International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 8 (2019), pp. 1313-1320

A Novel Mesh Simplification Method Based on Vertex Removal Using Surface Angle

Hongle Li

Department of Computer Engineering, Keimyung University, 1095, Dalgubeol-daero, Dalseo-gu, Daegu, Republic of Korea.

ORCID: 0000-0001-7857-1020

SeongKi Kim

Department of Computer Engineering, Keimyung University, 1095, Dalgubeol-daero, Dalseo-gu, Daegu, Republic of Korea.

ORCID: 0000-0002-2664-3632

Abstract:

With the development of computer technology, a large number of complex and diverse 3D models have been produced. Faced with these models, how to choose a model suitable for their own applications has become a new knowledge. In this paper, we propose a new research issue for the related problems of the triangular mesh model simplification and design a corresponding solution. The new design proposed in this paper is based on the angle of the surface and achieves the goal of simplifying the model through vertex removal and mesh reconstruction. In this paper, we divide the simplified process of the triangular mesh into three phases: vertex query, vertex removal, and mesh reconstruction. Data reduction is performed by deleting vertex data that is insignificant in the original model, and finally, the hole repairing algorithm is used to complete the simplification of the original triangular mesh model. In this paper, we not only implemented this design but also designed the corresponding test process. The realization of the data fully proves the superiority of our proposal and algorithm and achieves the purpose of quickly streamlining the original mesh model data while maximally maintaining the geometry of the original model. Our algorithm is easy to understand, and can effectively improve the rendering speed of the computer.

Keywords - Foveated Rendering, Mesh reconstruction, Mesh Simplification, Surface angle, Triangular mesh

I. INTRODUCTION

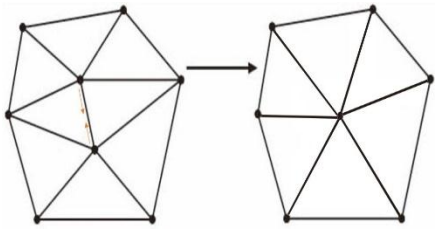
In recent years, with the rapid development of 3D scanning technology and computer graphics technology, the triangular mesh model generated and used has become more and more diverse, and the details are becoming more and more abundant. At the same time, people have higher requirements for the realism of the simulated environment and graphical interface presented by the computer. By using a detailed model to improve the rendering speed of the computer, the computer can often present a higher level of realism, however, the richness of the details often means that the computer needs to deal with more complex and diverse data volume, so it will definitely have a certain impact on the processing speed of the computer and the real-time rendering speed. The processing time and storage cost of mesh data are proportional to the number of triangular meshes included in the model, so it is especially important to get a good trade-off between the number of vertices in the mesh model and computer processing. If you only want to achieve a higher sense of realism, it is often unnecessary to use high-complexity models in all application areas or in different application scenarios in the same field. Therefore, how to effectively simplify the meshes under the premise of keeping the original geometry of the model basically unchanged, so as to improve the processing, transmission and drawing speed of the computer has become the focus of our research.

After nearly 50 years of research and development by many scholars, a series of representative algorithm ideas and technical theories have been produced, including quadratic error metric (QEM)[1], level of detail (LOD)[2], progressive grid and so on. Combined with the research results in this field, according to the characteristics of simplification, the mesh simplification algorithm can be roughly divided into the following categories: adaptive subdivision algorithm, geometric element algorithm, and sampling simplification algorithm. Among them, quadratic error measure (QEM), as a common model simplification algorithm, is one of the geometric element methods. However, the feature of details of the model is not enough, especially when the model is greatly simplified, the contour feature of the model will be seriously lost. But the simplification method using the level of detail often has the problem of large memory utilization and increased CPU computing load. In view of these problems, in this paper, we propose a new mesh simplification algorithm based on surface angle.

