

24 APR 1997

**MULTI-RESOLUTION GRAPHIC REPRESENTATION GENERATED BY
WEIGHT-CONTROLLED VERTEX CLUSTERING FOR INTERACTIVE
VISUALIZATION APPLICATIONS**

This Sheet Has Been Replaced



FIELD OF THE INVENTION

The present invention relates generally to computer graphics and solid modeling, and more particularly, to methods and apparatus for representing and displaying an object at multiple levels of resolution.

BACKGROUND OF THE INVENTION

REPLACED SHEETS	SG
☒ Description : paged to paged ²⁷	
☐ Claim(s) : paged to paged	
☒ Drawing(s) : paged ^{1/10, 1/11} to paged ^{1/12}	
☐ Abstract : paged to paged	
DATE OF RECEIPT	

In traditional modeling systems, detailed models are created by applying versatile modeling operations (such as extrusion, constructive solid geometry, and freeform deformations) to a vast array of geometric primitives. For display, these models must usually be tessellated into polygonal approximations. Each of these polygonal approximations typically contain millions of polygons. Enormous computing power is required to render such complex polygonal surface models, in real-time and at interactive frame rates.

In interactive visualization applications, real-time interactive manipulation of viewing conditions is vital to the understanding of an object's geometry and its relative position in the virtual scene. The delays in real-time graphic feedback and the jerky object motion will lead to overshooting, reduced feeling of control, and thus make the user less productive.

In order to accommodate complex polygonal models while still maintaining real-time interactive frame rates, methods for approximating the polygonal models and using multi-resolution models have been proposed. Model simplification algorithms can be used to generate multiple polygonal models at varying levels of

THIS SHEET HAS BEEN
REPLACED



detail, and techniques are employed by the display system to select the appropriate level of detail and render it appropriately. These approximations must resemble the original models from all directions, and the transitions between one approximation and the next must be barely noticeable in order to effectively reveal details as objects approach the viewpoint.

One of these model simplification methods is disclosed in U.S. Pat. No. 5,448,686, "Multi-Resolution Graphic Representation Employing At Least One Simplified Model For Interactive Visualization Applications" issued to Borrel and Rossignac. Their method is also described in their paper titled "Multi-Resolution 3D Approximations for Rendering Complex Scenes", Modeling in Computer Graphics, B. Falcidiena and T. L. Kunii, Eds. Springer-Verlag, pp. 455-465, 1993. This method is very fast in generating simplified models compared to other model simplification methods, and it is effective in achieving high data reduction rates. As such, it is very suitable for use in large Computer Aided Design (CAD) applications where designs are frequently edited and assemblies usually comprise hundreds of thousands of parts. However, the models it produces may be visually very different from the original models, and, moreover, the transitions between different levels of simplification during interactive viewing may not be satisfactory.

It is thus one object of this invention to provide a method for simplifying a polygonal surface model of an object such that (a) the simplified models have image fidelity close to the original model when displayed on the display screen, (b) the transitions between one level of simplification to the next are barely noticeable when the object is displayed using the simplified models, (c) the simplification process requires low computational cost, and (d) the method is effective in achieving high data reduction rates.

It is another object of this invention to provide methods and apparatus that employ one or more simplified object models for display to a viewer, the simplified models being stored and accessed as required. The aim is to provide real-time interactive frame rates during interactive visualization of the objects in the scene.



SUMMARY OF THE INVENTION

The invention described herein satisfies the above-identified needs and overcomes the limitations of the prior invention by Rossignac and Borrel. The present invention provides a method to construct a sequence of graphic models for an object, one for each desired level of simplification. Then, as the object moves in the object space during interactive viewing, the method successively switches to graphic models of the object having the most appropriate simplification resolutions. This results in a suppression of image detail, consequently with an improvement in real-time performance. The inventors have determined that many details in a displayed object can be eliminated, without a loss of visual information.

The method applies a basic principle both in the step of model simplification and the step of selecting the most appropriate model (either the original model or one of the simplified models) for displaying the object during interactive visualization.

The basic principle employed by the method is based on the following argument. In a synthetic scene, when an object is far away from the viewpoint, its image size is small. Due to the discreteness of the image space, many points on the object are mapped onto the same pixel, and this happens often when the object's model is complex and the image size is relatively small. For points mapped to the same pixel, only one point appears on the image at the pixel, and the others are eliminated by hidden-surface removal. This is a wastage in processing as many of such points are processed but never make their way to the final image. A potential solution to cutting down this wasteful processing is to identify those points that are going to fall onto the same pixel and use a new point to represent them. Only this new point is sent for rendering. Points that are most likely to be mapped to the same pixel, regardless of the direction of the viewpoint, are those points which are geometrically very close to one another in the object space.



The model simplification step of the method applies the above principle by using three-dimensional cells to identify vertices which are geometrically close to one another. These cells are placed in the object space to group vertices into clusters. Vertices in the same cluster are considered close to one another, and a new representative vertex is then created to replace these vertices in the cluster. Indirectly, determining the closeness of the vertices also helps to determine the closeness of the polygons. For example, two rectangles are close together if their corresponding vertices are close to each other. By using different clustering cell sizes, the definition of "closeness" will be different, and this allows the original model to be simplified into models of different levels of detail.

During display of the object, the size of the object space covered by a pixel on the display screen is determined, and the model selection step of the method selects the most appropriate model of the object for display. The model selection criterion is based on the determined size of the object space covered by a pixel, and the size of the clustering cells used in generating each of the simplified models.

More specifically, there is disclosed a method for execution by a data processor that prepares an object for display during interactive visualization. The method includes a step of preprocessing an original model, which is a polygonal boundary representation of the object, so as to produce a plurality of simplified models with each simplified model having fewer faces and vertices than the original model. The step of preprocessing includes the following steps:

- (1) Grading—a weight is computed for each vertex in the original model to reflect the vertex's visual importance. The value of the vertex's weight is a function of the angles between all pairs of edges incident to the vertex, and the lengths of these incident edges.
- (2) Sorting vertices—the vertices in the original model need to be sorted in non-increasing order of their weights.



(3) Triangulation—polygons in the original model are decomposed into non-overlapping triangles without introducing new vertices into the original model. The graded and triangulated original model needs to be stored for subsequent generation of other simplified models.

(4) Clustering—vertices are grouped into clusters based on geometric proximity. The positions of the clustering cells are based on the positions of the original vertices and the weights of the vertices.

(5) Thick-edge computation—for each group of triangles which are going to degenerate into the same edge (due to Step (6) — synthesis), compute an estimate for the thickness of the group of triangles.

(6) Synthesis—a representative vertex is computed to replace the vertices in each cluster and thus simplify some triangles into edges and points.

(7) Reconstruction of normals—the vertex-normals of each triangle are reconstructed.

(8) Elimination—duplicated triangles, edges and points in the simplified model are removed.

(9) Storing the simplified model for subsequent use.

Subsequent generation of other simplified models of the object requires execution of only Step (4) to Step (9) for each simplified model with a new clustering cell size for Step (4). A larger clustering cell size will generate a more simplified model (less detailed) while a smaller clustering cell size will generate a less simplified model (more detailed).

During display of the object, the method further includes a step of selecting either the original model or one of the simplified models for display. The step of selecting



a model includes a step of determining the size of the object space covered by a pixel on the display screen, and the step of selecting a model selects either the original model or one of the simplified models as a function of at least the determined size of the object space covered by a pixel, and the size of the clustering cells used in generating each of the simplified models.

The method achieves the objects of the present invention by using:

(a) weight-controlled vertex clustering in which the positions of the clustering cells are based on the positions of the original vertices and the weights of the vertices. This helps to preserve important features of the object, improves the approximation quality, and thus leads to simplified models with higher image fidelity to the original model. Using weight-controlled vertex clustering also results in more consistent placements of important features from one simplified model to the next. The more consistent placements of important features lead to smoother transitions from one level of simplification to the next when object is displayed using the simplified models;

(b) thick-edges to further improve the image quality of the displayed simplified model. Each thick-edge is rendered as a multi-pixel-wide thick-line graphics primitive on the display screen, and a dynamic normal is computed for the thick-line to improve its shading and intensity;

(c) an appropriate clustering cell size to achieve the desired level of simplification and data reduction rate;

(d) a model selection step to select either the original model or one of the simplified models, for display of the object during interactive visualization, as a function of at least the determined size of the object space covered by a pixel, and the size of the clustering cells used in generating each of the simplified models.



BRIEF DESCRIPTION OF THE DRAWING

- FIG. 1** is a block diagram of an exemplary raster graphics systems;
- FIG. 2** is a block diagram of the raster graphics system of **FIG. 1** that shows a memory for storing object models and simplified versions thereof;
- FIG. 3** is a flowchart that depicts the steps of the preprocessing step of the method of the invention;
- FIG. 4** is a flowchart that depicts the operation of the method of the invention;
- FIG. 5a** shows an exemplary object model;
- FIG. 5b** illustrates the data structure used to store the model in **FIG. 5a**;
- FIG. 6** illustrates part of the computation used to derive the weight of a vertex;
- FIG. 7a** shows an exemplary polygon which is to be triangulated;
- FIG. 7b** shows a triangulation of the polygon in **FIG. 7a**;
- FIG. 8a** shows an exemplary two-dimensional object model whose vertices are being clustered by cells;
- FIG. 8b** illustrates the resulting model after synthesizing the representative vertices for the vertex clusters shown in **FIG. 8a**;
- FIG. 8c** shows an exemplary two-dimensional object model identical to that in **FIG. 8a**, except that its vertices are being clustered by larger cells;
- FIG. 8d** illustrates the resulting model after synthesizing the representative vertices for the vertex clusters shown in **FIG. 8c**;
- FIG. 9** is a dataflow diagram that illustrates the operation of the preprocessing step of the method of the invention.

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 illustrates an exemplary raster graphics system which includes a main (Host) processor and a graphics subsystem. The Host processor executes an application program and dispatches graphics tasks to the graphics subsystem.

The graphics subsystem includes a pipeline of several components that perform



operations necessary to prepare geometric entities for display on a raster display device. For the purposes of describing the invention, a model of the graphics subsystem is employed that contains the following functional units. It should be realized that this particular model is not to be construed in a limiting sense upon the practice of the invention.

A Geometric Processor unit performs geometric and perspective transformations, exact clipping on primitives against screen (window) boundaries, as well as lighting computations. The resulting graphics primitives, e.g. points, lines, triangles, etc., are described in screen space (integral) coordinates.

A Scan Conversion (Rasterization) unit breaks down the graphics primitives into raster information, i.e. a description of display screen pixels that are covered by the graphics primitives.

A Graphics Buffer unit receives, stores, and processes the pixels from the Scan Conversion unit in accordance with the method of the invention. The most common buffers associated to the Graphics Buffer unit are the image buffer and the z-buffer.

A Display unit receives pixels from the Graphics Buffer unit and transforms these pixels into information displayed on the output device, typically a raster screen.

Having described an exemplary graphics processing system that is suitable for use in practicing the invention, a description is now provided of a presently preferred method of model simplification.

For the purposes of describing the invention, it is assumed that objects are described in polygonal boundary representations. Models with curved surfaces are in general tessellated, i.e. approximated with planar polygons, for graphic purposes and thus also benefit from the simplification technique of the invention.





Preprocessing Step

The preprocessing step is the model simplification step. The preprocessing step constructs one or several simplified models (with decreasing level of detail) for each object in a scene to be visualized. The step of preprocessing includes the following steps: (1) grading, (2) sorting vertices, (3) triangulation, (4) clustering, (5) thick-edge computation, (6) synthesis, (7) reconstruction of normals, (8) elimination, and (9) storing the simplified model. Step (1) to Step (3) need only be executed once for an object regardless of the number of simplified models to be generated. **FIG. 3** is a flowchart that depicts the flow of these steps of the preprocessing step.

For the purposes of describing the preprocessing step, it is assumed that each original model is described by a vertex table VT, a vertex-normal table VNT, and a polygon table PT. The vertex table VT contains the supporting vertices used in the model. Each entry of the vertex-normal table VNT contains a vector which will be used by some polygon vertex as the normal at the vertex (vertex-normal). The vertex-normal at each vertex of a polygon is used for lighting calculations at the vertex during rendering of the polygon. The polygon table PT contains the polygons used in the model with each polygon represented as an ordered list of references (indices) to the vertex table VT (this list will be called the polygon's vertex list). Each polygon in PT also has another ordered list of references (indices) to the vertex-normal table VNT, with each vertex-normal index for each polygon's vertex (this list will be called the polygon's vertex-normal list).

FIG. 5a and **FIG. 5b** gives an example of a polygonal model and its vertex table VT, vertex-normal table VNT and polygon table PT. **FIG. 5a** is a model made of three polygons P_1 , P_2 and P_3 . The vertices in the model are V_1 , V_2 , ..., V_6 . The small arrows at the polygon vertices indicate the vertex-normals at the polygon vertices. The normals used in the model are N_1 , N_2 , ..., N_5 . **FIG. 5b** depicts the vertex table VT, vertex-normal table VNT and the polygon table PT which,



together, describe the model in **FIG. 5a**.

(1) Grading

In this step, a weight is computed for each vertex in the original model, and it is stored in a weight table WT for use in subsequent processing steps. The weight of a vertex reflects the visual importance of the vertex. Two factors may be used to determine the visual importance of a vertex. A vertex V is visually important if (1) it has high probability of lying on the object's silhouettes from an arbitrary viewing direction, and (2) it bounds large faces that should not be affected by the removal of small details. The first factor (Factor 1) may be estimated using $\cos(\theta/2)$ where θ is the maximum angle between all pairs of edges incident to V . The second factor (Factor 2) may be estimated using the length of the longest among all of the edges incident to V . Finally, the weight of V is the product of the two estimated values.

The following is the reason why $\cos(\theta/2)$ is used as an estimation for Factor 1. With reference to **FIG. 6**, let V be the vertex to be assigned weight. Suppose θ is the maximum angle between all pairs of edges incident to V . Now, a sphere of radius r is centered at V , and also let V be the apex of a cone with tip angle θ and the base of the cone touches the sphere. A point on the sphere to V will represent a viewing direction from the point to V .

The apex of the cone represents a corner of the object, and the inside of the lower cone is the interior of the object. The only directions in which V cannot be seen are directions from the cap below the base of the cone. So, from any arbitrary viewing direction, the probability of seeing V on the silhouettes is



$$\frac{\text{area of sphere} - \text{area of bottom cap} - \text{area of top cap}}{\text{area of sphere}}$$

$$= \frac{4\pi r^2 - 2\pi r^2 (1 - \cos \frac{\theta}{2}) - 2\pi r^2 (1 - \cos \frac{\theta}{2})}{4\pi r^2}$$

$$= \cos \frac{\theta}{2}$$

In order to compute a weight for each vertex, it is necessary to construct (when not given as part of the input) the vertex-edge adjacency information from the vertex table VT and the polygon table PT of the original model. The following describes one possible approach. First, a vertex-edge adjacency list is associated to each vertex. The list associated to V is supposed to contain the indices of those vertices adjacent to V , where each adjacent vertex represents an edge incident to V . A single pass through the polygon table PT is sufficient to complete the task of building all the lists. For each vertex V , the computation of the estimate for Factor 1 involves traversing its vertex-edge adjacency list to identify the maximum angle θ between all pairs of edges incident to V . As for the estimate for Factor 2, just one pass through the list of adjacent vertices is enough to find the longest length L among all the edges incident to V . The weight W of V is the product of L and $\cos(\theta/2)$. W is then stored in the weight table WT for subsequent use. The i -th entry of the weight table WT contains the weight of the vertex stored in the i -th entry of the vertex table VT.

The vertex-edge adjacency lists constructed above may need to be stored for use in subsequent step in the preprocessing. For this, a vertex-edge adjacency table VEAT is created such that its i -th entry contains the vertex-edge adjacency list of the i -th vertex in the vertex table VT.

(2) Sorting vertices

To prepare the model for Step (4) (clustering step), the vertices in the vertex table VT need to be sorted in a non-increasing order on their weights in WT. An efficient way to accomplish this is to use a temporary storage to store the required ordering of the vertices without changing both VT and WT, and thus avoid changing the



polygon table PT. The temporary storage is a table which will be called the vertex-order table VOT. The number of entries in VOT is the same as that in VT and WT. The i -th entry of VOT is supposed to contain the index of the vertex (in VT) which has the i -th largest weight. Initially, VOT is initialized such that the i -th entry of VOT contains the index of the i -th vertex in VT. An efficient in-place sorting algorithm can be used to sort VOT. During sorting, to decide whether two VOT elements need to be exchanged, the sorting comparison is done on the weights of the vertices indexed by these two VOT elements.

