advantages and disadvantages. Different techniques to calculate the depth value have been inverstigated. Using polygon size between 400 to1750 pixels, the reported performance range was 5 to 30 seconds required for processing 10 to 230 polgons using PC programmed with C++ Language. However, a considerable time is lost accessing the extra memory required to implement the depth buffer [11]

In 2002 sigal Ar. Et al, presented a modified BSPT for walkthrough environments which is assumed to be efficient in responding to sequences of collision queries. The experimental results on different samples using the standard BSPT with number of faces between 28000 and 54000 required 1.312 to 221.058 seconds [12].

In 2004, Z. E, Younis described an approach for fast image generation of 3D letters using a binary tree. 3D Arabic letter are adopted as a modeling example. The binary tree, programmed using C++, is used to represent samples of letters and then this tree is traced by the implemented software to solve the hidden surface problem. The measured execution time was about 0.25 seconds for a number of 800 polygons [13].

In 2010 Csaba D. Tóth presented that every set of n disjoint line segments in the plane admits a BSP and an auto-partition of size O(nlogn/loglogn). These bounds are the best possible. The height of a BSP is the height of the tree of recursion. It is an important parameter for efficient manipulation of the BSP tree data structure [14].

In 2014 Joydeep Das, Prosenjit Gupta, Subhashis Majumder, proposed a decomposition based Recommendation Algorithm using BSPT. The divided the entire user's space into smaller regions

based on the location, and then applied the recommendation algorithm separately to each of the regions. The proposed system recommends item to a user in a specific region only using the rating data of that particular region. This reduces the quadratic complexity of the CF process as we avoiding the similarity computation over the entire data [15].

In 2019 Yansen Su, Neng Guo, Ye Tian, Xingyi Zhang, demonstrated the effectiveness of the proposed BSPT in solving complex optimization problems, and it is desirable to enhance the tree structure for solving more challenging problems in the future. On one hand, new non-revisiting algorithms can be established by embedding the BSPT in various evolutionary algorithms other than genetic algorithm. On the other hand, to further enhance the population diversity when constructing the BSP tree, more effective strategies can be designed to fine-tune similar solutions for a better balance of the tree structure [16].

In 2019 Vianney Kengne Tchendji, Jerry Lacmou Zeutouo, we propose a parallel algorithm on the CGM model, with p processors, for solving the optimal binary search tree problem (OBST problem), which is a polyadic non-serial dynamic programming problem. Firstly, we propose a dynamic graph model for solving the OBST problem and show that each instance of this problem corresponds to a one-to-all shortest path problem in this graph. Secondly, we propose a CGM parallel algorithm based on our dynamic graph to solve the OBST problem through one-to-all shortest paths in this graph. It uses our new technique of irregular partitioning of the dynamic graph to try to bring a solution to the well-known contradictory objectives of the minimization of the communication time and the load balancing of the

processors in this type of graph. This new CGM algorithm performs better than the previously most efficient solution, which uses regular partitioning of the tasks graph [17].

In 2020 Manuel Eberl, Max W. Haslbeck, Tobias Nipkow, where they a case study of the formal verification and complexity analysis of some famous probabilistic algorithms and data structures in the proof assist ant Isabelle/HOL . Inparticular, we consider the expected number of comparisons in randomised quicksort, the relationship between randomised quicksort and average-case deterministic quicksort, the expected shape of an unbalanced random Binary Search Tree, the randomised binary search trees described by Martínez and Roura, and the expected shape of a randomised treap. The last three have, to our knowledge, not been analysed using a theorem prover before and the last one is of particular interest because it involves continuous distributions [18] [19].

3. Hidden Surface Removal

A major consideration in the generation of 3D realistic images is the identification and removal of the parts of a scene that are not visible from a chosen viewing position. Faces are classified as back faces, which are entirely invisible, and front faces which can be completely visible when they are uncovered. On the other hand, front faces can be covered, partially or entirely, by some part of 3D object. The test for and elimination of a back face is direct using the normal to the plane of the face. The detection and elimination of covered front faces is more complicated and this problem is the one tackled in this article [20]. Many approaches can be taken to solve this problem and numerous algorithms have been devised for the identification of invisible objects for different types of applications. Some methods require more memory, some involve more processing time, and some apply only on special types of objects. The method chosen for a particular application depends on such factors as, the complexity of the scene, type of object whose image is to be displayed, available equipment, and whether static or animated displays are to be generated. The various algorithms are referred to as visible surface detection methods, sometimes these methods are also referred to as hidden surface elimination methods [21]. Visible surface detection algorithms are broadly classified according to whether they deal with object definition directly or with their projected images or with both. These three approach are called object space methods, image space methods and list priority methods. An object space method compares objects and parts of the objects to each other which the scene definition (or in object space) to determine which surface should label as visible or not. In an image space approach this comparison is done after projection at pixel level. List priority approach works partially in the object space and partially in
the image space for solving the visibility problem .