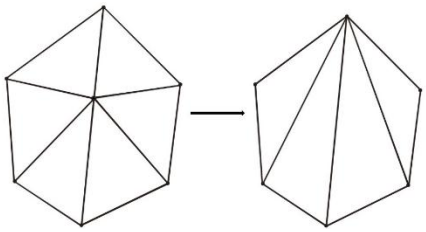
The mesh simplification algorithm based on the surface angle proposed in this paper belongs to the geometry removal method in the geometric element method, but it is different from the existing deletion method of geometric element. The existing deletion method often has high algorithm complexity and logical complexity. The algorithm implemented according to the idea of this paper has the characteristics of easy understanding, fast simplification, low complexity, and perfect simplification of the triangle. Especially for the holes formed after triangle vertices are deleted, the re-triangulation algorithm proposed in this paper can be used to realize the fast triangularization of holes, and at the same time improve the speed of graphics rendering, achieving the expected effect, and making some contributions to the simplification of the triangular meshes.

II. RELATED WORK

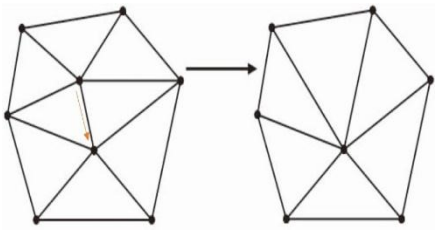
The simplification algorithms can be classified according to features. However, the essence of the mesh simplification algorithm is to minimize the number of triangles inside the mesh model and the number of vertices of the original model, while ensuring the original model geometry as much as possible [3]. At present, the mainstream research method is to simplify the model by the deletion method of geometric elements [4]. The deletion methods include vertex clustering method, vertex deletion method, iterative edge contraction method, and triangle contraction method. As shown in Fig 1:



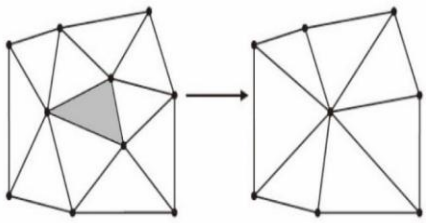
a. Vertex clustering method



b. Vertex removal method



c. Iterative edge contraction method



d. Triangle contraction method

Fig 1. Mesh simplification based on geometric elements

1) **Vertex clustering method:** As the name suggests, two or more vertices within a mesh model are aggregated. This kind of method is represented by the algorithm proposed by Rossignac et al. in 1993 [5]. The central idea of the algorithm is to first surround the original model with a bounding box, and then divide the bounding box into several regions. Then the vertices of the original model will fall into these subdivided regions, and then the vertices in the same region will be aggregated into one vertex. In this way, the vertices in the original model are reduced to the greatest extent without changing the basic shape of the model, thereby reducing the number of triangles and achieving the goal of simplifying the mesh.

2) **Vertex removal method:** this method is to point to in triangle mesh grid model, if the model of a vertex and its surrounding other coplanar, and it doesn't matter the vertices for geometry model, then the vertex can be deleted, but after vertex is deleted, in the current vertex positions will form a hole, then carry on the triangle repair to the current hole, can achieve the goal of mesh simplification.

   Schroeder et al. proposed an algorithm based on vertex removal in 1992 [6]. In this algorithm, their solution is to pass the mesh multiple times, and each vertex is likely to be deleted during the mesh transfer process. If the current vertex meets certain conditions, the vertex can be deleted. The standard here is the distance from the vertex to the plane or edge. If the distance between the vertex and the plane is less than the specified value, the vertex can be deleted; otherwise, it will be retained. But the biggest disadvantage of doing this is that the original geometry of the model will change significantly. This is because in 3D space, if only the distance from the vertex to the plane or edge is used as the evaluation criterion, and ignoring the angles between the vertices, especially in 3D models with sharp edges, the set shape of the original model is often greatly altered. Secondly, the simplified proportion of the model in this algorithm is usually difficult to grasp, because the algorithm uses the distance from the vertex to the edge of the plane as the evaluation criterion. However, different models may require different distances as the threshold. In particular, it is related to the degree of refinement of the grid model. So for the above two problems, we need more auxiliary conditions to overcome this problem.