(3) Triangulation

The triangulation step decomposes each polygon into one or more non-overlapping triangles. Triangulation can be accomplished using various known algorithms and need not introduce any new vertices into the original model. For lighting calculations, it is often assumed that a polygon's vertices are stored in an order that corresponds to a counter-clockwise rotation around a normal to the polygon's plane. Triangulation does not affect the normal, and does not alter the geometric object represented by the original model. After triangulation, the resulting triangles are stored in a triangle table TT. Each triangle in the triangle table TT is represented as an ordered list of three references (indices) to the vertex table VT, and an ordered list of three references (indices) to the vertex-normal table VNT.

FIG. 7a and **FIG. 7b** shows an example of triangulating a polygon. **FIG. 7b** illustrates a triangulation of the polygon in **FIG. 7a**. Each resulting triangle has three vertices associated therewith.

(4) Clustering

The clustering step uses one or more clustering cells to group vertices in VT into one or more clusters. In this invention, the clustering is done as follows:

- (i) Let V be the most highly weighted vertex in VT which has not been assigned to any cell,



- (ii) if V does not lie in any existing cell, then create a new cell centered at V and assign V to this new cell,
- (iii) else if V lies in exactly one existing cell, then assign V to the cell,
- (iv) else if V lies in the intersection of two or more existing cells, assign V to one of these cells according to one of the following cell selection criteria:
 - (a) assign V to the cell which has the most highly weighted center vertex,
 - (b) assign V to the cell whose center is nearest to V ,
 - (c) assign V to the cell C whose average length of the edges incident to V to the vertices already assigned to C is the shortest, and
 - (d) assign V to the cell which has the largest number of vertices adjacent to V (a vertex is adjacent to V if and only if there is an edge joining them),
- (v) go to step (i) if there is unassigned vertex in VT .

All the vertices assigned to a cell form a cluster. The result of the clustering step is a cluster table CT and a cluster-index table CIT . Each entry in CT corresponds to a cluster and it stores the center position of the clustering cell, and a list of indices of the vertices (in VT) which belongs to the cluster. As for the cluster-index table, its i -th entry contains the index of the cluster (in CT) which the i -th vertex in VT belongs to. Therefore, the number of entries in CIT is the same as that in VT .

The clustering cell used can be some simple three-dimensional shape, such as cube and sphere.

For each simplified model, the clustering step can be done in time linear to the number of vertices in the vertex table VT . To achieve such efficiency, the object space is first uniformly subdivided into cubical grids, each with width equal to the longest side of the smallest three-dimensional minimax box bounding the clustering cell used. For example, if the clustering cell used is an axis-aligned cube, then the width of each cubical grid is just the width of the clustering cube, whereas, if the clustering cell used is a sphere, then the width of each cubical grid is the sphere's diameter. For the purpose of describing this step of the method, it is assumed that axis-aligned cube is used as the clustering cell.



The main idea here is, whenever a new clustering cell is created, it is associated to the grid which contains the cell's center vertex. Though clustering cells can intersect one another, a cell's center vertex can never be inside another cell. This leads to a maximum of eight cells (or other upper bounds for other clustering shapes) that can be associated to each grid.

From the vertex-order table VOT, it is possible to visit each of the vertices in the vertex table VT exactly once in an ordered manner from the vertex with the highest weight to the vertex with the lowest weight. For each vertex V being visited, the grid it lies in can be known immediately. To find out whether V lies in any existing cells, it is necessary only to check the cells associated to the grid which contains V , and also to the other seven adjacent grids which are nearest to V . Only the cells associated to these eight grids can contain V .

If V does not lie in any of these cells, a new cell is created with its center at V , and associated to the grid which contains V . On the other hand, if V lies in exactly one of the cells, then assign V to the cell. Otherwise, if V lies in more than one cell, then assign V to one of these cells according to one of these cell selection criteria: (a) assign V to the cell which has the most highly weighted center vertex, (b) assign V to the cell whose center is nearest to V , (c) assign V to the cell C whose average length of the edges incident to V to the vertices already assigned to C is the shortest, and (d) assign V to the cell which has the largest number of vertices adjacent to V .

For selecting a cell using cell selection criterion (c) or (d), the vertex-edge adjacency information can be obtained from the vertex-edge adjacency table VEAT created in the grading step.

Besides using a three-dimensional array to represent the array of grids, it is practical to use a hash table to store only those non-empty grids. The number of entries in the hash table is proportional to the number of vertices in VT. The 3D coordinates of a fixed corner of each grid is used as the key for hashing.



In another embodiment of the method to find the cells that a vertex V lies in, instead of using a cubical grid the same size as the clustering cube, a cubical grid one eighth of the volume is used. For the smaller grid, at most one cell can be associated to it. To find out all the existing cells a vertex V lies in, it is only necessary to check the cell associated to the grid which V lies in, and the cells associated to the 26 adjacent grids.

In this clustering step of the invention, a weight-controlled vertex clustering approach is used to position the clustering cells based on the positions of the original vertices and the weights of the vertices. This helps to preserve important features of the object, improves the approximation quality, and thus leads to simplified models with higher image fidelity to the original model. Using weight-controlled vertex clustering also results in more consistent placements of important features from one simplified model to the next. The more consistent placements of important features lead to smoother transitions from one level of simplification to the next when object is displayed using the simplified models.

The clustering step, in another embodiment, includes a clustering approach which is capable of greatly reducing the cases of thin triangles in the original model becoming much wider triangles in the simplified model. Such effect is very noticeable during display of the object and can degrade the approximation quality of the simplified model significantly. The clustering approach is described as follows:

- (i) Let V be the most highly weighted vertex in VT which has not been visited,
- (ii) if V does not lie in any existing cell, then create a new cell centered at V and assign V to this new cell,
- (iii) go to step (i) if there is unvisited vertex in VT ,
- (iv) for each vertex V in VT which has not been assigned to any cell, assign it to the cell whose center vertex is nearest to V .



Similarly, the cubical grids used above can be employed in this clustering method too.

(5) Thick-edge computation

When elongated parts of the object are reduced to just edges (done in Step (6) – synthesis), they may become less visible on the display screen when rendered as one-pixel-wide lines. If the actual thickness of an elongated part in the object space is known, its pixel-width on the screen can be deduced. Then, by rendering the edge using thick-line primitive with varying pixel-width, the actual visual effects of the elongated part can be approximately reproduced.

Thickness of an edge can be estimated as follows. Let S be a set of triangles where each triangle has its vertices falling in clusters A and B only. These triangles will degenerate into edges in the synthesis step. Let V_A be the set of vertices that belong to cluster A and each is used by at least a triangle in S . Let P_A be the average position of all the vertices in V_A . And, V_B and P_B are similarly defined. Let L be the straight line passing through P_A and P_B . Let D_A be the average perpendicular distance of all points in V_A to the line L , and likewise for D_B . Then, the thickness of the edge, formed by collapsing triangles in S , is estimated by $(2D_A + 2D_B) / 2 = D_A + D_B$.

To begin the computation of thickness for all thick-edges, it is necessary to find all the cluster-pairs which have triangles, which are going to degenerate into edges, using the vertices in them. For this purpose, one efficient way is to use a hash table whose number of entries is proportional to the number of triangles in the triangle table TT . For each triangle T in the triangle table TT , if the three vertices of T belong to exactly two distinct clusters, then T will degenerate into an edge after Step(6) (synthesis step). The information about which cluster a vertex belongs to can be obtained from the cluster-index table CIT . Let C_1 and C_2 be the indices of the two clusters, where $C_1 < C_2$. Then the pair (C_1, C_2) is used as the key for hashing. The hash table element which (C_1, C_2) is being hashed to, contains C_1 , C_2 and a list of indices of triangles where each of them, like T , has



vertices belonging to only the two clusters indexed by C_1 and C_2 . The index of T is then added to the list.

After all the triangles in TT have been visited, a pass through the hash table is needed to visit the triangle-index list associated to each non-empty hash table element. For each triangle-index list, an estimate of the thickness of all the triangles indexed in the list is computed as described above. This thickness is then associated to each triangle indexed in the list. A temporary storage, called the edge thickness table ETT, is used to store these thickness. The i -th entry of ETT contains the thickness associated to the i -th triangle in the triangle table TT.

(6) Synthesis

In this step, a representative vertex is computed to replace the vertices in each cluster which has more than one vertex. A cluster which has only one vertex will have that vertex as the representative vertex of the cluster. For each cluster C which has more than one vertex, the vertices which belong to it can be readily found out from the cluster table CT. Then the representative vertex of the cluster C can be computed using one of these vertex synthesis criteria: (a) select the center of the clustering cell as the position of the representative vertex, (b) compute a weighted-average position of all the vertices in the cluster using their weights stored in the weight table WT, so that the representative vertex is located at the weighted-average position, and (c) select the position of the vertex which is nearest to the weighted-average position of all the vertices in the cluster.

The weighted-average position of vertices $V_1, V_2, \dots, V_n$ with positions $P_1, P_2, \dots, P_n$ and weights $W_1, W_2, \dots, W_n$ is computed as follows

$$(W_1 \cdot P_1 + W_2 \cdot P_2 + \dots + W_n \cdot P_n) / (W_1 + W_2 + \dots + W_n)$$

The new representative vertices are stored in a new vertex table SVT, where the i -th entry of SVT contains the representative vertex of the i -th cluster in the cluster table CT. The triangle table TT is copied to a new triangle table NTT, and the



vertex indices of each triangle in NTT is updated using the cluster index table CIT to make references to the new vertex table SVT.

When preparing many objects for display, one or more of the foregoing vertex synthesis criteria for computing a representative vertex for a cluster may be employed, depending on the geometry of the particular object being prepared. For each object, it is more preferable to use the same vertex synthesis criteria for generating all its simplified models. This will result in more consistent positions of the representative vertices across different levels of simplified models of the object, and this helps in improving the smoothness in transitions between different levels of simplified models when the object is displayed.

FIG. 8a, FIG. 8b, FIG. 8c and FIG. 8d illustrate two examples of clustering a two-dimensional object and synthesizing the representative vertices. **FIG. 8a** and **FIG. 8c** show the original model, which has already been graded and triangulated, but, for the sake of a simple illustration, the weights of the vertices and the triangles are not shown in the figures. The vertices in the model are intentionally labeled in such a way that vertex V_i has higher weight than V_{i+1} .

Referring to **FIG. 8a**, which shows clustering cells centered at some of the model's vertices. These cells are created in an order according to the order of the weights of their center vertices. In this example, when a vertex V lies in more than one cell, the cell selection criterion used during clustering is to assign V to the cell with the most highly weighted center vertex. As for the vertex synthesis criterion used during synthesis, the center of the clustering cell is selected as the position of the representative vertex. When vertices in the cells are replaced by their representative vertices, the result is the model in **FIG. 8b**.

The example illustrated in **FIG. 8c** and **FIG. 8d** is the same as that in **FIG. 8a** and **FIG. 8b**, except, now, the clustering cells used are larger. The model in **FIG. 8d** is the result of applying the clustering step and the synthesis step to the model in **FIG. 8c**.



(7) Reconstruction of normals

In this step, the vertex-normals of each new triangle have to be reconstructed in an attempt to preserve the smooth shadings on smooth faces and the discontinuities in shading at sharp edges and corners of the original model. The reconstruction requires, first, the construction of the vertex-normal information of the triangulated original model. The information from the clustering step is then used to update the vertex-normal information which is in turn used to compute new vertex-normals for each new triangle.

To construct the vertex-normal information, a vertex-normal set is initially associated to each vertex V in the vertex table VT of the original model. Each member of a vertex-normal set is a set of indices of the triangles in the triangle table TT. This set of indices of triangles will be called a triangle-index set. In each triangle-index set of V , the indexed triangles use the same vertex-normal (in the vertex-normal table VNT) at the triangle vertices that use V . So, intuitively, each triangle-index set of V represents a vertex-normal that is shared by the indexed triangles at V , and the number of triangle-index sets in the vertex-normal set associated to V corresponds to the number of different vertex-normals at V .

The vertex-normal information constructed is then updated using the information (in the cluster table CT and the cluster-index table CIT) from the clustering step. Assuming that vertices $V_1, V_2, \dots, V_k$ in the vertex table VT are in the same cluster, and their respective vertex-normal sets are $S_1, S_2, \dots, S_k$, the vertex-normal information can be updated as follows:

- (i) let $S \leftarrow S_1 \cup S_2 \cup \dots \cup S_k$,
- (ii) if $\exists X, Y \in S$ such that $X \neq Y$ and $X \cap Y \neq \emptyset$ then

$$S \leftarrow (S - \{X\} - \{Y\}) \cup \{X \cup Y\},$$
- (iii) go to step (ii) if S is changed.

The set S is the new vertex-normal set which must be associated to the representative vertex, computed in the synthesis step, of the cluster. The above



vertex-normal information updating step has to be done once for each and every cluster, and each new vertex-normal set is associated to its respective representative vertex of the cluster.

For those new triangles (in triangle table NTT) which degenerate into edges and points, as a result of the synthesis step, their indices must be removed from all triangle-index sets in the new vertex-normal sets. If any triangle-index set becomes empty, the empty set is removed from the corresponding new vertex-normal set.

Now, for each vertex V in the vertex table SVT created in the synthesis step, there is a new vertex-normal set associated to it. The next step is to compute the vertex-normals for each new triangle from the new vertex-normal sets. For each triangle-index set in the new vertex-normal set of V (in SVT), an average vector is computed using the unit vectors normal to the indexed triangles (in the triangle table NTT). This average vector is added to the new vertex-normal table SVNT, and the appropriate vertex-normal index in each indexed triangle's vertex-normal list is updated to reference this new average vector. The updated triangle table NTT is called NTT' in FIG. 9.

In practice, a vertex-normal set may be represented by a list (will be called a vertex-normal set list), in which each element is again a list (will be called a triangle-index set list) of indices of triangles in the triangle table TT. Also, three pairs of pointers (or three pointer-pairs) are created for each triangle in TT, such that in each pair, the first pointer points to a triangle-index set list which has an element that contains the index of the triangle, and the second pointer points to this element in this triangle-index set list.

The updating of the vertex-normal set lists for a cluster of vertices $V_1, V_2, \dots, V_k$ is done as follows. Let Q be the new vertex-normal set list associated to the representative vertex U of this cluster, and it is initially empty. For each triangle T which is any triangle indexed in the triangle-index set lists associated to $V_1, V_2, \dots, V_k$, the triangle-index set lists pointed to by the first pointers in the three pointer-



pairs of T are determined. If all three triangle-index set lists are associated to some of $U, V_1, V_2, \dots, V_k$, then delete the indices of T (with the help of the second pointers in the three pointer-pairs of T) from these triangle-index set lists, move all these lists out from their corresponding vertex-normal set lists, append them as a new triangle-index set list, and add the new list to Q . On the other hand, if exactly two of the three triangle-index set lists are associated to some of $U, V_1, V_2, \dots, V_k$, then delete the indices of T from the three triangle-index set lists, but only move the two lists out from their corresponding vertex-normal set lists, append them as a new triangle-index set list, and add the new list to Q . Otherwise, if exactly one of the three triangle-index set lists is associated to some of $U, V_1, V_2, \dots, V_k$, then move the list out from its vertex-normal set list, and add it to Q . Note that one of the three pointer-pairs of each triangle indexed in the list must be updated after a triangle index set list is added to Q . All these, however, can be done efficiently in a depth-first manner in exploring triangles in the same cluster.

The above operations are repeated for the cluster until every vertex-normal set list associated to $V_1, V_2, \dots, V_k$ is empty.

When the above step is done for every cluster, the new vertex-normal set lists are used to compute the new vertex-normals (in SVNT) for each new triangles, as described in a previous paragraph.

(8) Elimination

As a result of merging of vertices in a cluster into one vertex, some triangles degenerate into edges and points. A triangle with three identical vertices will degenerate into a point, while a triangle with exactly two identical vertices will degenerate into an edge. These degenerated triangles are identified from NTT' and for each such triangle, its duplicated vertex indices and the corresponding vertex-normal indices are removed.

The next thing to do is to remove those duplicated triangles, edges and points in the updated NTT'. More specifically, these are removed: (a) duplicated triangles,



(b) duplicated edges, (c) duplicated points, (d) edges that are the same as some triangles' edges, and (e) points that fall on the end-points of some edges or on the vertices of some triangles.