This paper tend to study and measure the performance of the most commonly used methods(ZB and BSPT) for removing hidden surface and compare the results. The designed algorithms and software are to be implemented using OpenGL [22].

Depth Buffer Method (ZB)

A commonly used image space method for detecting visible surface is the depth buffer method. This method (also called z buffer) involves solving the hidden surface problem by storing the depth of the nearest pixel location to the viewer. A frame buffer (pair of intensity and z values) can be used to display images and remove hidden surface by simply scan converting all front faces, in a random order, and updating a pixel value only when the depth of a point that projects on the pixel is less than the depth stored in the buffer. In figure (1), a flowchart of ZB algorithm is given [23].

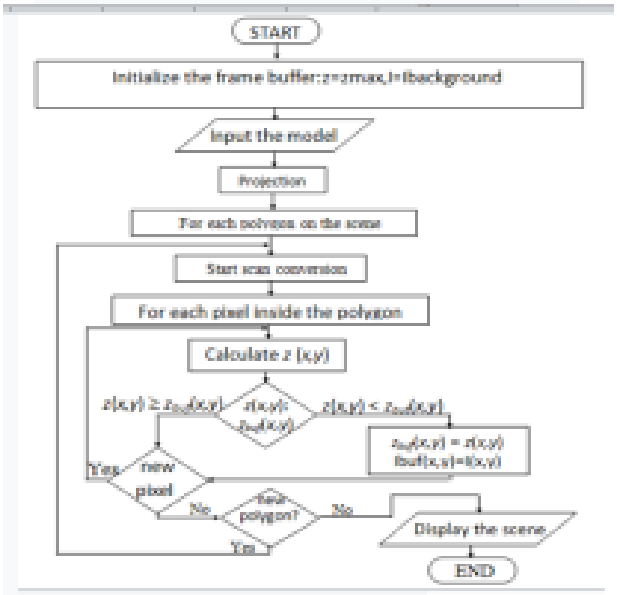


Figure (1) ZB algorithm

uses list priority approach for determining object visibility by painting surface into the frame buffer from back to front as in the painter's algorithm. The idea is to sort the polygons for display in back to front order by selecting planes called

(separating planes) that splits the polygonal object into groups of polygons called (clusters) some of them in front of a separating plane and some are behind. For each of these two subgroups again a plane should be selected to separate each subgroup [24].

This process is After the tree have been created, its traversal can be used to obtain the polygon in back-to-front order when it is visited. Each separating plane can be one of the polygons to be displayed or can be a new recursively repeated until all polygons have been sorted or each cluster is convex. A convex cluster of polygons is simply defined as a group of polygons where each of them can not cover any other polygon in the group. The result can be pictured as a binary tree. At each node of the tree is a separating plane [25]. In one sub tree, or branch, all polygons are in front of this separating plane and in the other branch are those polygons which lie behind it or vice versa . A simple example with a single node is shown in figure (2)