3) **Iterative edge contraction method:** This method is much better to understand, the method is to shrink the two endpoints of the selected edge into a new vertex in the calculation process of each iteration. Its classical algorithm is represented by Michael Garland et al. in 1997 [1]. The core idea of QEM is to first define a variable that describes the cost of edge shrinking, and then maintain a heap by calculating a cost for each edge of the entire mesh model. Each iteration removes the least cost side of

the heap and then recalculates the lost values for each existing edge to maintain the heap until the given simplification rate is reached. Since only the cost of edge contraction is considered in this algorithm, Li Shijuan et al. in their paper [7], through the design method, proved that the mesh model obtained by QEM algorithm has lower performance in terms of quality. The QEM algorithm produces a lot of long and narrow triangles, so it can be said that the algorithm is not sufficient to maintain the detailed features of the original model.

4) **Triangle contraction method:** Similar to the basic idea of the iterative edge contraction method, the triangle contraction method shrinks the triangle region of the original model into a new vertex in each calculation process, thereby achieving the purpose of simplifying the mesh. In the same situation, simplifying the mesh by the triangle contraction method is much faster than the iterative edge contraction method.

In this paper, after studying the current many algorithms, combining the characteristics of each algorithm, and analyzing the advantages and disadvantages of the algorithm, a new vertex deletion algorithm based on a surface angle is proposed, which effectively avoids the above problems. At the same time, the simplicity of the algorithm is greatly reduced, and the simplification of the triangular mesh model is maximized within a limited range.

III. ALGORITHM DESCRIPTION

The triangle simplification algorithm based on the surface angle proposed in this paper mainly calculates the angle relationship between the triangular faces associated with the vertices. Traversing all model vertices, if the number of angles between the triangle faces associated with the vertex reaches our defined threshold, the vertex is marked as the vertex that should be removed, and the vertex coordinates of the original data set, modified so as to remove the position information of the vertices. The vertices are removed, then the other associated with the vertices were sorted, according to the spatial symmetry principle of triangulation, the realization of hole repairing, so as to achieve the mesh model simplification. Therefore, the algorithm proposed in this paper is mainly divided into the following three steps:

1) Traverse the included angles between all triangular surfaces in the model
2) Evaluate and adjust the data of a certain point according to the threshold
3) Triangulation and repair the holes generated after data adjustment

A. Traverse the included angles between all triangular surfaces in the model

Traversing the angle between all the triangle surfaces in the model is the first step of the algorithm. First, we need to load the 3D model into the program, and separately read the vertex coordinate and the vertex index contained in the mesh model. In the container of the content, the three-dimensional coordinate information of the corresponding vertex is then queried according to the vertex index data and saved. Then, in the order of the vertex numbers in the vertex index, the index data of all the vertices related to the currently selected vertex is extracted into the memory container, and the relevant 3D coordinate information is found correspondingly through the index data stored in the container. According to the information obtained by querying, we can get the normal information of the relevant plane. Finally, by calculating the two pairs of normals, we can get the angle between the planes and store them separately.

For the calculation of the angle between the triangular faces in the mesh model, it can be simplified into the calculation of the angle between the simple faces. Although the calculation methods are various, in this paper we use the simple method. The calculation method is that the two orthogonal vectors on the plane are used to find the normal vector information perpendicular to the two vectors, and finally the angle between the planes is calculated by the angle between the normal vectors of the two planes. Since all the planes contained in our grid model are triangular planes, the three vertices of the triangular plane are $P_1(x_1, y_1, z_1)$, $P_2(x_2, y_2, z_2)$, $P_3(x_3, y_3, z_3)$, and the normal line information is set as (d_x, d_y, d_z), then the normal line satisfies the following Eq. 1:

$$\begin{aligned}(x_2 - x_1) * d_x + (y_2 - y_1) * d_y + (z_2 - z_1) * d_z &= 0\\ (x_3 - x_1) * d_x + (y_3 - y_1) * d_y + (z_3 - z_1) * d_z &= 0 \qquad (1)\\ (x_3 - x_2) * d_x + (y_3 - y_2) * d_y + (z_3 - z_2) * d_z &= 0\end{aligned}$$

According to Eqs. 1 and 2, we can get the value of the normal vector (d_x, d_y, d_z) the current triangular plane, Finally, the Eq. 2 can be calculated by the vector angle, and the angle of the normal angle can be obtained (as shown in Fig 2), and then the angle between the two adjacent triangular faces can be obtained.

$$\cos\theta = \frac{x_1x_2 + y_1y_2 + z_1z_2}{\sqrt{x_1^2+y_1^2+z_1^2} * \sqrt{x_2^2+y_2^2+z_2^2}} \qquad (2)$$

After that, we can save the degree of Angle between all triangular surfaces related to the current vertex, and then enter the second step to evaluate the vertex according to the degree.