A hash table may be used to efficiently identify duplications of primitives (triangles, edges and points). The number of entries in the hash table is proportional to the number of primitives stored in NTT' . For each primitive P in NTT' , if P is a triangle, then its three vertex indices V_1 , V_2 and V_3 , where $V_1 < V_2 < V_3$, are used as a key for hashing. If the triangle (V_1, V_2, V_3) is found in the hash table, then P is deleted from NTT' . On the other hand, if the triangle (V_1, V_2, V_3) is not found in the hash table, then (V_1, V_2, V_3) is added to the hash table and P is retained, and each of (V_1, V_2) , (V_2, V_3) , (V_1, V_3) , (V_1) , (V_2) and (V_3) is also hashed and added to the hash table if it is not already inside. If P is an edge, then its two vertex indices V_1 and V_2 , where $V_1 < V_2$, are used as a key for hashing. If the edge (V_1, V_2) is found in the hash table, then P is deleted from NTT' . On the other hand, if the edge (V_1, V_2) is not found in the hash table, then (V_1, V_2) is added to the hash table and P is retained, and each of (V_1) and (V_2) is also hashed and added to the hash table if it is not already inside. If P is a point, then its vertex index V_1 is used as a key for hashing. If the point (V_1) is found in the hash table, then P is deleted from NTT' . On the other hand, if the point (V_1) is not found in the hash table, then (V_1) is added to the hash table and P is retained.

The use of the above algorithm requires NTT' to be pre-sorted in such a way that all the triangles come before all the edges, and all the edges come before all the points, so that, during the identification of duplicated primitives, all triangles are visited before all edges, and all edges are visited before all points.

The retained triangles in NTT' are transferred to a triangle table STT , the retained edges, together with their edge thickness from the edge thickness table ETT , are transferred to a edge table SET , and the retained points are transferred to a point table SPT . The primitives in STT , SET and SPT all make references to the vertex table SVT created in Step (6) (synthesis step).



(9) Storing simplified model

The simplified model is described by the vertex table SVT, the vertex-normal table SVNT, the triangle table STT, the edge table SET and the point table SPT. These tables, together with the size of the clustering cell used, are stored in the memory or on other storage media for subsequent use.

No reconstruction of normals

In another embodiment of the preprocessing step, a simplified model is generated without the reconstruction of vertex-normals (Step (7)). In this case, the simplified model does not have the new vertex-normal table SVNT, and for each triangle in the triangle table STT, the triangle's vertex-normal list makes references to the normals stored in the vertex-normal table VNT of the original model. The vertex-normal table VNT is thus shared between the original model and the simplified model.

Next simplified model

For the next level of simplified model of the same object, only Step (4) to Step (9) need to be executed with a new clustering cell size for Step (4). The vertex table VT, the triangle table TT, the weight table WT, the vertex-edge adjacency table VEAT, and the vertex-order table VOT can be reused.

Having generated a plurality of simplified models for the object, the object's original model and the simplified models are stored within a database of graphic models. FIG. 2 illustrates a memory that is coupled to the CPU of FIG 1. The memory stores a plurality of object models (MODEL 1 to MODEL n) and, in accordance with the invention, also stores one or more simplified models ($S_{1,1}$, $S_{1,2}$, etc.) for each of the object models. The cell sizes ($C_{1,1}$, $C_{1,2}$, etc.) used for the generation of the simplified models are also stored together with the simplified models.



Choosing clustering cell sizes

To choose the cell sizes for generating the sequence of simplified models of an object, a simple method can be adopted. A minimax box bounding the object is computed first. Then, the clustering cell widths can be some fractions of the length of one of the side of the object's bounding box. For example, if the length of a side of the bounding box is L , the cell widths can be $k_1 \cdot L$, $k_2 \cdot L$, $k_3 \cdot L$, ... and $k_m \cdot L$, where each k_i is a constant of value between 0 and 1 and $k_i < k_{i+1}$, and m is the number of simplified models to be generated. These constant fractions can be adjusted by the user to suit the complexity of the original model, the number of objects in the scene, and the rendering power of the graphics display apparatus. For instance, very often, for a very complex original model with many small features, a relatively small cell size is enough to significantly reduce its complexity. If there are many objects in the scene or the rendering power of the display apparatus is low, it is more preferable to use some cell sizes which are relatively much bigger.

FIG. 9 shows the dataflow between the steps of the preprocessing for the generation of a single level of simplified model.

Model Selection Step

The method of the invention further includes a step of selecting either the original model or one of the simplified models for displaying an object. For example, when an object is very far away from the viewpoint, it may be sufficient to choose a very simplified model to display the object. This will save some time in displaying the object compared to always displaying the original model. Whereas, if the object is near to the viewpoint, a more detailed model (or a less simplified model) may be required to reveal the details on the object.

The step of selecting a model includes a step of determining the size of the object space covered by a pixel on the display screen, and this will be called the object-space-to-pixel-space-ratio or R for short. The value of R of an object with a distance D from the viewpoint is



$$R = (2D \tan (FOV/2)) / H$$

where FOV is the vertical field of view angle of the camera, and H is the height of the viewing window in pixels. The distance D of an object from the viewpoint can be readily estimated using the shortest distance between the viewpoint and a pre-computed simple bounding volume enclosing the object. An enclosing sphere of the object is a convenient bounding volume for computing D . If the object or the viewpoint moves, such that D changes, or FOV or H is changed, then R has to be recomputed.

For displaying the object, the model with the largest clustering cell width which is less than or equal to R will be chosen. Here, it is said that the original model has clustering cell of width 0. Let M be the model chosen. There is no benefit in choosing a model more detailed than M for display, since at this distance from the viewpoint, each cell width in the object space will be mapped to approximately one or less pixel width. This model selection is done independently for each object, and each has its own value of R .

However, to maintain real-time interactive frame rates, a model less detailed than M may need to be selected for display. This model is the most detailed model of the object whose rendering time, including the calculation of perspective transformation, lighting computation, scan conversion, etc., is consistent with providing smooth motion of the object on the display.

An adaptive model selection technique can be integrated to select a model for each object in the scene for display such that the total display time of all the objects is consistent with providing a good frame rate. An example of such adaptive model selection technique is described by Funkhouser and Séquin in "Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments", Computer Graphics (SIGGRAPH '93 Proceedings), pp. 247-254, 1993.



Usually, when there are many complex objects in the scene, the rendering workload will generally be high almost all the time. To speed up the adaptive model selection step, an attempt is made to reduce the number of models of each object that need to be considered by the optimization algorithm in the model selection step. This is done by choosing a model which is not more detailed than M' where M' is the model with the largest cell width less than or equal to $R \cdot X$, and $X > 1$. X can be adjusted, automatically by the adaptive model selection algorithm or manually by the user, according to the general rendering workload. Note that the use of $X > 1$ does not stop the most detailed model of an object from being selected.

Referring again to **FIG. 2**, the system includes user input devices such as a keyboard, mouse or trackball. During use, an entire scene may be moved according to a user's command. The user inputs these commands using the input devices. In response, the CPU may access a simplified model to be used for each object within the scene. The system automatically switches to the simplified models when the user initiates the motion of the objects in the scene. When motion stops, the system returns to the full resolution mode, triggered by an explicit user action, or by itself if there is no user intervention for a predefined interval of time. The goal is to select from the memory the most detailed model for each object, while also permitting real-time interactive motion to be achieved. As the model selection step requires the clustering cell size used for generating each simplified model, the clustering cell size is kept together with the simplified model in the memory.

Rendering a model

When a model is to be displayed, the Host processor will send the necessary commands and data from the model to the graphics subsystem.

To display a triangle, its vertices and its vertex-normals are sent to the graphics subsystem. For rendering a thick-edge using a thick-line, besides the vertices at the two endpoints, the pixel-width of the thick-line primitive must also be sent to



the graphics subsystem. The pixel-width of the thick-line is computed as T / R , where T is the estimated thickness of the thick-edge, and R is the object-space-to-pixel-space-ratio of the object as described above.

The shading or intensity of a thick-line which renders a thick-edge can, in general, be improved by using a dynamic normal instead of a fixed one for the thick-line. This dynamic normal is only computed during rendering. The improvement is particularly good if that the elongated part approximated by the edge is cylinder-like. The dynamic normal is the vector perpendicular to the edge, in the plane containing the edge and the camera, and the angle it makes with the vector from any point on the edge to the camera is acute. The dynamic normal is calculated using the formula $(V_1 - C) \times (V_2 - C) \times (V_2 - V_1)$ where V_1 , V_2 are the edge's endpoints and C is the camera's position.

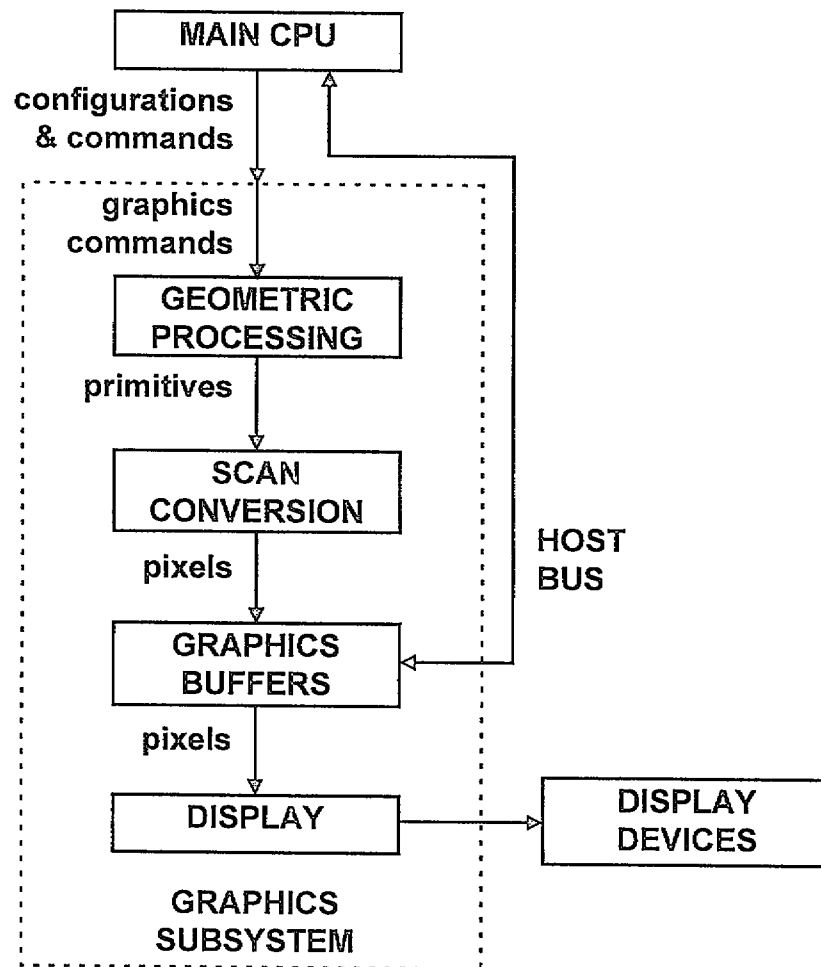
Operation Of The Method Of Invention

FIG 4 is a flowchart that depicts the operation of the method of the invention. First, the system loads or generates a database of objects appearing in a scene to be displayed. These objects may be elements of, for example, a CAD database or a virtual environment. For each of these object, one or more simplified models are generated by the preprocessing step depicted in **FIG 3**. The original models and the simplified models of all objects are stored in the memory as shown in **FIG 2**. At this point, the system is ready to begin an interactive display session with a user.

In response to a command from the user to change the position of the displayed object, the system accesses the memory to obtain a suitable model. As was previously described, it is within the scope of the invention for the system to select an appropriate simplified model for display. The selected model is passed to the graphics subsystem, resulting in the display of the selected model on the display screen.