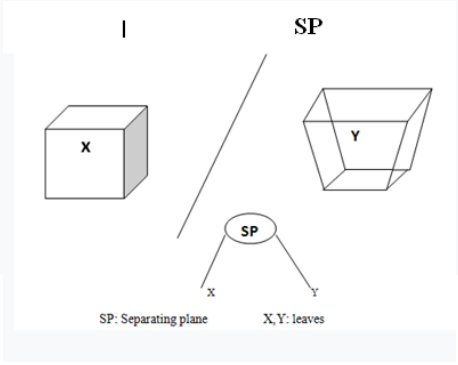
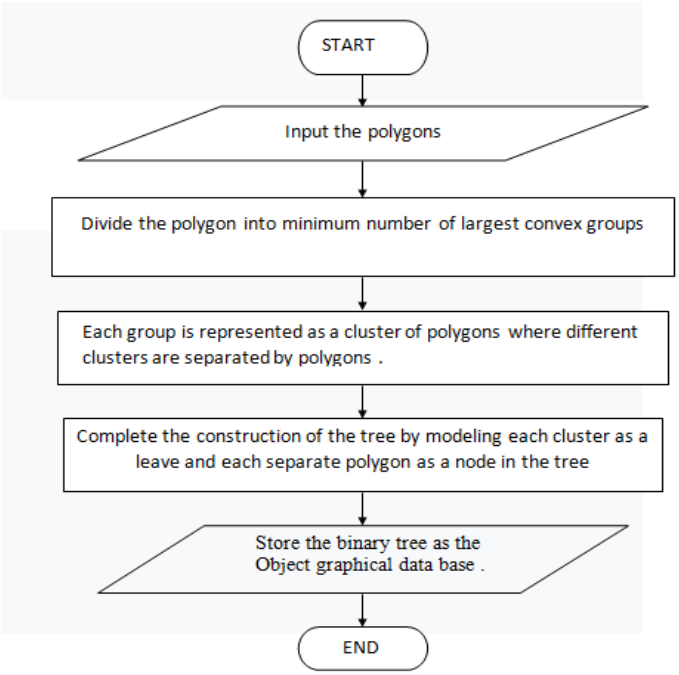


Figure (2) Atree of two cluster()

Figure (3) BSPT algorithm()



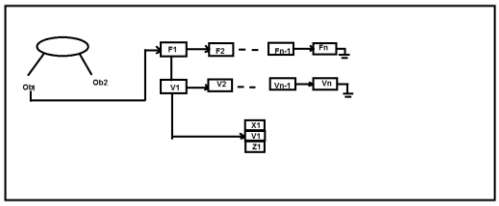
BSPT Implementation Steps

To implement the BSPT algorithm the main step required for the creation of a graphical data base in a form of a binary tree are discussed in the following articles.

Modeling

The polygon boundary representation is used in this work to input data information to the computer about the object in order to display its image. This technique describes a 3D object as a set of surface or polygons that separate the object interior from the environment. Each polygon is represented by its edges, which are in turn defined by their vertices. The polygon is defined by a set of three-dimensional vertices. In addition to the representation of object, the structure organization of the database is very significant in the implementation to collect all information about the object of a scene [27]. The link list and the binary tree are used to construct a structural graphical database. In figure (4) it is

demonstrated, as a simple example, how do this for two object only



F: Face F: Face

Figure (4) Structure database organization

Creating BSPT

To create a BSPT for this model, a list of all polygons is modeled and one polygon is appointed as a splitter plane that is called root polygon or plane. The root plane is used to partition the environment into half-spaces. One half- space contains all polygons in the front space of the root plane, the other contains all polygons behind. The progress of dividing the scene into smaller sets, or the space hierarchical partitioning, is recursively continued until each subset is convex set of polygons [28]. There are several ways to split up the set of polygons into smaller set. For example, an arbitrary plane can be chosen in space, which divides the polygons into two groups assuming the group in the front side of the plane in the right sub tree and the other group, which is in the back side, in the left sub tree. The problem with this approach is that it is sometimes difficult to find a plane that divides the polygons into two equally sized sets, however, non even partitioning creates no problem but only considered as a separating plane, whenever possible. Figure (5) illustrates the first two steps only for how the portioning is accomplished and figure (6) shows the outcome tree.

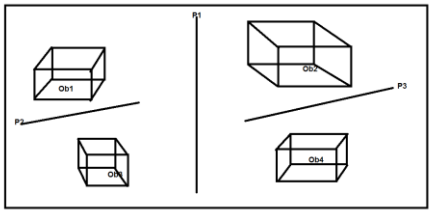


Figure (5) Object space partition of a scene

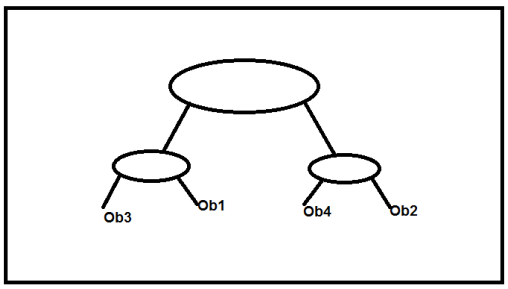


Figure (6) BSPT for the scene in figure (5)