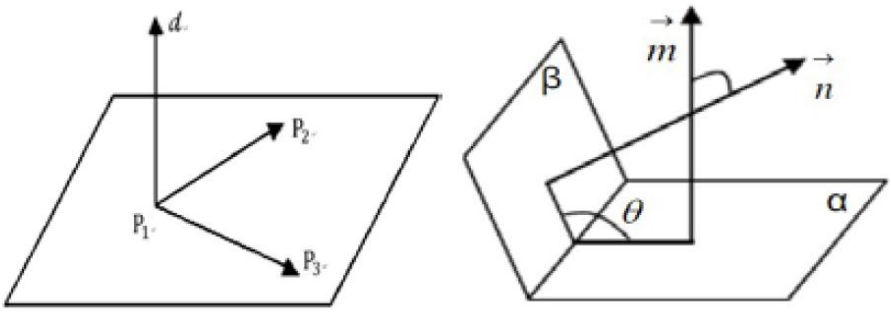


Fig 2. Normal calculation and angle

B. Evaluate and adjust the data of a certain point according to the threshold

In the previous step, we have calculated the number of angles between the triangle faces based on the angle between the plane normal vectors. Next, we compare against the given threshold to determine if the vertex should be retained or removed.

It is known that the degree of refinement of the simplified 3D mesh model varies according to the given threshold. The smaller the included Angle of the surface is, the sharper the geometric shape is. Therefore, theoretically, when the threshold degree is smaller, more vertices should be removed, and the mesh of the generated simplified model should be sparse, and the geometric shape of the model should change the most. Since our goal is to simplify the mesh while maintaining the geometric shape of the model to the maximum, we also need to limit the simplification process of the mesh with the second condition, that is the number of angles meeting the criteria.

When determining whether vertex can be deleted, we adopted the way of traversing angles, as mentioned above, we added a second decision condition - in line with the number of angle determination conditions in the process of mesh simplification, conform to the conditions to determine the angle of the number can be set according to their own needs, but as a result of our premise condition is in the basic don't change the original model under the geometric shape of mesh simplification, so here, we will control the second limitation of the process of mesh simplification conditions: the angle of the eligible number option is set to "all". That is if all the included angles in the container are greater than the given threshold, mark the vertex as a point that can be deleted, otherwise keep the vertex. In this way, while marking the vertices according to angles, the geometric shape of the model can be maintained to the greatest extent.

Since we finally need to evaluate the algorithm, we set the limit of the degree to four different levels as the simplified criteria for the model, provided that the second constraint is set to "all". They are greater than 30°, 60°, 120°, and 150°, and tested for different constraints during the test phase. Finally, after determining the vertex index to be removed, the index is deleted in the container where the original index information is kept, and the coordinate information of the vertex corresponding to the index number is set to empty in the original vertex coordinate.

C. Triangulation and repair the holes generated after data adjustment

In the second step, we completed the determination and modification of the original data of vertex and remove the grid model, when a vertex is deleted, in the current position can form a hole (as shown in Fig 1-b), so at this stage mainly for generated holes to triangle, to fill the hole.

This process seems easy, but the actual process of triangulation is problematic, because the vertices, whether in a 2D plane or 3D space, can be easily distinguished and triangulated to the naked eye. But in the program, how to do it without overlapping, not intersecting and completely triangulating, actually tests the robustness of the algorithm. In the paper, we propose a new algorithm that is different from Schroeder's proposed algorithm [6], which is different from [8], and our algorithm is simpler and easier to understand.