FIG. 1

1



This Sheet Has Been Replaced



REPLACED

FIG. 2

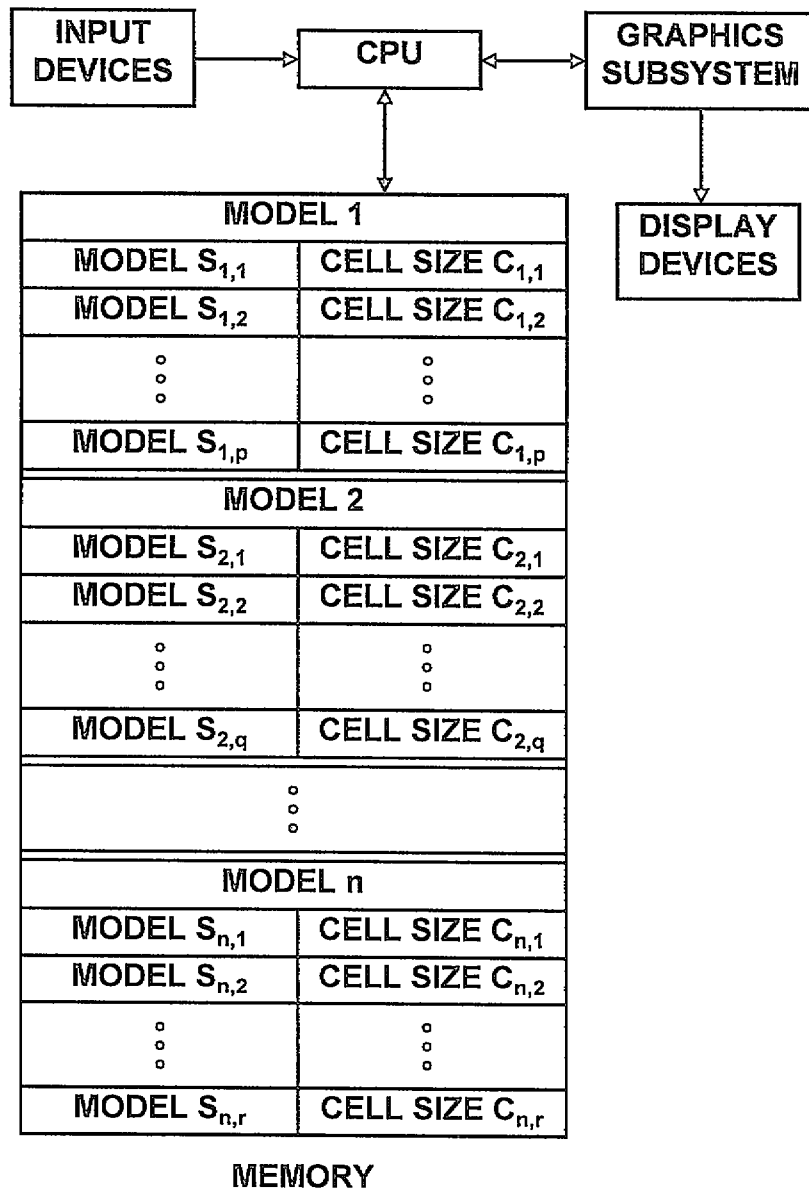


FIG. 3

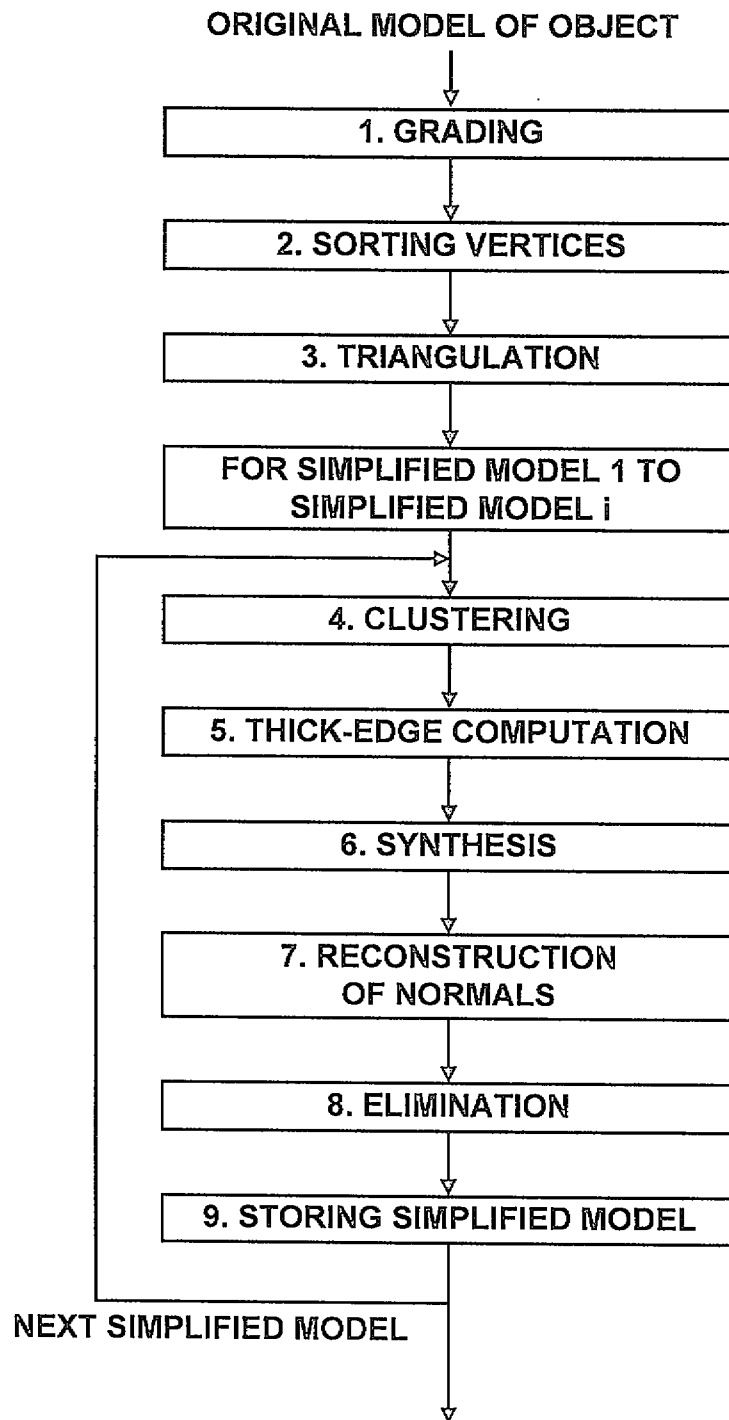


FIG 4

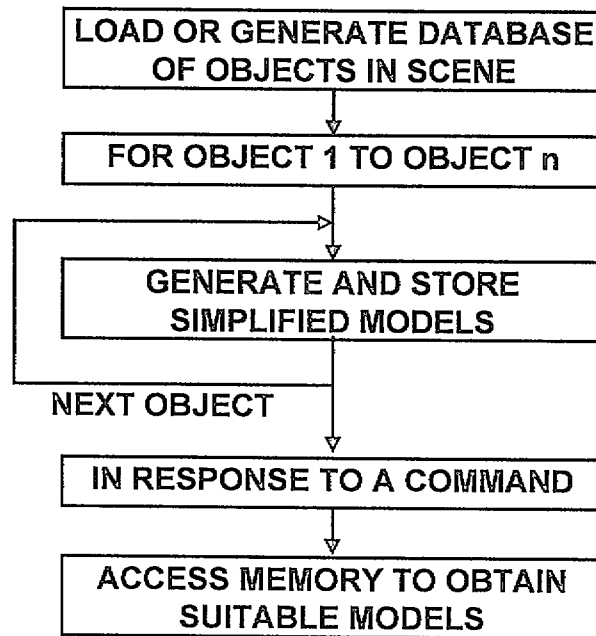


FIG. 5a

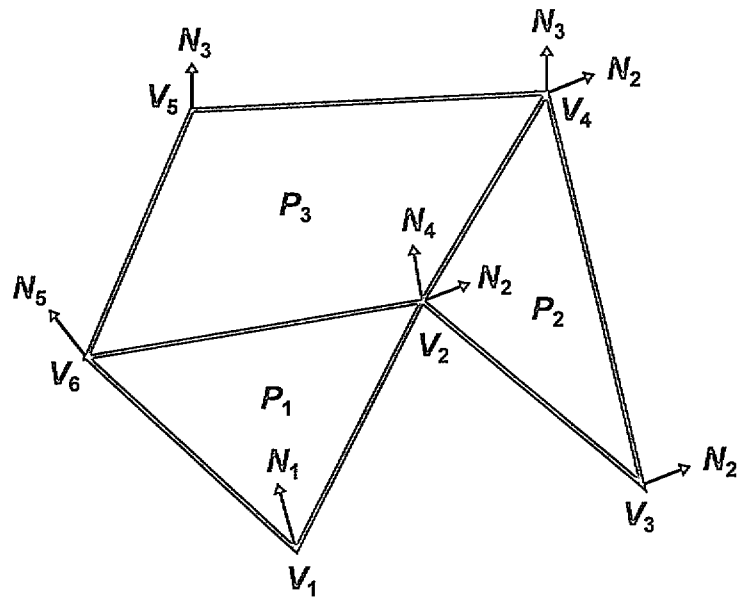


FIG. 5b

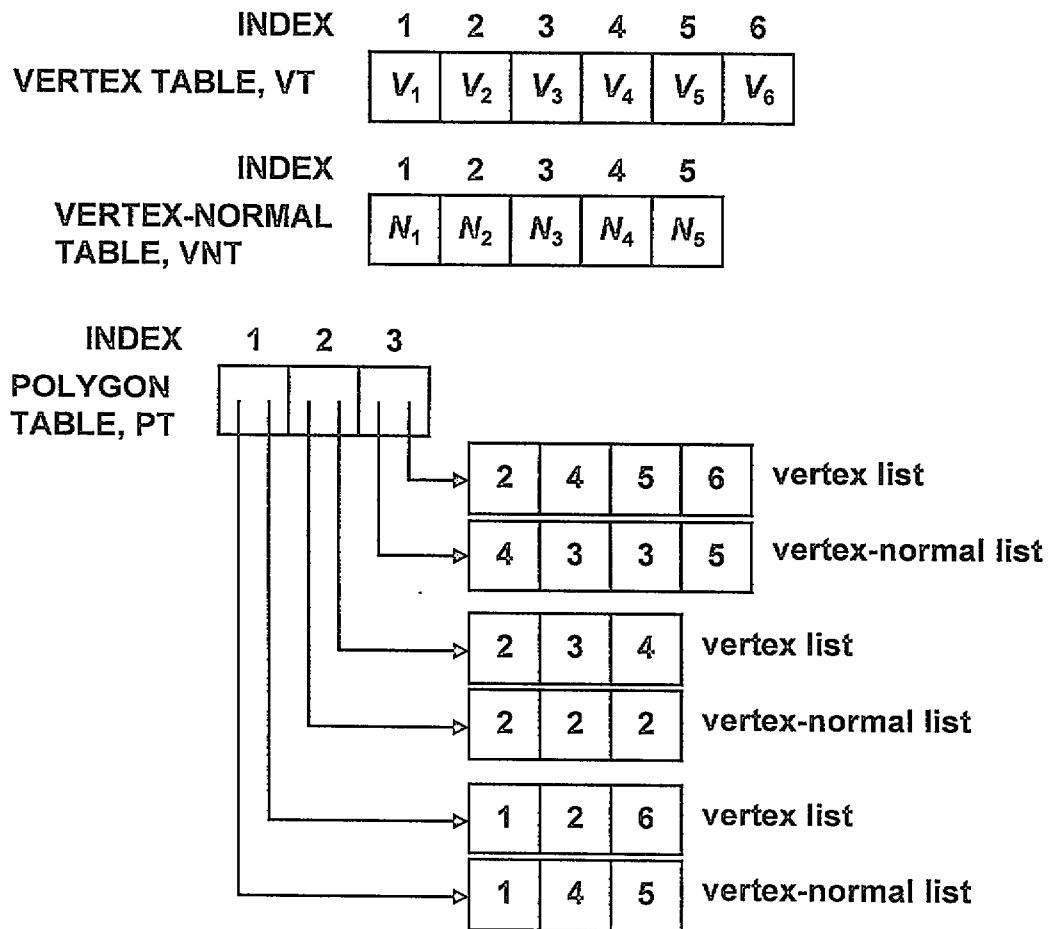


FIG. 6

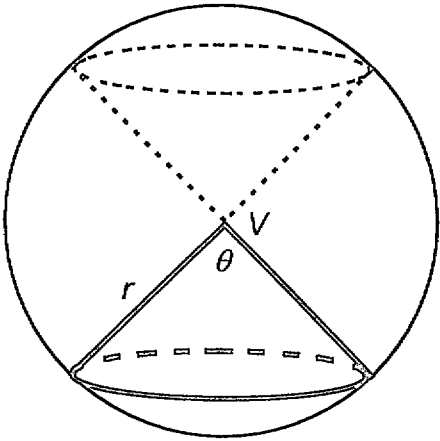


FIG. 7a

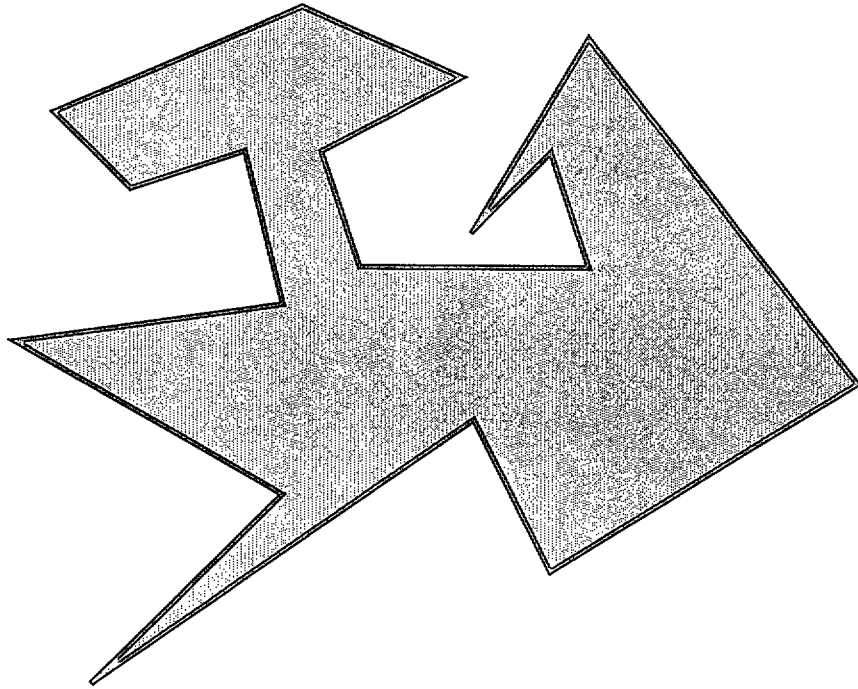


FIG. 7b

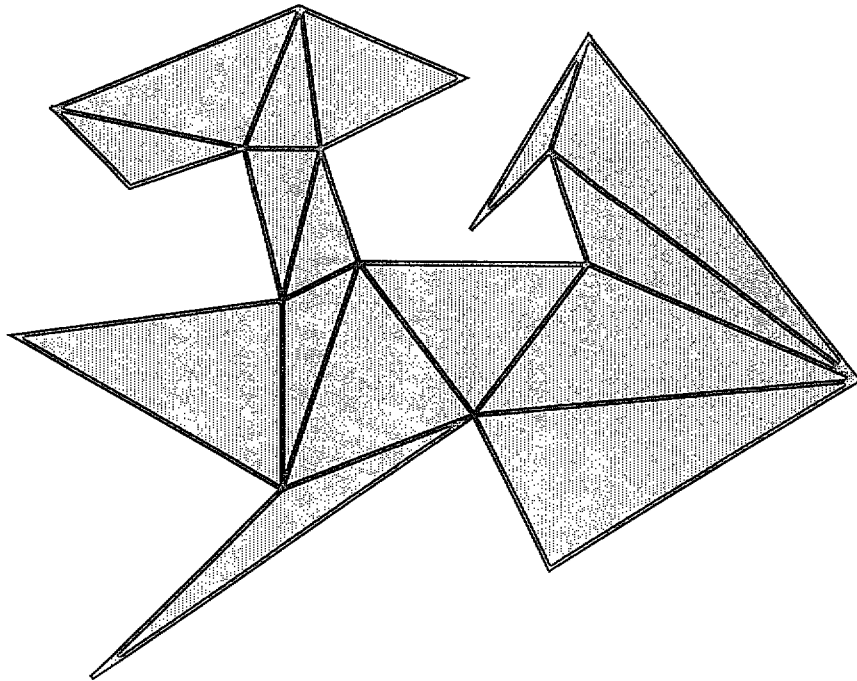


FIG. 8a

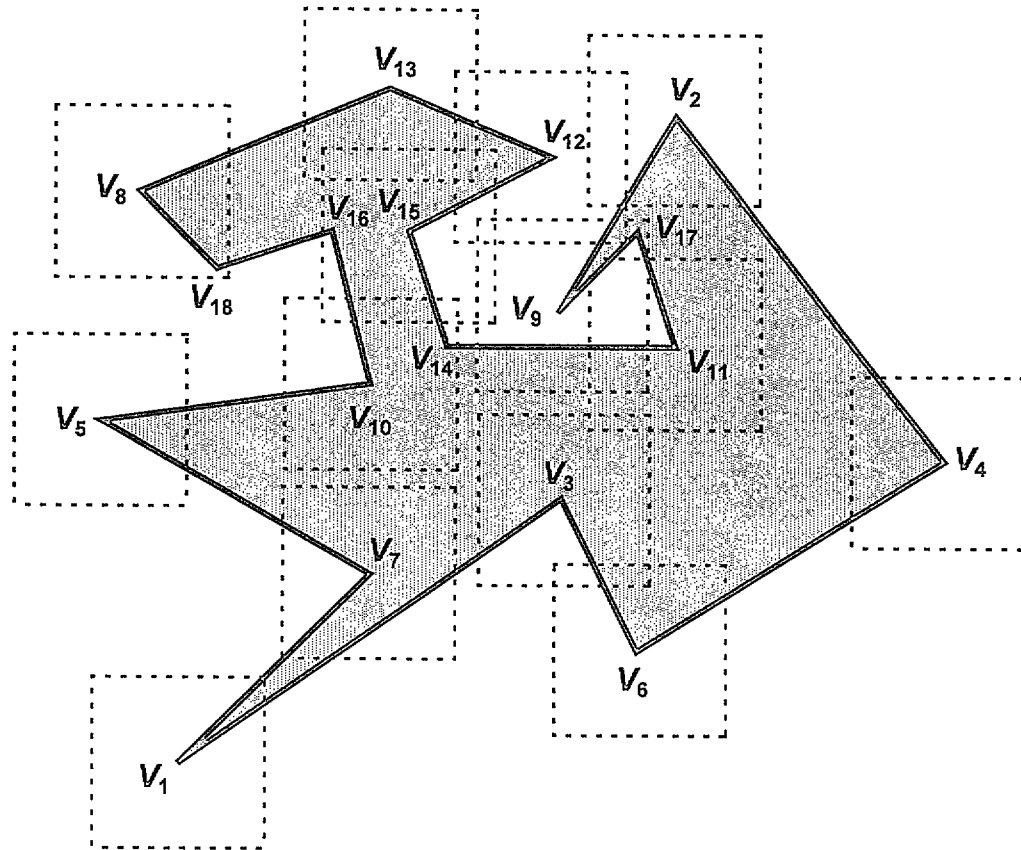


FIG. 8b

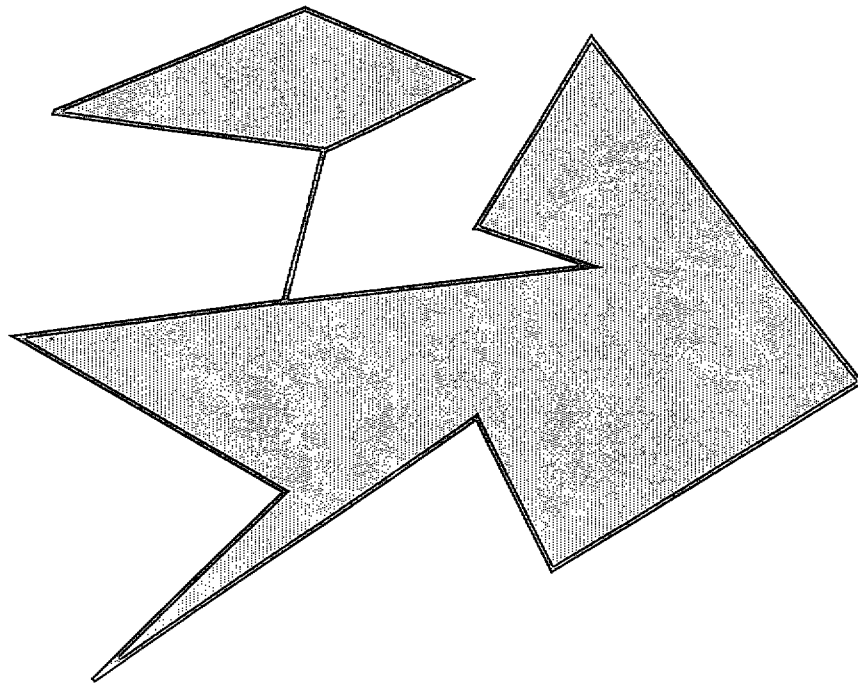


FIG. 8c

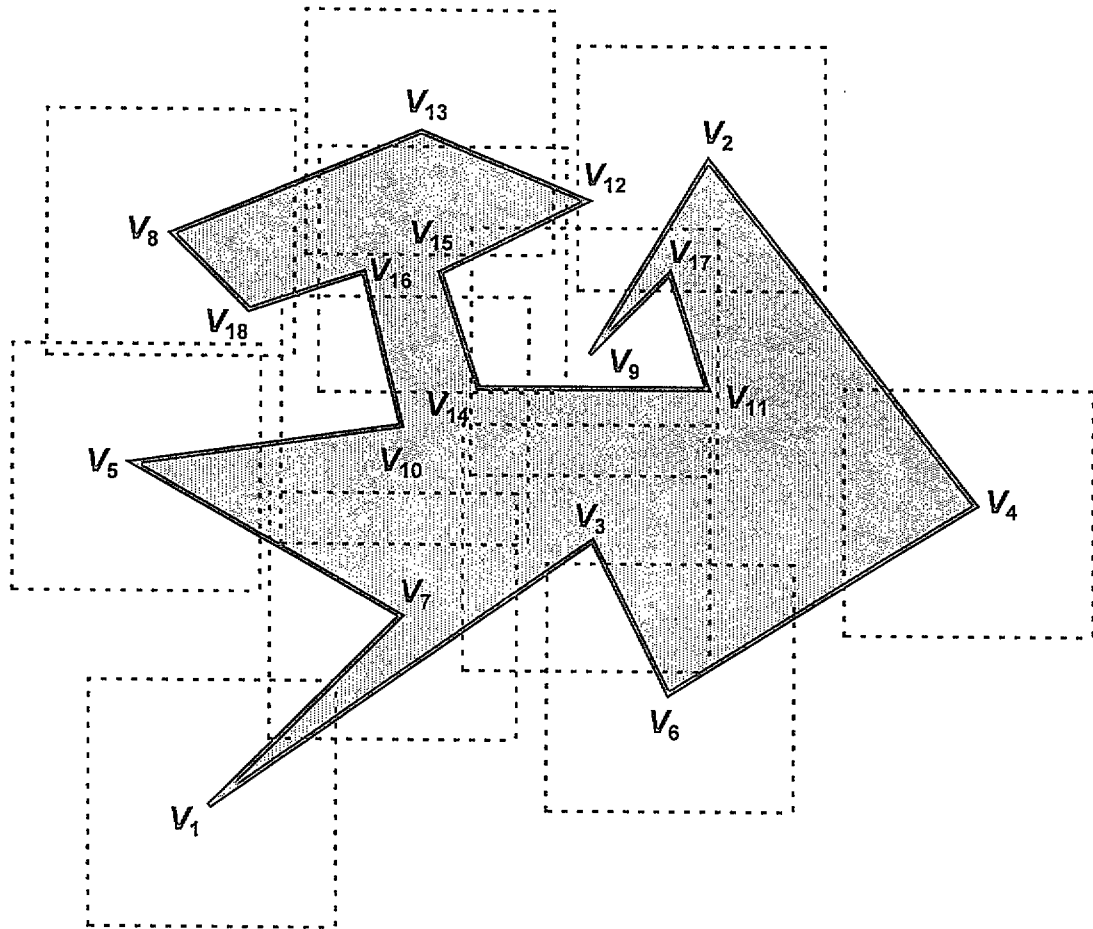


FIG. 8d

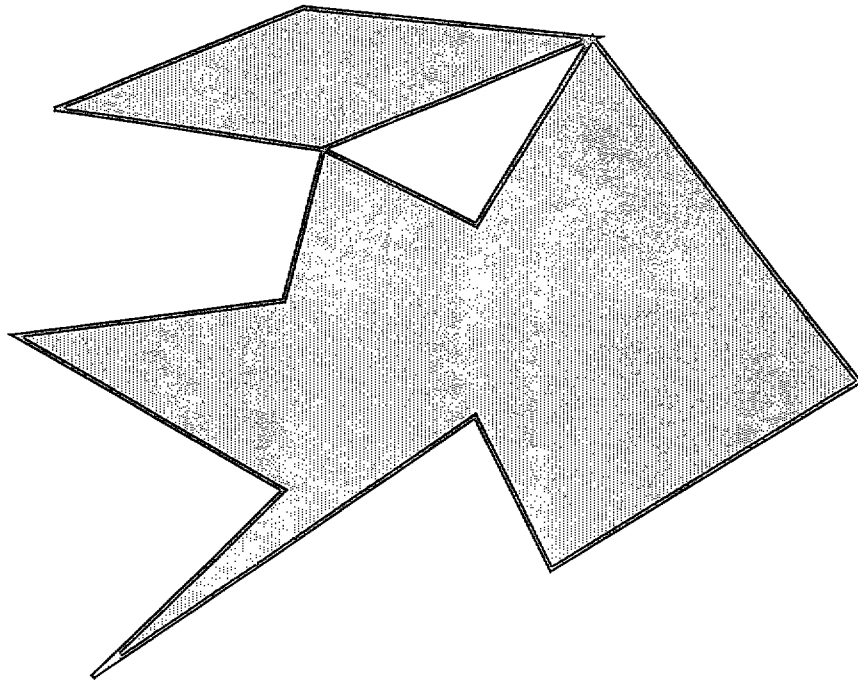
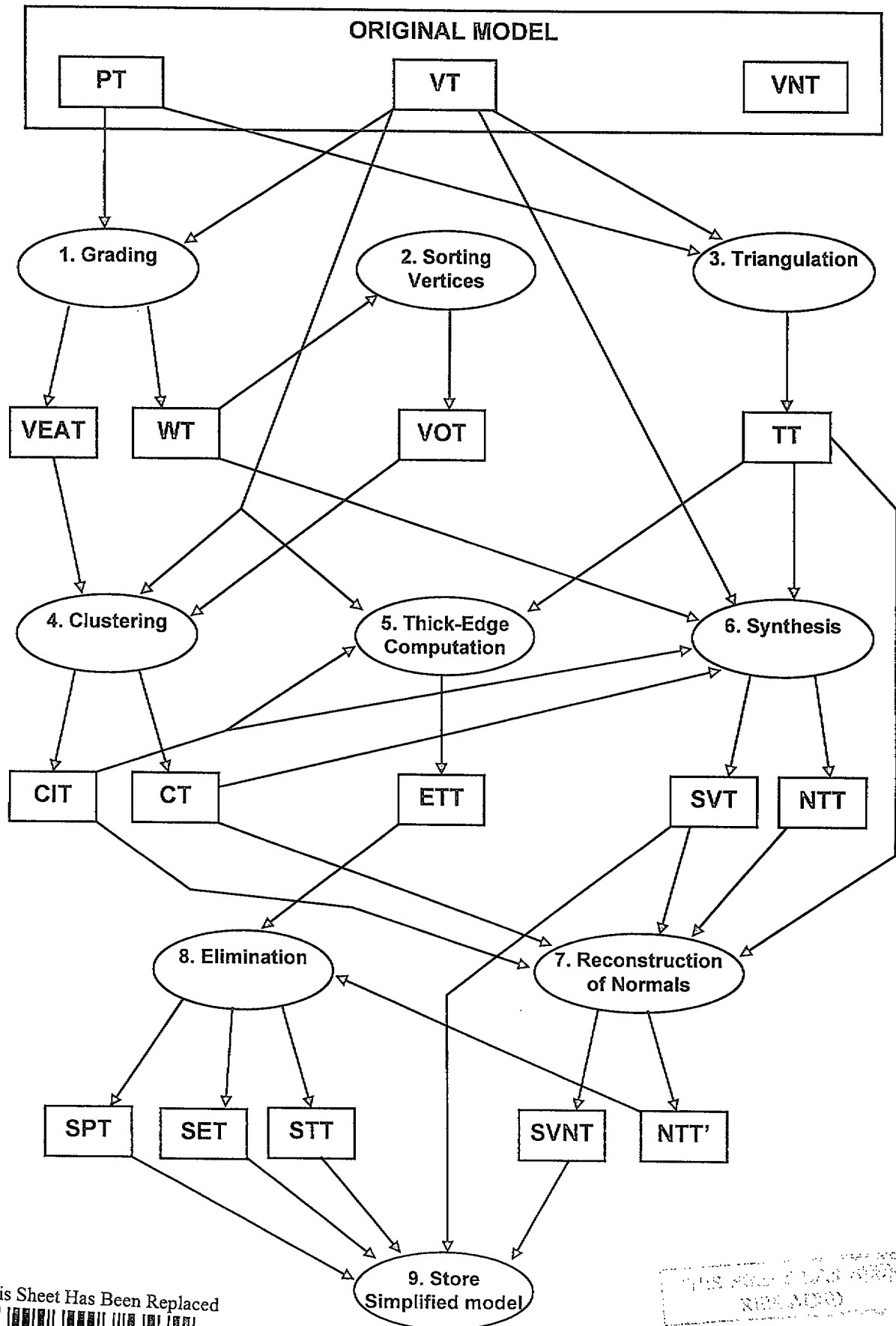


FIG. 9



7201303-1 1844

REPLACED SHEETS		SG
☐	Description : paged	to paged
☒	Claim(s) : paged <u>25</u>	to paged <u>34</u>
☐	Drawing(s) : paged	to paged
☐	Abstract : paged	to paged
DATE OF RECEIPT		

28

REPLACEMENT SHEETS		SG
☐	Description : paged	to paged
☒	Claim(s) : paged <u>25</u>	to paged <u>34</u>
☐	Drawing(s) : paged	to paged
☐	Abstract : paged	to paged
DATE OF RECEIPT <u>6/97</u>		

Thus, while the invention has been particularly shown and described with respect to preferred embodiments thereof, it will be understood by those skilled in the art that changes in form and details may be made therein without departing from the scope and spirit of the invention.

Having thus described our invention, what we claim as new, and desire to secure by Letters Patent are:

1. A method for execution by a data processor to prepare an object for display on a display means wherein the object is represented by an original model which is described by a plurality of vertices and planar faces each having boundaries defined in terms of the plurality of vertices, the method comprising the steps of:

preprocessing the original model to produce one or several simplified models with each simplified model having fewer faces and vertices than the original model;

model selecting to select either the original model or one of the simplified models for display.

2. A method as set forth in claim 1 wherein the step of preprocessing to produce the first simplified model comprises the steps of:

grading each vertex in the original model by computing a weight for the vertex to reflect its visual importance;

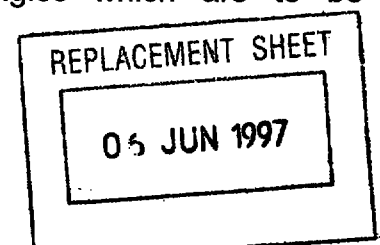
sorting vertices in the original model by ordering them according to the order of their weights;

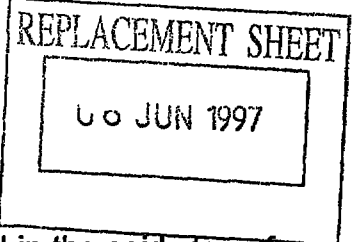
triangulating each face in the original model by partitioning it into one or more non-overlapping triangles;

clustering vertices in the original model by using a first plurality of clustering cells to group vertices into a plurality of clusters;

computing the thickness of each group of triangles which are to be degenerated into the same edge;

This Sheet Has Been Replaced





synthesizing a representative vertex for each cluster created in the said step of clustering to replace all the vertices in the cluster, and thus simplifying some triangles into edges and points;

reconstructing vertex-normals for each new triangles;

eliminating duplicated triangles, edges and points in the simplified model; and

storing the first simplified model for subsequent use.

3. A method as set forth in claim 2 wherein the step of grading includes a step of using the value of $\cos(\theta/2)$ as the estimate, for each vertex, of the probability of seeing the vertex on the silhouettes of the object from any arbitrary viewing direction, where θ is the maximum angle between all pairs of edges incident to the vertex.