Traversing the BSPT

To draw all the visible faces of a space in a proper order, each node is visited recursively in turn in a certain order and each cluster of polygons (one group of faces) contained in a leaf is rendered after applying the back face test to draw the front faces only. To draw the scene, the root node in the tree is visited first then the back face test on the separating plane in this node is performed**.** According to the outcome of the test, traverse of the left or right branch of this node is done first. This process continues on all other nodes until reaching one of the leaves of the tree and then rendering of the front faces only of that leaf is done. After that the computer returns back to the other passed branches for processing. This continues, recursively, until visiting the whole tree [30]. The order of traversing the tree is determined by the position of the eye relative to the faces representing its separating planes. If the eye position is changed, the same tree can be used to draw the scene and the algorithm is unchanged but the order of visiting the tree branches is dynamically varied and adapted to solve the visibility problem in a correct manner [31].

Performance

One of the major concerns of real time graphics is the speed of execution. The execution time of a graphic algorithm is a function of the complexity of the polygonal graphical database, which is usually measured either with the number of polygons or with number of vertices.

To answer the question which method (ZB or BSPT) performs better, an identical graphical load must be applied. This load should be chosen such that wide variations of its value is feasible so that the performance examination can be carried out through a wide range of complexity level.

The number of polygons (or vertices) is increased in many steps where in each step the execution time of the software is measured. This is necessary to see how this time behaves as a function of image complexity. To increase the graphic load linearly the data base is build from

objects each of them is a cube (with 6 faces or 8 vertices) then the number of these objects is increased starting with six only and going finally toward (147456) cube. So in terms of processed polygons they are increased from 12(16 vertices) to 3072 polygons or faces (4096 vertices) and the execution time is measured. This test or measurement is repeated 1000 times, at different orientations, by rotating the objects in a step-angle of 1 degree and the average speed or (execution time) is recorded. From these measurements, we can determine the behavior of the performance of the two methods, under study, and compare them. The result are sketched in figure (7). These measurements are done using personal computer with a CPU frequency 2GHZ and a cache of 2GB. A sample of the test display is shown in figure (8).

BSPT affords a complete solution to the hidden surface problem. It is very useful structure that is used to create a 3D graphical database. It is used to solve the visibility problem by sorting the polygons, using convex clusters, to be able to be drawn in a correct order. Though it requires a bit complicated work, most of the efforts are done off-line (not in real time) to build the tree. BSPT is view independent binary tree structure that allows view dependent front to back or front back to front traversal. The main task of the BSPT algorithm is arranging the polygons in a correct order to perform dynamic sort.

This means whenever the viewpoint changes position the same tree produces correct order. However, the BSPT is effective when the view reference point changes and the object parts are at fixed position with respect to each other, otherwise the tree requires to be reconfigured. The main advantage of this method, compared to the ZB method, is that no extra memory is required to store depth values. On the other hand, ZB method is very simple and can be adopted for any application, with no problem, but requires extra memory to store the depth or Z values.

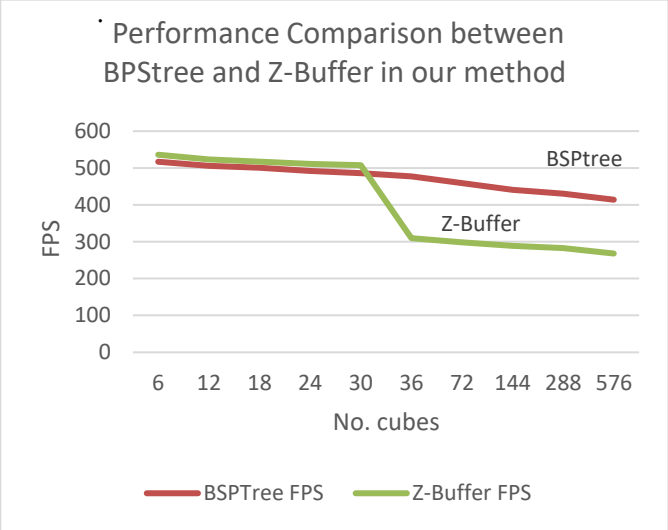
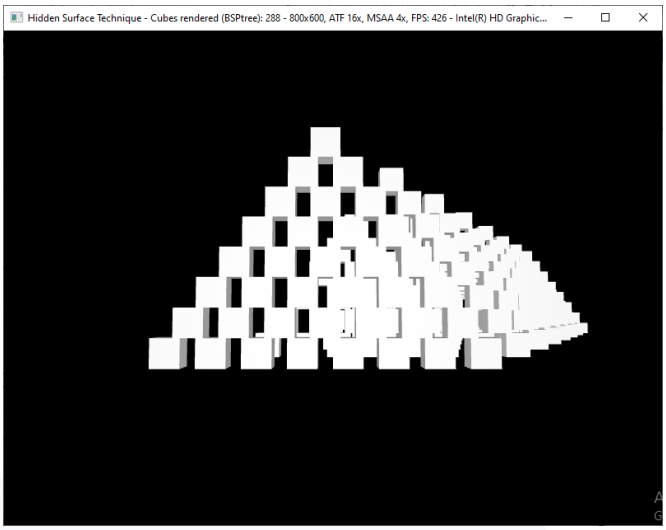