For the process of hole triangulation, our algorithm is pushed from the 2D plane to the 3D space. The idea of our proposed algorithm is as follows. You can imagine that for the holes in the 2D plane (i.e. the sequence of points, as shown in figure 3), we can actually find a line ℓ that crosses one of the vertices v_1 and separate the points on both sides of line ℓ. At this time, starting from the vertex v_1, one vertex (vertex v_2 and vertex v_7) is taken on the left and right sides of the straight line, and the first triangle is triangulated. Then, with the vertex v_2 and the vertex v_7 as edges, the other side of the straight line is taken to obtain the vertex v_3, and the vertex v_2, v_7, v_3 is triangulated. And so on, until the last vertex. This completes the triangulation process of 2D planar holes.

This method is extended to three-dimensional space, as shown in Fig 3. Suppose that the hole formed by deleting the vertex is made up of the six vertices v_1, v_2, v_3, v_4, v_5, and v_6. Since these are points in the space, we can sort the vertex data according to the values of the three dimensions of the vertex coordinates (x value is the main sorting condition, y and z value is the secondary sorting condition). In this way, we actually arrange these vertices in 3D space on both sides of the diagonal ℓ_1 of 3D space. In a similar way to the 2D plane, take the v_1 vertex with the largest coordinate value as the center and triangulate the points on the hole according to the

order of coordinate size. That is, if the ordered vertex order is $v_1, v_3, v_5, v_4, v_2, v_6$, then our triangulation process is (v_1, v_3, v_5), (v_1, v_5, v_4), (v_1, v_4, v_2)..... By analogy, this method can be used to repair the holes. In this way, we have completed the triangulation repair of 3D space holes. (Since it is a 3D space, we also need to triangulate the selected vertex v_1 with the vertex v_6 and the last vertex v_3 so that the holes are completely filled.)

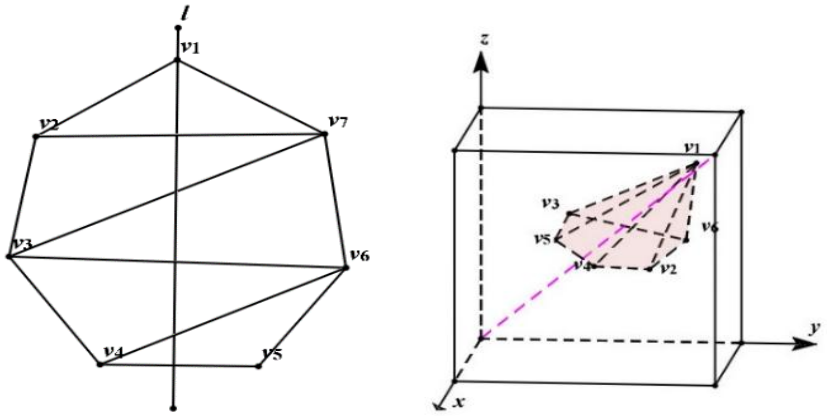


Fig 3. Hole stitching in 2D plane and 3D spaces

IV. RESULTS AND DISCUSSIONS

According to the described design in Section 3, under the premise that the number of eligible angles is set to "all", we divide the evaluation criteria of the model into four different levels and complete the preparation of the program. In the test and evaluation process, we compare the number of primitives in the triangular mesh model, the change in the number of vertices, and the time and speed of the newly generated simplified model in rendering and the algorithm were evaluated accordingly.

Our test was carried out in the Windows 10 x64 version, the hardware configuration is equipped with the Intel Core i7-8700K CPU, 32GB of memory and an NVIDIA GeForce GTX 1080Ti graphics card. In the choice of the model, we mainly chose the mesh model of the monkey's head. Its original mesh model has 1521 vertices and 2904 triangular mesh combinations. After the test starts, we first render the original model and the simplified model generated by using our algorithm separately (the rendering effect of the original model is shown in Fig 4), at the same time, the number of vertices before and after grid simplification, the number of triangle faces, and the time-consuming information of 100 times between the 100th and 200th renderings in the rendering process are recorded separately, and the averaged elapsed time of rendering is measured (As shown in Table 1). Finally, the superiority of the design is strongly confirmed by comparing the processed model with the original model.