4. A method as set forth in claim 2 wherein the step of grading includes a step of using the product of L and $\cos(\theta/2)$ as the weight of each vertex where L is the length of the longest among all of the edges incident to the vertex and θ is the maximum angle between all pairs of edges incident to the vertex.

5. A method as set forth in claim 2 wherein the step of clustering comprises:

letting the candidate vertex V be the most highly weighted vertex in the original model which has not been assigned to any cell,

determining if V does not lie in any existing cell, then creating a new cell centered at V and assigning V to this new cell,

else if V lies in exactly one existing cell, then assigning V to the cell,

else if V lies in two or more existing cells, assigning V to one of these cells according to a cell selection criterion, and

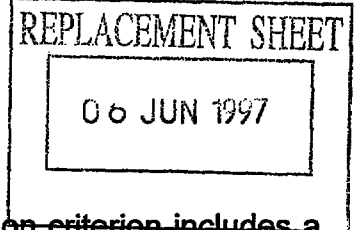
repeating the above step if necessary, until all the vertices have been assigned to some cells where the vertices assigned to each cell form a cluster.

6. A method as set forth in claim 5 wherein the cell selection criterion includes a step of selecting the cell with the most highly weighted center vertex.



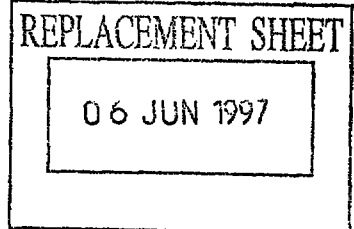
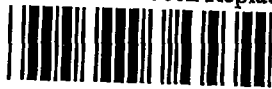
This Sheet Has Been Replaced

A standard 1D barcode located at the bottom right of the page.



7. A method as set forth in claim 5 wherein the cell selection ~~criterion includes a~~ step of selecting the cell whose center vertex is nearest to the candidate vertex.
8. A method as set forth in claim 5 wherein the cell selection criterion includes a step of selecting the cell *C* whose average length of the edges incident to the candidate vertex to the vertices already assigned to *C* is the shortest.
9. A method as set forth in claim 5 wherein the cell selection criterion includes a step of selecting the cell which has the largest number of vertices adjacent to the candidate vertex.
10. A method as set forth in claim 5 wherein the step of identifying the cells the candidate vertex lies in includes the step of using a plurality of three-dimensional grids to partition the object space, so as to identify the cells the candidate vertex lies in.
11. A method as set forth in claim 10 wherein the plurality of the three dimensional grids containing some vertices are stored in a hash table.
12. A method as set forth in claim 2 wherein the step of clustering, in another embodiment, comprises:
- letting the candidate vertex *V* be the most highly weighted vertex in the original model which has not been visited,
 - determining if *V* does not lie in any existing cell, then creating a new cell centered at *V* and assigning *V* to this new cell,
 - repeating the above steps when there is still unvisited vertex in the original model, thereafter
 - for each vertex *V* in the original model which has not been assigned to any cell, assigning it to the cell whose center vertex is nearest to *V*,
- so that all the vertices assigned to a cell form a cluster.
13. A method as set forth in claim 12 wherein the step of identifying the cells the





candidate vertex lies in includes the step of using a plurality of three-dimensional grids to partition the object space, so as to identify the cells the candidate vertex lies in.

14. A method as set forth in claim 13 wherein the plurality of the three dimensional grids containing some vertices are stored in a hash table.

15. A method as set forth in claim 2 wherein the step of computing the thickness includes a step of finding each cluster-pair which contains vertices which are used by a group of triangles that are to be degenerated into the same edge.