Figure (7) Performance Comparison between BPS tree and Z-Buffer methods



In the binary space partition tree algorithm we can control the proximity or distance of the scene from the camera by changing the Z (depth) value, and we note that the closer the scene is, the fewer

cubes are rendered, so the binary tree hash algorithm is good for close scenes compared to the Z-Buffer algorithm. This is illustrated in Figure (9), where the greater the value of the depth (Z), the less the number of objects (cubes) are rendered, meaning there is an inverse proportion between them.
The scene, whether near or far, the number of objects (cubes) that are displayed remains the same, meaning it does not change, so the Z-Buffer algorithm is ineffective in the near scene, therefore BSPT is preferred for near scenes and Z-Buffer for remote scenes.

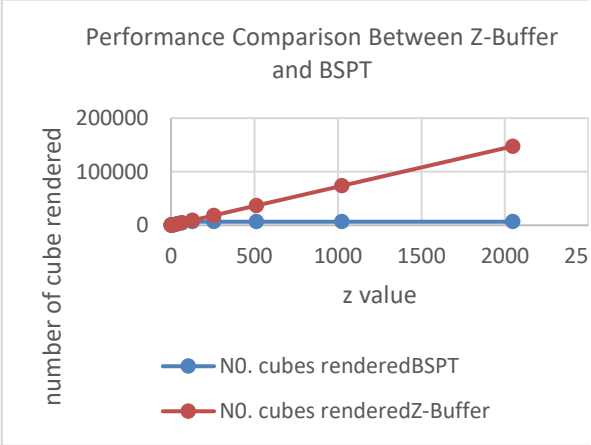


Figure (9) Performance our methods in near and far scene

Conclusions

In this work, the performance of each method, BSPT and ZB, is measured using the same graphical load and both result are drawn at different levels of complexity in figure (9). From these curves, we can conclude the followings:

1- In BSPT method, there is nearly a linear increase in execution time, or decrease in fps with complexity (linear performance). This is because the increase of execution time is mainly due to increasing the number of separating planes where each requires solving the normal to the plane equation. This mathematical overhead increases linearly with complexity.
2- In ZB method, there is a decrease of performance with the complexity. The mathematical load with this method comes from accessing each individual pixel for depth comparison. The total number of pixels for each face is decreased when the number of faces is increased because, due to coherence, faces become smaller when there are more of them in a scene. Due to this fact, there is a decreasing increase of execution time with load leading to the saturation form after the load 36 cubes.
3- With a scene of smaller number of objects, the Binary Space Partition Tree method may take more time as compared to the Z-Buffer method, so the Z-Buffer is recommended for such a situation. On the other hand, for a scene with a high number of objects (more complex) the Binary Space Partition Tree works faster. They have approximately the same speed when the number of objects is around 30. This novel number represents the intersection of the two curves at a number of polygons or faces which is 360 or number of vertices that is 240.
4- The above mentioned number are affected by the performance power of the computer used to execute the software so greater speeds are feasible using more powerful computers.
5- It is also concluded that the use of a cube as 3D object element, for measurement, facilitates a increase of load so it is highly recommended for 3D performance examination and testing.
6- The model of the cube was used in this work for several reasons, the most important of which is that it is a regular shape, easy to handle and clear (consisting of six faces, 12 triangles, and 36 vertices). Rising the complexity, it does the job we want (perfect model).
7- We can control the proximity or distance of the scene from the camera by changing the Z (depth) value, and we note that the closer the scene is, the fewer cubes are rendered.