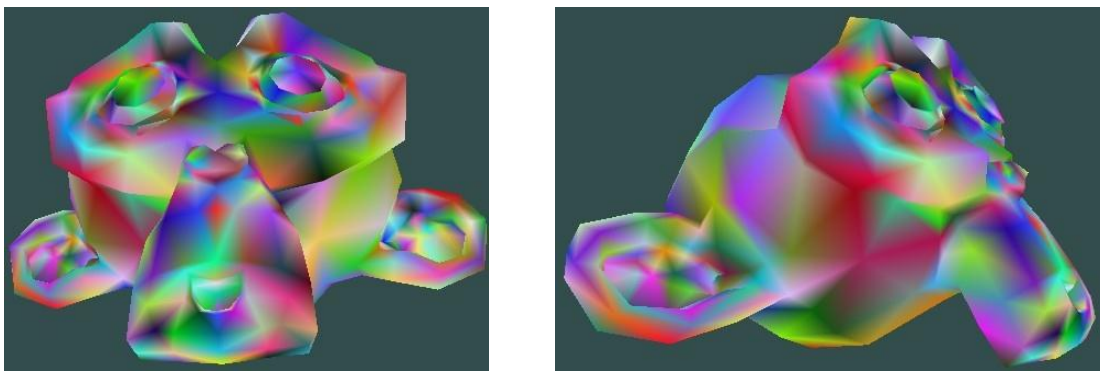


Fig 4. Positive-side rendering effect of the original mesh model

A. Simplified result of surface angle greater than 30°

Under the condition that the Angle between faces is greater than 30 degrees, the original 1521 valid vertices are reduced to 1329 by our algorithm, and the number of triangular meshes is reduced from 2904 to 2562, and 342 triangular faces are simplified. Using the newly generated grid model for rendering tests, we measured an average rendering time of 15.894 milliseconds, which improved the overall rendering speed by 0.441 milliseconds compared to 16.335 milliseconds (as shown in table 1) of the original model. The specific rendering effect under this condition is shown in Fig 5 below:

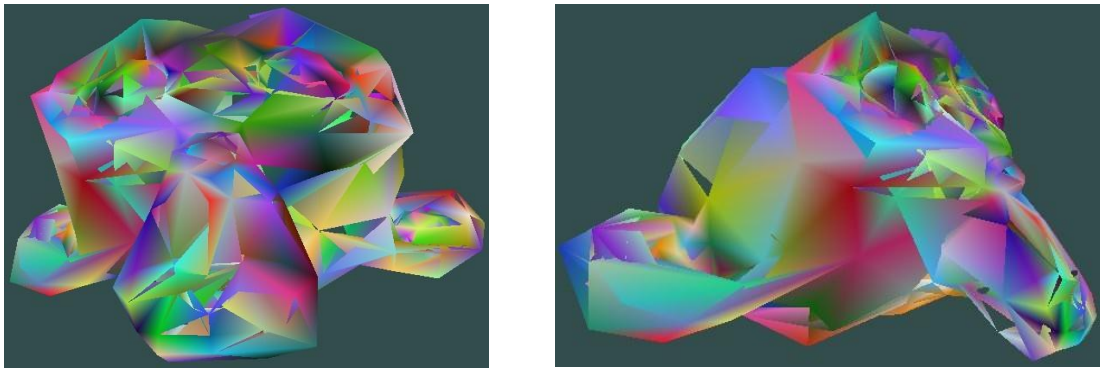


Fig 5. Positive-side rendering effect of treated with 30° model

B. Simplified result of surface angle greater than 60°

In the rendering test where the Angle between faces is greater than 60°, the number of valid vertices is reduced to 1373, and the number of triangles is reduced to 2574, which is 330 less than the number of triangles in the original model. As the number of reduced triangles is not as large as that of 30°, the rendering time is slightly longer than that of 30° (time: 15.945 milliseconds), but it has certain advantages over the original model, as shown in table 1. In terms of rendering effect, the number of triangles in the eyes and ears of the monkey head has been greatly changed, but there is no significant difference in the geometric shape of the model as a whole. The result is shown in Fig 6 below.