16. A method as set forth in claim 15 wherein those cluster-pairs containing some triangles are stored in a hash table.

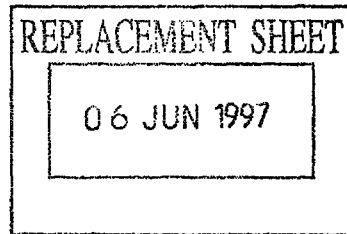
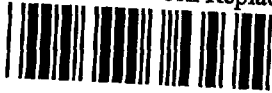
17. A method as set forth in claim 2 wherein the step of computing the thickness includes a step of estimating the thickness of each group of triangles in a cluster-pair as a function of the perpendicular distances of all the vertices of these triangles to the line joining the two average positions of these vertices in the two clusters of the cluster-pair.

18. A method as set forth in claim 2 wherein the step of synthesizing includes a step of selecting the center of the clustering cell as the position of the representative vertex.

19. A method as set forth in claim 2 wherein the step of synthesizing includes a step of computing the weighted-average position of the vertices within a cluster, and a step of selecting the weighted-average position as a position of the representative vertex.

20. A method as set forth in claim 2 wherein the step of synthesizing includes a step of computing the weighted-average position of the vertices within a cluster, and a step of selecting as the representative vertex the vertex in the cluster that is





nearest to the weighted-average position.

21. A method as set forth in claim 2 wherein the step of reconstructing vertex-normals includes the steps of:

constructing the vertex-normal information of each vertex in the original model;
updating the vertex-normal information to derive the vertex-normal information of representative vertices; and
computing new vertex-normals for each new triangle using the vertex-normal information of representative vertices.

22. A method as set forth in claim 2 wherein the step of eliminating includes a step of identifying duplications of triangles, edges and points with hash table.

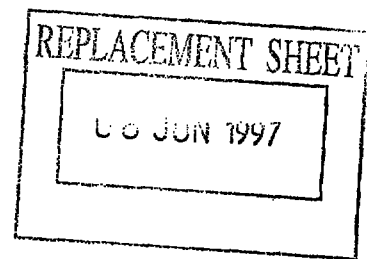
23. A method as set forth in claim 2 further including the steps, as set forth in the preceding claims, of:

clustering vertices in the original model by using a second plurality of clustering cells, wherein said second plurality of clustering cells has a larger dimension than a corresponding dimension of said first plurality of clustering cells, to group vertices into a plurality of clusters;
computing the thickness of each group of triangles which are to be degenerated into the same edge;
synthesizing a representative vertex for each cluster created in the step of clustering to replace all the vertices in the cluster, and thus simplifying some triangles into edges and points;
reconstructing vertex-normals for each new triangles;
eliminating duplicated triangles, edges and points in the simplified model; and
storing the second simplified model for subsequent use.

24. A method as set forth in claim 23 wherein the steps of:

clustering vertices in the original model;
computing the thickness of each group of triangles which are to be





degenerated into the same edge;
synthesizing a representative vertex for each cluster created in the step of clustering to replace all the vertices in the cluster;
reconstructing vertex-normals for each new triangles;
eliminating duplicated triangles, edges and points in the simplified model, and
storing the second simplified model for subsequent use;
may be repeated as many times as needed provided each time the clustering step is done using a plurality of clustering cells with dimension larger than the preceding one.

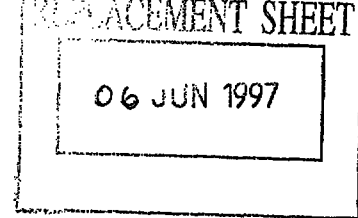
25. A method as set forth in claim 1 wherein the step of model selecting includes a step of determining the object space covered by a pixel on the display means, and wherein the step of model selecting selects either the original model or one of the simplified models as a function of at least the determined size of the object space covered by a pixel, and the size of the clustering cells used in generating each of the simplified models.

26. A method as set forth in claim 1 further including the step of computing the pixel-width of each thick-line primitive which renders a thick-edge, during display of the object using one of the simplified models.

27. A method as set forth in claim 1 further including the step of computing the dynamic normal for each thick-line primitive which renders a thick-edge, during display of the object using one of the simplified models.

28. Graphics display apparatus for displaying at least one object, comprising:
memory means for storing an original model of an object and for storing at least one simplified model of the object, the original model having a first plurality of vertices and at least one simplified model having a second plurality of vertices that is less than the first plurality of vertices;
means, responsive to a command from a user of the apparatus, for indicating that a motion of displayed object is to occur; and





processing means, coupled to and responsive to the indicating means, for accessing the memory means to select for display either the original model of the object or one of the simplified models of the object.

29. Graphics display apparatus as set forth in claim 28 wherein said processing means includes means for preprocessing the original model of the object, as set forth in claim 2, claim 23 and claim 24 so as to produce said at least one simplified model.

30. Graphics display apparatus as set forth in claim 29 and further including means for storing an array data structure containing vertices of each model.

31. Graphics display apparatus as set forth in claim 28 and further including means for storing clustering cell sizes used in generating the said models, and wherein said processing means is responsive to a determined size of object space covered by a pixel on the display means, and to said stored clustering cell sizes, to select one of the said models for display.

32. Apparatus for displaying an object represented by an original model which is described by a plurality of vertices and planar faces each having boundaries defined in terms of the plurality of vertices, comprising:

- means for preprocessing the original model to produce at least one simplified model with each simplified model having fewer faces and vertices than the original model; and
- means for model selecting to select either the original model or one of the simplified models for display.



REPLACEMENT SHEETS		SG
☒	Description	: paged to paged
☒	Claim(s)	: paged ~~31~~ to paged ~~34~~
☒	Drawing(s)	: paged to paged
☒	Abstract	: paged to paged
CLAIMS		
DATE OF RECEIPT 7/1/97		

02 JUL 1997

Thus, while the invention has been particularly shown and described with respect to preferred embodiments thereof, it will be understood by those skilled in the art that changes in form and details may be made therein without departing from the scope and spirit of the invention.

THIS SHEET HAS BEEN REPLACED

Having thus described our invention, what we claim as new, and desire to secure by Letters Patent are:

1. A method for execution by a data processor to prepare an object for display on a display means wherein the object is represented by an original model which is described by a plurality of vertices and planar faces each having boundaries defined in terms of the plurality of vertices, the method comprising the steps of:

preprocessing the original model to produce one or several simplified models with each simplified model having fewer faces and vertices than the original model;

model selecting to select either the original model or one of the simplified models for display.

2. A method as set forth in claim 1 wherein the step of preprocessing to produce the first simplified model comprises the steps of:

grading each vertex in the original model by computing a weight for the vertex to reflect its visual importance;

sorting vertices in the original model by ordering them according to the order of their weights;

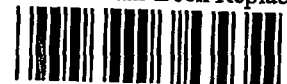
triangulating each face in the original model by partitioning it into one or more non-overlapping triangles;

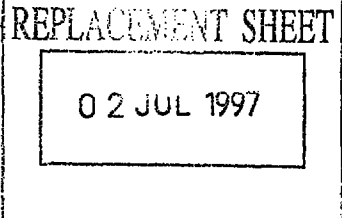
clustering vertices in the original model by using a first plurality of clustering cells to group vertices into a plurality of clusters;

computing the thickness of each group of triangles which are to be degenerated into the same edge;



This Sheet Has Been Replaced





synthesizing a representative vertex for each cluster created in the said step of clustering to replace all the vertices in the cluster, and thus simplifying some triangles into edges and points;

reconstructing vertex-normals for each new triangles;

eliminating duplicated triangles, edges and points in the simplified model; and

storing the first simplified model for subsequent use.

3. A method as set forth in claim 2 wherein the step of grading includes a step of using the value of $\cos(\theta/2)$ as the estimate, for each vertex, of the probability of seeing the vertex on the silhouettes of the object from any arbitrary viewing direction, where θ is the maximum angle between all pairs of edges incident to the vertex.

4. A method as set forth in claim 2 wherein the step of grading includes a step of using the product of L and $\cos(\theta/2)$ as the weight of each vertex where L is the length of the longest among all of the edges incident to the vertex and θ is the maximum angle between all pairs of edges incident to the vertex.

5. A method as set forth in claim 2 wherein the step of clustering comprises:

letting the candidate vertex V be the most highly weighted vertex in the original model which has not been assigned to any cell,

determining if V does not lie in any existing cell, then creating a new cell centered at V and assigning V to this new cell,

else if V lies in exactly one existing cell, then assigning V to the cell,

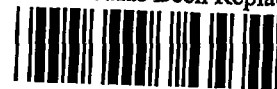
else if V lies in two or more existing cells, assigning V to one of these cells according to a cell selection criterion, and

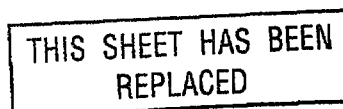
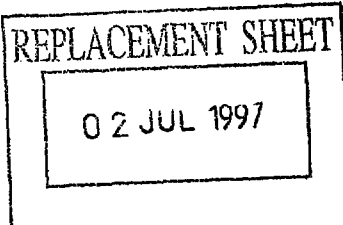
repeating the above step if necessary, until all the vertices have been assigned to some cells where the vertices assigned to each cell form a cluster.

6. A method as set forth in claim 5 wherein the cell selection criterion includes a step of selecting the cell with the most highly weighted center vertex.



This Sheet Has Been Replaced





7. A method as set forth in claim 5 wherein the cell selection criterion includes a step of selecting the cell whose center vertex is nearest to the candidate vertex.

8. A method as set forth in claim 5 wherein the cell selection criterion includes a step of selecting the cell *C* whose average length of the edges incident to the candidate vertex to the vertices already assigned to *C* is the shortest.

9. A method as set forth in claim 5 wherein the cell selection criterion includes a step of selecting the cell which has the largest number of vertices adjacent to the candidate vertex.

10. A method as set forth in claim 5 wherein the step of identifying the cells the candidate vertex lies in includes the step of using a plurality of three-dimensional grids to partition the object space, so as to identify the cells the candidate vertex lies in.

11. A method as set forth in claim 10 wherein the plurality of the three dimensional grids containing some vertices are stored in a hash table.

12. A method as set forth in claim 2 wherein the step of clustering, in another embodiment, comprises:

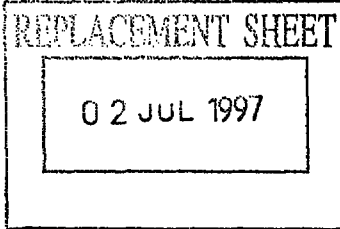
 letting the candidate vertex *V* be the most highly weighted vertex in the original model which has not been visited,
 determining if *V* does not lie in any existing cell, then creating a new cell centered at *V* and assigning *V* to this new cell,
 repeating the above steps when there is still unvisited vertex in the original model, thereafter
 for each vertex *V* in the original model which has not been assigned to any cell, assigning it to the cell whose center vertex is nearest to *V*,
so that all the vertices assigned to a cell form a cluster.

13. A method as set forth in claim 12 wherein the step of identifying the cells the



This Sheet Has Been Replaced





candidate vertex lies in includes the step of using a plurality of three-dimensional grids to partition the object space, so as to identify the cells the candidate vertex lies in.

14. A method as set forth in claim 13 wherein the plurality of the three dimensional grids containing some vertices are stored in a hash table.

15. A method as set forth in claim 2 wherein the step of computing the thickness includes a step of finding each cluster-pair which contains vertices which are used by a group of triangles that are to be degenerated into the same edge.

16. A method as set forth in claim 15 wherein those cluster-pairs containing some triangles are stored in a hash table.

17. A method as set forth in claim 2 wherein the step of computing the thickness includes a step of estimating the thickness of each group of triangles in a cluster-pair as a function of the perpendicular distances of all the vertices of these triangles to the line joining the two average positions of these vertices in the two clusters of the cluster-pair.

18. A method as set forth in claim 2 wherein the step of synthesizing includes a step of selecting the center of the clustering cell as the position of the representative vertex.

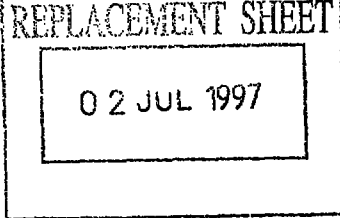
19. A method as set forth in claim 2 wherein the step of synthesizing includes a step of computing the weighted-average position of the vertices within a cluster, and a step of selecting the weighted-average position as a position of the representative vertex.

20. A method as set forth in claim 2 wherein the step of synthesizing includes a step of computing the weighted-average position of the vertices within a cluster, and a step of selecting as the representative vertex the vertex in the cluster that is



This Sheet Has Been Replaced





THIS SHEET HAS BEEN
REPLACED

nearest to the weighted-average position.

21. A method as set forth in claim 2 wherein the step of reconstructing vertex-normals includes the steps of:

constructing the vertex-normal information of each vertex in the original model;
updating the vertex-normal information to derive the vertex-normal information of representative vertices; and
computing new vertex-normals for each new triangle using the vertex-normal information of representative vertices.

22. A method as set forth in claim 2 wherein the step of eliminating includes a step of identifying duplications of triangles, edges and points with hash table.

23. A method as set forth in claim 2 further including the steps, as set forth in the preceding claims, of:

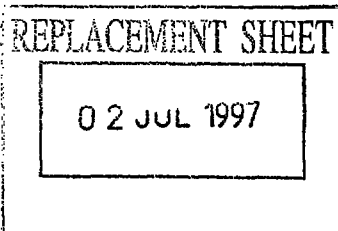
clustering vertices in the original model by using a second plurality of clustering cells, wherein said second plurality of clustering cells has a larger dimension than a corresponding dimension of said first plurality of clustering cells, to group vertices into a plurality of clusters;
computing the thickness of each group of triangles which are to be degenerated into the same edge;
synthesizing a representative vertex for each cluster created in the step of clustering to replace all the vertices in the cluster, and thus simplifying some triangles into edges and points;
reconstructing vertex-normals for each new triangles;
eliminating duplicated triangles, edges and points in the simplified model; and
storing the second simplified model for subsequent use.

24. A method as set forth in claim 23 wherein the steps of:

clustering vertices in the original model;
computing the thickness of each group of triangles which are to be

This Sheet Has Been Replaced





degenerated into the same edge;
synthesizing a representative vertex for each cluster created in the step of clustering to replace all the vertices in the cluster;
reconstructing vertex-normals for each new triangles;
eliminating duplicated triangles, edges and points in the simplified model, and
storing the second simplified model for subsequent use;
may be repeated as many times as needed provided each time the clustering step is done using a plurality of clustering cells with dimension larger than the preceding one.

25. A method as set forth in claim 1 wherein the step of model selecting includes a step of determining the object space covered by a pixel on the display means, and wherein the step of model selecting selects either the original model or one of the simplified models as a function of at least the determined size of the object space covered by a pixel, and the size of the clustering cells used in generating each of the simplified models.

26. A method as set forth in claim 1 further including the step of computing the pixel-width of each thick-line primitive which renders a thick-edge, during display of the object using one of the simplified models.

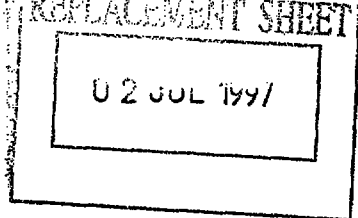
27. A method as set forth in claim 1 further including the step of computing the dynamic normal for each thick-line primitive which renders a thick-edge, during display of the object using one of the simplified models.

28. Graphics display apparatus for displaying at least one object, comprising:
memory means for storing an original model of an object and for storing at least one simplified model of the object, the original model having a first plurality of vertices and at least one simplified model having a second plurality of vertices that is less than the first plurality of vertices;
means, responsive to a command from a user of the apparatus, for indicating that a motion of displayed object is to occur; and



This Sheet Has Been Replaced





THIS SHEET HAS BEEN
REPLACED

34

processing means, coupled to and responsive to the indicating means, for accessing the memory means to select for display either the original model of the object or one of the simplified models of the object.

29. Graphics display apparatus as set forth in claim 28 wherein said processing means includes means for preprocessing the original model of the object, as set forth in claim 2, claim 23 and claim 24 so as to produce said at least one simplified model.

30. Graphics display apparatus as set forth in claim 29 and further including means for storing an array data structure containing vertices of each model.

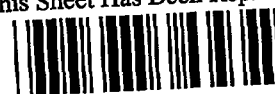
31. Graphics display apparatus as set forth in claim 28 and further including means for storing clustering cell sizes used in generating the said models, and wherein said processing means is responsive to a determined size of object space covered by a pixel on the display means, and to said stored clustering cell sizes, to select one of the said models for display.