References:

1- Edward Angle, " Interactive Computer Graphic", A Top-Down Approach Using OpenGL, Third Edition 2003.
2- Mark Segal and Kurt Akeley, Silicon Graphics, Computer System 2011 N. Shoreline Bivd, Mountain View, CA 94039 " The Design of the OpenGL Graphics Interface", 1998.
3- F.S. Hill , Jr., " Computer Graphics Using OpenGL" . Department of

Electrical and Computer Engineering, University of Massachusetts, Prentice Hall 2001, Second Edition.

4- John Kessenich, Dave Baldwin, and Randi Rost, "The OpenGL Shading Language ", April. 2004.
5- Mark Segal . Kurt Akeley, " The OpenGL Graphics System: A specification", Silicon Graphics Inc, 2002.
6- Jonathan Cohen and Nathaniel Duca, University Virginia. David Luebke and Brenden Schubert, Johns Hopkins University. "GLOD: A Geometric Level of Detall System at the OpenGL API Level",2003.
7- Frederic I. Parke , " Simulation and Expected Performance Analysis of Multiple Processor Z-Buffer Systems ", Computer Graphics , Vol. 14 No. 3, July 1980, pp: 48-56.
8- Michel Gangnet, " Depth Sorting by Windowing", Proc of Eurographics Association. 1983,pp: 321-331.
9- Andreas Schilling, Wolfgang Straber," Exact: Algorithm and Hardware Architecture for an Improved A-Buffer", Computer Graphics, vol.27.no.4, August 1993,pp: 85-92.
10- Mohammed H. AL-Jammas ," Algorithms For Display 3D Spild Objects Using Octree", Master thesis, Dept of Electrical Engineering Mosul University,1994.
11- F. H . All and B.MK.Younis,"Simulated Performance Of Z-Buffer For 3D Graphics", Al-Rafidain Engg.Vol.8,No.2,2000.
12- Sigal Ar, Gil Montag and Ayellet Tal, " Deferred, Self-Organizing BSPT Tree", Eurographics, Vol.21,No.3,2002.
13- Zena Ez-aldeen Younis ," Binary Tree For Solving Hidden Surface Problem Of 3D Arabic Letters",Master thesis, Computer Department, Techniqual College in Mosul ,2004.
14- B. Xue, M. Zhang, and W. N. Browne. New fitness functions in binary particle swarm optimisation for feature selection. In Proceedings of the IEEE Congress on Evolutionary Computation, 2012.
15- Y. Sun, G. G. Yen, and Z. Yi. Evolving unsupervised deep neural networks for learning meaningful representations. IEEE Transactions on Evolutionary Computation, 23(1):89–103, 2019.
16- Tchendji VK, Myoupo JF, Dequen G (2016) High performance CGM-based parallel algorithms for the optimal binary search tree problem. Int J Grid High Perform Comput 8(4):55–77
17- Stüwe,D.,Eberl,M.:Probabilisticprimality testing.ArchiveofFormalProofs(2019).http://isa-afp.org/ entries/Probabilistic_Prime_Tests.html, Formal proof development
18- Haslbeck,M.,Eberl,M.,Nipkow,T.:Treaps .ArchiveofFormalProofs(2018).http://isa-afp.org/entries/ Treaps.html, Formal proof development.
19- Today Howard, Department of Computer Science, University of Manchester, " Graphics Programming With OpenGL", January 26th. 2004.
20- Sherif Ghalib Max Planck, Institute for Computer Science, Saarbrucken, Germany," Depth of Field Implementation With OpenGL ", 2006.
21- Tin-Tin Yu, Department of Computer Science, Michigan Technological University, " Depth of Field Implementation With OpenGL ", 1997.
22- Frederic I. Parke , " Simulation and Expected Performance Analysis of Multiple Processor Z-Buffer Systems ", Computer Graphics , Vol. 14 No. 3, July 1980, pp: 48-56.
23- Nisha, N. (2017). Visible Surface Detection Algorithms: A Review. International Journal of Advanced Engineering Research and Science, 4(2).
24- T. Wang, G. Zhang, J. Zhao, Z. He, J. Wang, and M. J. Perez-Jim´ enez.´ Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems. IEEE Transactions on Power Systems, 30(3): 1182–1194, 2015
25- Y. Sun, G. G. Yen, and Z. Yi. Evolving unsupervised deep neural networks for learning meaningful representations. IEEE Transactions on Evolutionary Computation, 23(1):89–103, 2019.
26- Anusha, N., & Bharathi, B. (2019, February). An overview on Change Detection and a Case Study Using Multi-