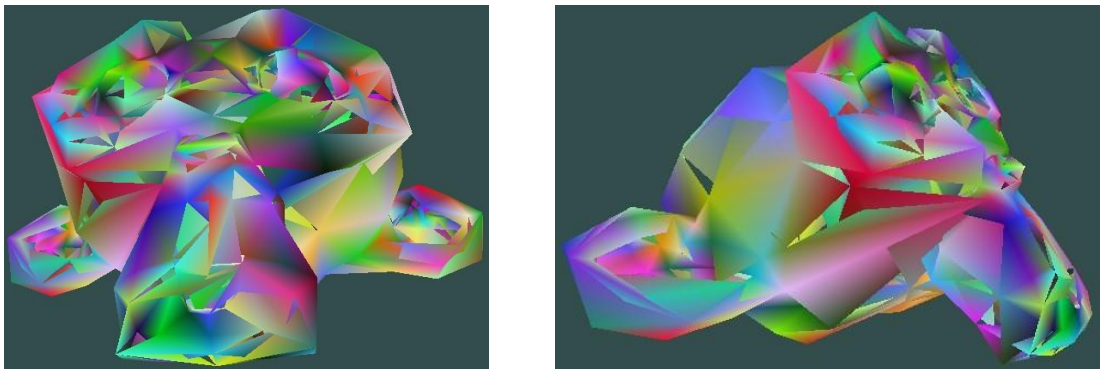


Fig 6. Positive-side rendering effect of treated with 60° model

C. Simplified result of surface angle greater than 120°

In tests with angles greater than 120°, the simplified data changes are small, which means that in this model, the angles of the triangular faces associated with a vertex are all greater than 120°. In this model, only 213 triangles have been reduced, becoming 2691 triangles, and the number of valid vertices has become 1430, indicating that only 91 vertices meet the current conditions. In terms of the average time-consuming measurement, as shown in Table 1, the average time consumption has become 16.139 milliseconds, and the rendering speed has increased by 0.196 milliseconds compared to the original model. In terms of intuitive presentation, the main changes occurred in the mouth and side of the monkey's head, as shown in Fig 7.

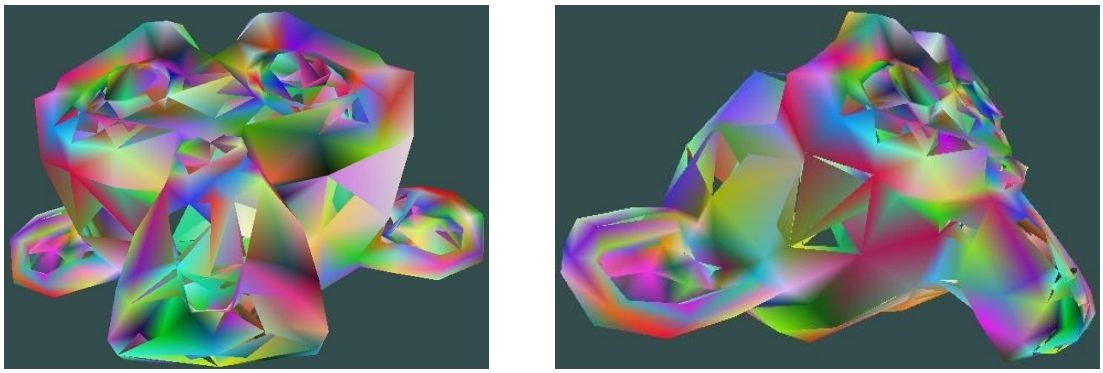


Fig 7. Positive-side rendering effect of treated with 120° model

D. Simplified result of surface angle greater than 150°

In the final phase of the test, we used the face angle greater than 150° as the threshold. Because in the current model, there are fewer vertices with a face angle greater than 150°, so in the test of it, 35 vertices are simplified, and the total number of vertices becomes 1486. The number of triangles has also been reduced by 73, which has become 2832. In terms of rendering time-consuming testing, the average time of 100 times is 16.321 milliseconds (as shown in Fig 8). Although this result is very close to the rendering time of the original model, it is relatively low compared to the original model, which is what we expected.