32. Apparatus for displaying an object represented by an original model which is described by a plurality of vertices and planar faces each having boundaries defined in terms of the plurality of vertices, comprising:

means for preprocessing the original model to produce at least one simplified model with each simplified model having fewer faces and vertices than the original model; and

means for model selecting to select either the original model or one of the simplified models for display.

This Sheet Has Been Replaced



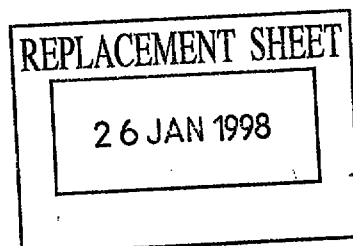
* C L M S *

ABSTRACT**MULTI-RESOLUTION GRAPHIC REPRESENTATION GENERATED BY
WEIGHT-CONTROLLED VERTEX CLUSTERING FOR INTERACTIVE
VISUALIZATION APPLICATIONS**

A method for execution by a data processor that prepares an object for display during interactive visualization. The method includes a step of preprocessing an original model, which is a polygonal boundary representation of the object, so as to produce a plurality of simplified models with each simplified model having fewer faces and vertices than the original model. The step of preprocessing includes the steps of grading, sorting vertices, triangulation, clustering, thick-edge computation, synthesis, reconstruction of normals, elimination, and storing the simplified model. During display of the object, the method further includes a step of selecting either the original model or one of the simplified models for display so as to achieve real-time interactive motion of the object.

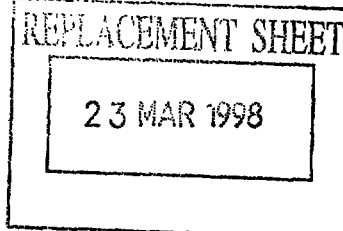
Fig 8a & Fig.8b are to be published.

THIS SHEET HAS BEEN
REPLACED



This Sheet Has Been Replaced

A standard 1D barcode with vertical black bars of varying widths on a white background.



9701303-1 1803

1

**MULTI-RESOLUTION GRAPHIC REPRESENTATION GENERATED BY
WEIGHT-CONTROLLED VERTEX CLUSTERING FOR INTERACTIVE
VISUALIZATION APPLICATIONS**

FIELD OF THE INVENTION

The present invention relates generally to computer graphics and solid modeling, and more particularly, to methods and apparatus for representing and displaying an object at multiple levels of resolution.

BACKGROUND OF THE INVENTION

REPLACEMENT SHEETS		SG
☒ Description	: paged 1.. to paged 30..	
☒ Claim(s)	: paged 31.. to paged 32..	
☒ Drawing(s)	: paged 33.. to paged 34..	
☒ Abstract	: paged 35.. to paged 36..	
DATE OF RECEIPT		23/3/98

In traditional modeling systems, detailed models are created by applying versatile modeling operations (such as extrusion, constructive solid geometry, and freeform deformations) to a vast array of geometric primitives. For display, these models must usually be tessellated into polygonal approximations. Each of these polygonal approximations typically contain millions of polygons. Enormous computing power is required to render such complex polygonal surface models, in real-time and at interactive frame rates.

In interactive visualization applications, real-time interactive manipulation of viewing conditions is vital to the understanding of an object's geometry and its relative position in the virtual scene. The delays in real-time graphic feedback and the jerky object motion will lead to overshooting, reduced feeling of control, and thus make the user less productive.

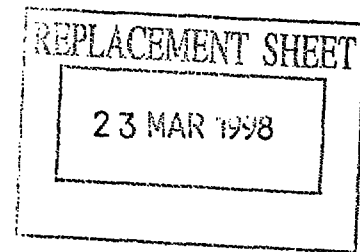
In order to accommodate complex polygonal models while still maintaining real-time interactive frame rates, methods for approximating the polygonal models and using multi-resolution models have been proposed. Polygonal model simplification algorithms can be used to generate multiple polygonal models at varying levels of

detail, and techniques are employed by the display system to select the appropriate level of detail and render it appropriately. These approximations must resemble the original models from all directions, and the transitions between one approximation and the next must be barely noticeable in order to effectively reveal details as objects approach the viewpoint.

One of these model simplification methods is disclosed in U.S. Pat. No. 5,448,686, "Multi-Resolution Graphic Representation Employing At Least One Simplified Model For Interactive Visualization Applications" issued to Borrel and Rossignac. Their method is also described in their paper titled "Multi-Resolution 3D Approximations for Rendering Complex Scenes", Modeling in Computer Graphics, B. Falcidiena and T. L. Kunii, Eds. Springer-Verlag, pp. 455-465, 1993. This method is very fast in generating simplified models compared to other polygonal simplification methods, and it is effective in achieving high data reduction rates. As such, it is very suitable for use in large Computer Aided Design (CAD) applications where designs are frequently edited and assemblies usually comprise hundreds of thousands of parts. However, the models it produces may be visually very different from the original models, and, moreover, the transitions between different levels of simplification during interactive viewing may not be satisfactory.

It is thus one object of this invention to provide a method for simplifying a polygonal surface model of an object such that (a) the simplified models have image fidelity close to the original model when displayed on the display screen, (b) the transitions between one level of simplification to the next are barely noticeable when the object is displayed using the simplified models, (c) the simplification process requires low computational cost, and (d) the method is effective in achieving high data reduction rates.

It is another object of this invention to provide methods and apparatus that employ one or more simplified object models for display to a viewer, the simplified models being stored and accessed as required. The aim is to provide real-time interactive frame rates during interactive visualization of the objects in the scene.



SUMMARY OF THE INVENTION

The invention described herein satisfies the above-identified needs and overcomes the limitations of the prior invention by Rossignac and Borrel. The present invention provides a method to construct a sequence of graphic models for an object, one for each desired level of simplification. Then, as the object moves in the object space during interactive viewing, the method successively switches to graphic models of the object having the most appropriate simplification resolutions. This results in a suppression of image detail, consequently with an improvement in real-time performance. The inventors have determined that many details in a displayed object can be eliminated, without a loss of visual information.

The method applies a basic principle both in the step of model simplification and the step of selecting the most appropriate model (either the original model or one of the simplified models) for displaying the object during interactive visualization.

The basic principle employed by the method is based on the following argument. In a synthetic scene, when an object is far away from the viewpoint, its image size is small. Due to the discreteness of the image space, many points on the object are mapped onto the same pixel, and this happens often when the object's model is complex and the image size is relatively small. For points mapped to the same pixel, only one point appears on the image at the pixel, and the others are eliminated by hidden-surface removal. This is a wastage in processing as many of such points are processed but never make their way to the final image. A potential solution to cutting down this wasteful processing is to identify those points that are going to fall onto the same pixel and use a new point to represent them. Only this new point is sent for rendering. Points that are most likely to be mapped to the same pixel, regardless of the direction of the viewpoint, are those points which are geometrically very close to one another in the object space.

The model simplification step of the method applies the above principle by using three-dimensional cells to identify vertices which are geometrically close to one another. These cells are placed in the object space to group vertices into clusters. Vertices in the same cluster are considered close to one another, and a new representative vertex is then created to replace these vertices in the cluster. Indirectly, determining the closeness of the vertices also helps to determine the closeness of the polygons. For example, two rectangles are close together if their corresponding vertices are close to each other. By using different clustering cell sizes, the definition of "closeness" will be different, and this allows the original model to be simplified into models of different levels of detail.

During display of the object, the size of the object space covered by a pixel on the display screen is determined, and the model selection step of the method selects the most appropriate model of the object for display. The model selection criterion is based on the determined size of the object space covered by a pixel, and the size of the clustering cells used in generating each of the simplified models.

More specifically, there is disclosed a method for execution by a data processor that prepares an object for display during interactive visualization. The method includes a step of preprocessing an original model, which is a polygonal boundary representation of the object, so as to produce a plurality of simplified models with each simplified model having fewer faces and vertices than the original model. The step of preprocessing includes the following steps:

(1) Grading—a weight is computed for each vertex in the original model to reflect the vertex's visual importance. In one embodiment, the value of the vertex's weight is a function of the angles between all pairs of edges incident to the vertex, and the lengths of these incident edges.

(2) Sorting vertices—the vertices in the original model need to be sorted in non-increasing order of their weights.

(3) Triangulation—polygons in the original model are decomposed into non-overlapping triangles without introducing new vertices into the original model. The graded and triangulated original model needs to be kept for subsequent generation of other simplified models.

(4) Clustering—vertices are grouped into clusters based on geometric proximity. The positions of the clustering cells are based on the positions of the original vertices and the weights of the vertices.

(5) Thick-edge computation—for each group of triangles which are going to degenerate into the same edge (due to Step (6) — synthesis), compute an estimate for the thickness of the group of triangles.

(6) Synthesis—a representative vertex is computed to replace the vertices in each cluster and thus simplify some triangles into edges and points.

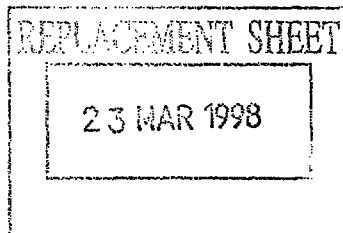
(7) Reconstruction of normals—the vertex-normals of each triangle are reconstructed.

(8) Elimination—duplicated triangles, edges and points in the simplified model are removed.

(9) Storing the simplified model for subsequent use.

Subsequent generation of other simplified models of the object requires execution of only Step (4) to Step (9) for each simplified model with a new clustering cell size for Step (4). A larger clustering cell size will generate a more simplified model (less detailed) while a smaller clustering cell size will generate a less simplified model (more detailed).

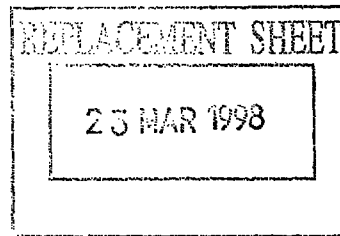
During display of the object, the method further includes a step of selecting either the original model or one of the simplified models for display. The step of selecting



a model includes a step of determining the size of the object space covered by a pixel on the display screen, and the step of selecting a model selects either the original model or one of the simplified models as a function of at least the determined size of the object space covered by a pixel, and the size of the clustering cells used in generating each of the simplified models.

The method achieves the objects of the present invention by using:

- (a) weight-controlled vertex clustering in which the positions of the clustering cells are based on the positions of the original vertices and the weights of the vertices. This helps to preserve important features of the object, improves the approximation quality, and thus leads to simplified models with higher image fidelity to the original model. Using weight-controlled vertex clustering also results in more consistent placements of important features from one simplified model to the next. The more consistent placements of important features lead to smoother transitions from one level of simplification to the next when object is displayed using the simplified models;
- (b) thick-edges to further improve the image quality of the displayed simplified model. Each thick-edge is rendered as a multi-pixel-wide thick-line graphics primitive on the display screen, and a dynamic normal is computed for the thick-line to improve its shading and intensity;
- (c) an appropriate clustering cell size to achieve the desired level of simplification and data reduction rate;
- (d) a model selection step to select either the original model or one of the simplified models, for display of the object during interactive visualization, as a function of at least the determined size of the object space covered by a pixel, and the size of the clustering cells used in generating each of the simplified models.



BRIEF DESCRIPTION OF THE DRAWING

- FIG. 1** is a block diagram of an exemplary raster graphics systems;
- FIG. 2** is a block diagram of the raster graphics system of FIG. 1 that shows a memory for storing object models and simplified versions thereof;
- FIG. 3** is a flowchart that depicts the steps of the preprocessing step of the method of the invention;
- FIG. 4** is a flowchart that depicts the operation of the method of the invention;
- FIG. 5a** shows an exemplary object model;
- FIG. 5b** illustrates the data structure used to store the model in FIG. 5a;
- FIG. 6** illustrates part of the computation used to derive the weight of a vertex;
- FIG. 7a** shows an exemplary polygon which is to be triangulated;
- FIG. 7b** shows a triangulation of the polygon in FIG. 7a;
- FIG. 8a** shows an exemplary two-dimensional object model whose vertices are being clustered by cells;
- FIG. 8b** illustrates the resulting model after synthesizing the representative vertices for the vertex clusters shown in FIG. 8a;
- FIG. 8c** shows an exemplary two-dimensional object model identical to that in FIG. 8a, except that its vertices are being clustered by larger cells;
- FIG. 8d** illustrates the resulting model after synthesizing the representative vertices for the vertex clusters shown in FIG. 8c;
- FIG. 9** is a dataflow diagram that illustrates the operation of the preprocessing step of the method of the invention.

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 illustrates an exemplary raster graphics system which includes a main (Host) processor and a graphics subsystem. The Host processor executes an application program and dispatches graphics tasks to the graphics subsystem.

The graphics subsystem includes a pipeline of several components that perform operations necessary to prepare geometric entities for display on a raster display device. For the purposes of describing the invention, a model of the graphics subsystem is employed that contains the following functional units. It should be realized that this particular model is not to be construed in a limiting sense upon the practice of the invention.

A Geometric Processor unit performs geometric and perspective transformations, exact clipping on primitives against screen (window) boundaries, as well as lighting computations. The resulting graphics primitives, e.g. points, lines, triangles, etc., are described in screen space (integral) coordinates.

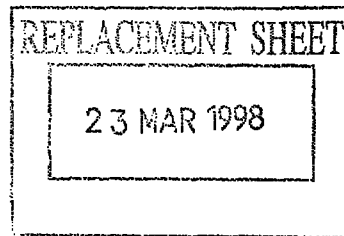
A Scan Conversion (Rasterization) unit breaks down the graphics primitives into raster information, i.e. a description of display screen pixels that are covered by the graphics primitives.

A Graphics Buffer unit receives, stores, and processes the pixels from the Scan Conversion unit in accordance with the method of the invention. The most common buffers associated to the Graphics Buffer unit are the image buffer and the z-buffer.

A Display unit receives pixels from the Graphics Buffer unit and transforms these pixels into information displayed on the output device, typically a raster screen.

Having described an exemplary graphics processing system that is suitable for use in practicing the invention, a description is now provided of a presently preferred method of model simplification.

For the purposes of describing the invention, it is assumed that objects are described in polygonal boundary representations. Models with curved surfaces are in general tessellated, i.e. approximated with planar polygons, for graphic purposes and thus also benefit from the simplification technique of the invention.



Preprocessing Step

The preprocessing step is the model simplification step. The preprocessing step constructs one or several simplified models (with decreasing level of detail) for each object in a scene to be visualized. The step of preprocessing includes the following steps: (1) grading, (2) sorting vertices, (3) triangulation, (4) clustering, (5) thick-edge computation, (6) synthesis, (7) reconstruction of normals, (8) elimination, and (9) storing the simplified model. Step (1) to Step (3) need only be executed once for an object regardless of the number of simplified models to be generated. FIG. 3 is a flowchart that depicts the flow of these steps of the preprocessing step.

For the purposes of describing the preprocessing step, it is assumed that each original model is described by a vertex table VT, a vertex-normal table VNT, and a polygon table PT. The vertex table VT contains the supporting vertices used in the model. Each entry of the vertex-normal table VNT contains a vector which will be used by some polygon vertex as the normal at the vertex (vertex-normal). The vertex-normal at each vertex of a polygon is used for lighting calculations at the vertex during rendering of the polygon. The polygon table PT contains the polygons used in the model with each polygon represented as an ordered list of references (indices) to the vertex table VT (this list will be called the polygon's vertex list). Each polygon in PT also has another ordered list of references (indices) to the vertex-normal table VNT, with each vertex-normal index for each polygon's vertex (this list will be called the polygon's vertex-normal list). If the model consists of polygons of different surface colored materials, a material table MT is needed, in which each entry is a description of a colored material. In this case, each polygon in PT will make a reference to an entry in the material table MT. It is assumed that a polygon can only have one colored material throughout its whole surface area.

23 MAR 1998

FIG. 5a and **FIG. 5b** gives an example of a polygonal model and its vertex table VT, vertex-normal table VNT and polygon table PT. **FIG. 5a** is a model made of three polygons P_1 , P_2 and P_3 . The vertices in the model are V_1 , V_2 , ..., V_6 . The small arrows at the polygon vertices indicate the vertex-normals at the polygon vertices. The normals used in the model are N_1 , N_2 , ..., N_5 . **FIG. 5b** depicts the vertex table VT, vertex-normal table VNT and the polygon table PT which, together, describe the model in **FIG. 5a**.