temporal Satellite Imagery. In 2019 International Conference on Computational Intelligence in Data Science (ICCIDS) (pp. 1-6). IEEE.
27- Tóth, Cs.D.: Binary space partition for axis-aligned fat rectangles. SIAM J. Comput. 38(1), 429–447 (2008).
28- Toby Howard ," An Introduction to Graphics Programming With OpenGL Tutorial and Reference Manual ", Department of Computer Science University of Manchester , at web site www.SIGGRAPH.com , January 26, 2004.
29- David F. Rogers " Mathematical element for computer Graphic " (1997), McGraw-Hill Inc.
30- Ron Goldman " Hidden Surface Algorithm" ,Department of computer science ,Rice University ,2002 .
31- Hershberger, J., Suri, S., Tóth, Cs.D.: Binary space partitions of orthogonal subdivisions. SIAM J. Comput. 34(6), 1380–1397 (2005).

تقييم طرق إزالة الأسطح المخفية باستخدام Open GL

لبنى مزاحم سعيد البك*
lubnasaeed81@gmail.com

فخر الدين حامد علي**
lubnasaeed81@gmail.com

مركز التحسس النائي، جامعة الموصل*

جامعة الموصل - كلية الهندسة - قسم هندسة الحاسوب**

الخلاصة:

ان من أشهر الدوال الحديثة في مجال الرسومات الحاسوبية وتطبيقاتها هي الدوال المكتبية OpenGL وذلك بوصفها أداة تمكن المبرمج من توليد صور الاجسام ثلاثية الابعاد. ونظرا" لما حققته هذه الأداة من انتشار واسع وسريع في الأوساط العلمية ولأهمية هذه الأداة فقد تم تنفيذ بعض العمليات الأساسية التي تتضمنها الرسومات المتولدة باستخدام الحاسوب من خلال هذه المكتبة في هذه الرسالة، لغرض دراسة الأداء لهذه الدوال المكتبية فقد تم اعتماد معالجة أحد اهم المشاكل المعقدة التي تواجه الباحثين في مجال الرسم المجسم باستخدام الحاسوب وهي مشكلة التخلص من الأوجه المخفية. حيث انه هنالك عدة طرق تم استخدامها في السنوات الماضية من اجل حل هذه المشكلة غير ان طريقتين هما فقط استمر استخدامهما حتى الوقت الحاضر الطريقة الأولى هي خوارزمية الشجرة الثنائية والثانية هي طريقة ذاكرة العمق.

ان الطريقتين الأولى والثانية فقد تمت برمجتهما وتنفيذهما بدءا" من الصفر باستخدام مكتبة الرسوم المفتوحة. لغرض دراسة الأداء ومقارنته بالنسبة للطريقتين حيث تم اعتماد حمل متزايد بشكل خطي وبالتدريج وقياس الأداء عند نقاط محددة. حيث تم اختيار الحمل المستخدم مره مكعب ومره اخرى هرم وزيادة هذا الحمل بشكل خطي وبالتدريج وقياس الأداء عند قيم معينة.

اما المسار الاخر الذي تم اعتماده لغرض اجراء الدراسة هو تنفيذ خوارزمية ذاكرة العمق وخوارزمية التجزئة للشجرة الثنائية بوجود وحدة المعالجة الرسومية (Graphic Processing Unit) وقياس الأداء عند نقاط محددة، ومن ثم المقارنة بين الأداءين بوجود CPU وبوجودGPU .

كما تطرقت الرسالة الى دراسة أداء خوارزمية التجزئة للشجرة الثنائية وخوارزمية ذاكرة العمق في المشاهد القريبة والبعيدة.

وأخيرا" تم اقتراح تبني فكرة الأسلوب التكيفي في تطبيقات المحاكاة (أجهزة التدريب الإلكترونية) وفي صناعة الألعاب الإلكترونية والسينما وغيرها من الأغراض. بالإضافة الى اقتراح اختيار عنصر نمذجة جديد مناسب لتنفيذ خوارزمية التجزئة للشجرة الثنائية وخوارزمية ذاكرة العمق للحصول على أفضل أداء ممكن.

الكلمات الداله :

السطح المخفي ، السطح الخلفي ، نقاط ، التقسيم الثنائي للشجرة ، مكتبة الرسوم المفتوحة.