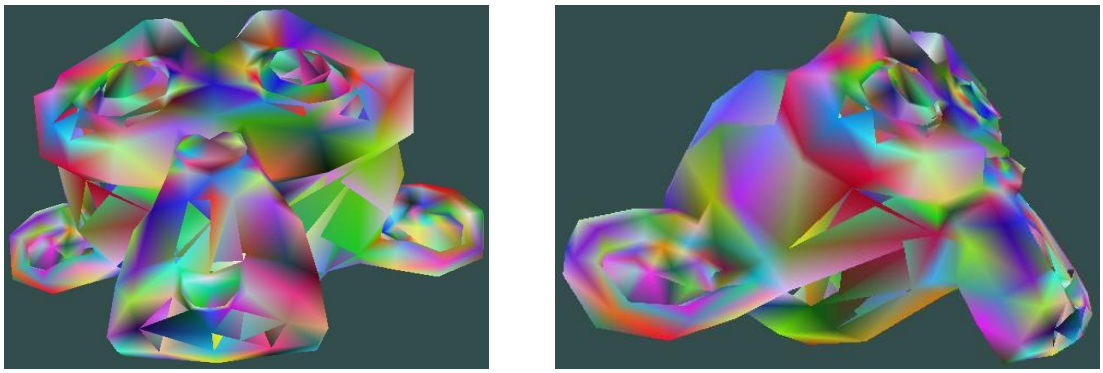


Fig 8. Positive-side rendering effect of treated with 150° model

Table 1. Test Results

	original	**30°**	**60°**	**120°**	**150°**
Number of vertices	1521	1329	1373	1430	1486
Number of triangle	2904	2562	2574	2691	2832
Average time of rendering	16.335	15.894	15.945	16.139	16.321

(Unit: ms(Average time of rendering))

V. CONCLUSION

In this paper, we have described a simple algorithm for a mesh model, and by combining the simplified test of the model at different angles, combined with the number of triangular meshes and the number of vertices before and after the simplification, at the same time, with reference to the calculation of the time-consuming rendering, at each different angle, the measured rendering time is in line with our original expected results. Our algorithm realizes that, under the premise of keeping the original geometry of the model basically unchanged, the grid of the model is effectively simplified, and the processing, transmission and drawing speed of the computer are effectively improved, although the simplified model chosen in this paper is a relatively simple monkey head model with only 2904 triangular meshes.

Compared with other papers referenced in the research, our algorithm uses the simple method of face angle calculation and face normal calculation, and uses a simple and easy to understand hole repair algorithm, the complexity of the algorithm is reduced from the initial design level. Meanwhile, the model is simplified rapidly and the number of triangular meshes is reduced to the greatest extent under the premise of keeping the geometric shape basically unchanged, which achieves our goal. So we think our algorithm is satisfactory. In subsequent studies, we will gradually apply this algorithm to the study of gaze-point rendering techniques on VR devices.

Acknowledgements

This research was supported by NRF in Korea (NRF-2017R1A1A1A05069806). SeongKi Kim is the corresponding author. The streamlined version of this paper has been briefly published on “15th International Conference on Multimedia Information Technology and Applications(MITA2019)” jointly organized by Korea Multimedia Society and Vietnam National University, and this paper gives a more detailed introduction to the algorithm we designed.

REFERENCES

[1] Michael Garland, Paul S. Heckbert, Surface Simplification Using Quadric Error Metrics, SIGGRAPH-97, 1997.

[2] Kok-Why Ng, Zhi-Wen Low, Simplification of 3D Triangular Mesh for Level of Detail Computation, CGIV-2014, 2014.

[3] He huiguang, Tian jie, Zhang xiaopeng, Zhao changming, Li guangming, 网格模型化简综述, Journal of Software, 2002.

[4] P Cignoni, C Montani, R Scopigno, A comparison of mesh simplification algorithms, Computers & Graphics, 1998, 37-54.

[5] J Rossignac, P. Borrel. Heckbert, Multi-resolution 3D approximations for rendering complex scenes, Modeling in Computer Graphics, 1993, 455-465.

[6] William J. Schroeder, Jonathan A. Zarge, William E. Lorensen, Decimation of triangle meshes, ACM SIGGRAPH Computer Graphics, 1992, 65-70.

[7] Li Shijun, Jiang Xiaotong, Tang Hui, High-quality simplified algorithm of texture model for detailed features preserving, Application Research of Computers, 2018.

[8] A. Ciampalini, P. Cignoni, C. Montani, R. Scopigno, Multiresolution decimation based on global error, The Visual Computer, 1997, 228-246.