(1) Grading

In this step, a weight is computed for each vertex in the original model, and it is stored in a weight table WT for use in subsequent processing steps. The weight of a vertex reflects the visual importance of the vertex. Two factors may be used to determine the visual importance of a vertex. A vertex V is visually important if (1) it has high probability of lying on the object's silhouettes from an arbitrary viewing direction, and (2) it bounds large faces that should not be affected by the removal of small details. The first factor (Factor 1) may be estimated using $\cos(\theta/2)$ where θ is the maximum angle between all pairs of edges incident to V . The second factor (Factor 2) may be estimated using either (a) the length of the longest edge among all the edges incident to V , or (b) the average length of all the edges incident to V . Finally, the weight of V is the product of the estimated value of Factor 1 and that of Factor 2.

The following is the reason why $\cos(\theta/2)$ is used as an estimation for Factor 1. With reference to **FIG. 6**, let V be the vertex to be assigned weight. Suppose θ is the maximum angle between all pairs of edges incident to V . Now, a sphere of radius r is centered at V , and also let V be the apex of a cone with tip angle θ and the base of the cone touches the sphere. A point on the sphere to V will represent a viewing direction from the point to V .

The apex of the cone represents a corner of the object, and the inside of the lower cone is the interior of the object. The only directions in which V cannot be seen

23 MAR 1998

are directions from the cap below the base of the cone. So, from any arbitrary viewing direction, the probability of seeing V on the silhouettes is

$$\begin{aligned} & \frac{\text{area of sphere} - \text{area of bottom cap} - \text{area of top cap}}{\text{area of sphere}} \\ &= \frac{4\pi r^2 - 2\pi r^2(1 - \cos \frac{\theta}{2}) - 2\pi r^2(1 - \cos \frac{\theta}{2})}{4\pi r^2} \\ &= \cos \frac{\theta}{2} \end{aligned}$$

In order to compute a weight for each vertex, it is necessary to construct (when not given as part of the input) the vertex-edge adjacency information from the vertex table VT and the polygon table PT of the original model. The following describes one possible approach. First, a vertex-edge adjacency list is associated to each vertex. The list associated to V is supposed to contain the indices of those vertices adjacent to V , where each adjacent vertex represents an edge incident to V . A single pass through the polygon table PT is sufficient to complete the task of building all the lists. For each vertex V , the computation of the estimate for Factor 1 involves traversing its vertex-edge adjacency list to identify the maximum angle θ between all pairs of edges incident to V . As for the estimation for Factor 2, just one pass through the list of adjacent vertices is enough to find L , which is either (a) the length of the longest edge among all the edges incident to V , or (b) the average length of all the edges incident to V . The weight W of V is the product of L and $\cos(\theta/2)$. W is then stored in the weight table WT for subsequent use. The i -th entry of the weight table WT contains the weight of the vertex stored in the i -th entry of the vertex table VT.

The vertex-edge adjacency lists constructed above may need to be stored for use in subsequent step in the preprocessing. For this, a vertex-edge adjacency table VEAT is created such that its i -th entry contains the vertex-edge adjacency list of the i -th vertex in the vertex table VT.

23 MAR 1998

Before introducing the next embodiment for estimating Factor 1, the following definitions are first made: (i) the edge vector of an edge incident to V is a vector in the direction from V to the other vertex of the edge, (ii) the average edge of a vertex V is defined as the edge incident to V and whose edge vector is the average of the unit edge vectors of all edges incident to V . In this second embodiment of the grading step, as a more efficient alternative, $\cos(\phi)$ can be used to estimate Factor 1 for a vertex V , where ϕ is either (a) the maximum angle among all the angles between the average edge of V and all the edges incident to V , or (b) the average of all the angles between the average edge of V and all the edges incident to V . If ϕ is greater than 90° , then let ϕ be 90° . More formally, ϕ can be defined as follows. Let $(V, V_1), (V, V_2), \dots, (V, V_n)$ be all the edges incident to V , and $V, V_1, V_2, \dots, V_n$ have positions $P, P_1, P_2, \dots, P_n$ respectively, and also let (V, V_a) be the average edge of V , and V_a has position P_a . Then

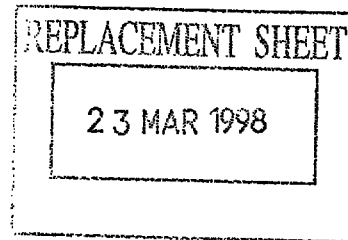
$$P_a - P = (1/n) \sum_{1 \leq i \leq n} ((P_i - P) / |P_i - P|)$$

and

- (a) $\phi = \min \{ 90^\circ, \max_{1 \leq i \leq n} \{ \text{angle between } (P_a - P) \text{ and } (P_i - P) \} \}$, or
- (b) $\phi = \min \{ 90^\circ, (1/n) \sum_{1 \leq i \leq n} (\text{angle between } (P_a - P) \text{ and } (P_i - P)) \}$

The implementation for computing $\cos(\phi)$ for each vertex is straightforward. The vertex-edge adjacency information, as described before, has to be constructed before the computation of the weight for each vertex. The weight of each vertex is then stored in the weight table WT for subsequent use.

In the third embodiment of the grading step for estimating Factor 1, the estimation for Factor 1 for a vertex V is done as follows. A small enough sphere is first centered at V . Assuming the 3D object is a solid, then the intersection of the solid and the sphere is a spherical polygon on the sphere. The area of this spherical polygon can be readily computed by applying some basic geometry. Finally, the estimation for Factor 1 is computed as



$(\text{area of sphere} - 2 * A) / \text{area of sphere}$

where

$A = \min \{ \text{area of spherical polygon}, \text{area of sphere} - \text{area of spherical polygon} \}.$

(2) Sorting vertices

To prepare the model for Step (4) (clustering step), the vertices in the vertex table VT need to be sorted in a non-increasing order on their weights in WT. An efficient way to accomplish this is to use a temporary storage to store the required ordering of the vertices without changing both VT and WT, and thus avoid changing the polygon table PT. The temporary storage is a table which will be called the vertex-order table VOT. The number of entries in VOT is the same as that in VT and WT. The i -th entry of VOT is supposed to contain the index of the vertex (in VT) which has the i -th largest weight. Initially, VOT is initialized such that the i -th entry of VOT contains the index of the i -th vertex in VT. An efficient in-memory sorting algorithm can be used to sort VOT. During sorting, to decide whether two VOT elements need to be exchanged, the sorting comparison is done on the weights of the vertices indexed by these two VOT elements.

(3) Triangulation

The triangulation step decomposes each polygon into one or more non-overlapping triangles. Triangulation can be accomplished using various known algorithms and need not introduce any new vertices into the original model. For lighting calculations, it is often assumed that a polygon's vertices are stored in an order that corresponds to a counter-clockwise rotation around a normal to the polygon's plane. Triangulation does not affect the normal, and does not alter the geometric object represented by the original model. After triangulation, the resulting triangles are stored in a triangle table TT. Each triangle in the triangle table TT is represented as an ordered list of three references (indices) to the vertex table VT, and an ordered list of three references (indices) to the vertex-normal table VNT.

FIG. 7a and **FIG. 7b** shows an example of triangulating a polygon. **FIG. 7b** illustrates a triangulation of the polygon in **FIG. 7a**. Each resulting triangle has three vertices associated therewith.

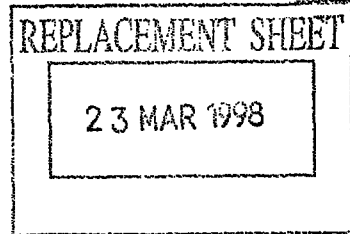
(4) Clustering

The clustering step uses one or more clustering cells to group vertices in VT into one or more clusters. In this invention, the clustering is done as follows:

- (i) Let V be the most highly weighted vertex in VT which has not been assigned to any cell,
- (ii) if V does not lie in any existing cell, then create a new cell centered at V and assign V to this new cell,
- (iii) else if V lies in exactly one existing cell, then assign V to the cell,
- (iv) else if V lies in the intersection of two or more existing cells, assign V to one of these cells according to one of the following cell selection criteria:
 - (a) assign V to the cell which has the most highly weighted center vertex,
 - (b) assign V to the cell whose center is nearest to V ,
 - (c) assign V to the cell C whose average length of the edges incident to V to the vertices already assigned to C is the shortest, and
 - (d) assign V to the cell which has the largest number of vertices adjacent to V (a vertex is adjacent to V if and only if there is an edge joining them),
- (v) go to step (i) if there is unassigned vertex in VT.

All the vertices assigned to a cell form a cluster. The result of the clustering step is a cluster table CT and a cluster-index table CIT. Each entry in CT corresponds to a cluster and it stores the center position of the clustering cell, and a list of indices of the vertices (in VT) which belongs to the cluster. As for the cluster-index table, its i -th entry contains the index of the cluster (in CT) which the i -th vertex in VT belongs to. Therefore, the number of entries in CIT is the same as that in VT.

The clustering cell used can be some simple three-dimensional shape, such as cube and sphere.

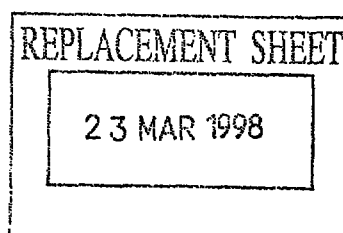


For each simplified model, the clustering step can be done in time linear to the number of vertices in the vertex table VT. To achieve such efficiency, the object space is first uniformly subdivided into cubical grids, each with width equal to the longest side of the smallest three-dimensional minimax box bounding the clustering cell used. For example, if the clustering cell used is an axis-aligned cube, then the width of each cubical grid is just the width of the clustering cube, whereas, if the clustering cell used is a sphere, then the width of each cubical grid is the sphere's diameter. For the purpose of describing this step of the method, it is assumed that axis-aligned cube is used as the clustering cell.

The main idea here is, whenever a new clustering cell is created, it is associated to the grid which contains the cell's center vertex. Though clustering cells can intersect one another, a cell's center vertex can never be inside another cell. This leads to a maximum of eight cells (or other upper bounds for other clustering shapes) that can be associated to each grid.

From the vertex-order table VOT, it is possible to visit each of the vertices in the vertex table VT exactly once in an ordered manner from the vertex with the highest weight to the vertex with the lowest weight. For each vertex V being visited, the grid it lies in can be known immediately. To find out whether V lies in any existing cells, it is necessary only to check the cells associated to the grid which contains V , and also to the other seven adjacent grids which are nearest to V . Only the cells associated to these eight grids can contain V .

If V does not lie in any of these cells, a new cell is created with its center at V , and associated to the grid which contains V . On the other hand, if V lies in exactly one of the cells, then assign V to the cell. Otherwise, if V lies in more than one cell, then assign V to one of these cells according to one of these cell selection criteria: (a) assign V to the cell which has the most highly weighted center vertex, (b) assign V to the cell whose center is nearest to V , (c) assign V to the cell C whose average length of the edges incident to V to the vertices already assigned to C is



the shortest, and (d) assign V to the cell which has the largest number of vertices adjacent to V .

For selecting a cell using cell selection criterion (c) or (d), the vertex-edge adjacency information can be obtained from the vertex-edge adjacency table VEAT created in the grading step.

Besides using a three-dimensional array to represent the array of grids, it is practical to use a hash table to store only those non-empty grids. The number of entries in the hash table is proportional to the number of vertices in VT. The 3D coordinates of a fixed corner of each grid is used as the key for hashing.

In another embodiment of the method to find the cells that a vertex V lies in, instead of using a cubical grid the same size as the clustering cube, a cubical grid one eighth of the volume is used. For the smaller grid, at most one cell can be associated to it. To find out all the existing cells a vertex V lies in, it is only necessary to check the cell associated to the grid which V lies in, and the cells associated to the 26 adjacent grids.

In this clustering step of the invention, a weight-controlled vertex clustering approach is used to position the clustering cells based on the positions of the original vertices and the weights of the vertices. This helps to preserve important features of the object, improves the approximation quality, and thus leads to simplified models with higher image fidelity to the original model. Using weight-controlled vertex clustering also results in more consistent placements of important features from one simplified model to the next. The more consistent placements of important features lead to smoother transitions from one level of simplification to the next when object is displayed using the simplified models.

The clustering step, in another embodiment, includes a clustering approach which is capable of greatly reducing the cases of thin triangles in the original model becoming much wider triangles in the simplified model. Such effect is very

23 MAR 1998

noticeable during display of the object and can degrade the approximation quality of the simplified model significantly. The clustering approach is described as follows:

- (i) Let V be the most highly weighted vertex in VT which has not been visited,
- (ii) if V does not lie in any existing cell, then create a new cell centered at V and assign V to this new cell,
- (iii) go to step (i) if there is unvisited vertex in VT ,
- (iv) for each vertex V in VT which has not been assigned to any cell, assign it to the cell whose center vertex is nearest to V .

Similarly, the cubical grids used above can be employed in this clustering method too.

(5) Thick-edge computation

When elongated parts of the object are reduced to just edges (done in Step (6) – synthesis), they may become less visible on the display screen when rendered as one-pixel-wide lines. If the actual thickness of an elongated part in the object space is known, its pixel-width on the screen can be deduced. Then, by rendering the edge using thick-line primitive with varying pixel-width, the actual visual effects of the elongated part can be approximately reproduced. Such edges, which have associated thicknesses, are called thick-edges. The use of thick-edges can improve the image quality significantly, especially for objects with many elongated parts.

Thickness of an edge can be estimated as follows. Let S be a set of triangles where each triangle has its vertices falling in clusters A and B only. These triangles will degenerate into edges in the synthesis step. Let V_A be the set of vertices that belong to cluster A and each is used by at least a triangle in S . Let P_A be the average position of all the vertices in V_A . And, V_B and P_B are similarly defined. Let L be the straight line passing through P_A and P_B . Let D_A be the average perpendicular distance of all points in V_A to the line L , and likewise for D_B .

23 MAR 1998

Then, the thickness of the edge, formed by collapsing triangles in S , is estimated by $(2D_A + 2D_B) / 2 = D_A + D_B$.

If the triangles in S are not of the same colored material, then these triangles have to be separated into groups where each group contains triangles of the same colored material. The thickness computations are then done separately and independently for each group.

To begin the computation of thickness for all thick-edges, it is necessary to find all the cluster-pairs which have triangles, which are going to degenerate into edges, using the vertices in them. For this purpose, one efficient way is to use a hash table whose number of entries is proportional to the number of triangles in the triangle table TT . For each triangle T in the triangle table TT , if the three vertices of T belong to exactly two distinct clusters, then T will degenerate into an edge after Step(6) (synthesis step). The information about which cluster a vertex belongs to can be obtained from the cluster-index table CIT . Let C_1 and C_2 be the indices of the two clusters, where $C_1 < C_2$. Then the pair (C_1, C_2) is used as the key for hashing. The hash table element which (C_1, C_2) is being hashed to, contains C_1, C_2 and a list of indices of triangles where each of them, like T , has vertices belonging to only the two clusters indexed by C_1 and C_2 . The index of T is then added to the list.

After all the triangles in TT have been visited, a pass through the hash table is needed to visit the triangle-index list associated to each non-empty hash table element. For each triangle-index list, an estimate of the thickness of all the triangles indexed in the list is computed as described above. This thickness is then associated to each triangle indexed in the list. A temporary storage, called the edge thickness table ETT , is used to store these thickness. The i -th entry of ETT contains the thickness associated to the i -th triangle in the triangle table TT .

(6) Synthesis

In this step, a representative vertex is computed to replace the vertices in each cluster which has more than one vertex. A cluster which has only one vertex will have that vertex as the representative vertex of the cluster. For each cluster C which has more than one vertex, the vertices which belong to it can be readily found out from the cluster table CT. Then the representative vertex of the cluster C can be computed using one of these vertex synthesis criteria: (a) select the center of the clustering cell as the position of the representative vertex, (b) compute a weighted-average position of all the vertices in the cluster using their weights stored in the weight table WT, so that the representative vertex is located at the weighted-average position, and (c) select the position of the vertex which is nearest to the weighted-average position of all the vertices in the cluster.

The weighted-average position of vertices $V_1, V_2, \dots, V_n$ with positions $P_1, P_2, \dots, P_n$ and weights $W_1, W_2, \dots, W_n$ is computed as follows

$$(W_1 \cdot P_1 + W_2 \cdot P_2 + \dots + W_n \cdot P_n) / (W_1 + W_2 + \dots + W_n)$$

The new representative vertices are stored in a new vertex table SVT, where the i -th entry of SVT contains the representative vertex of the i -th cluster in the cluster table CT. The triangle table TT is copied to a new triangle table NTT, and the vertex indices of each triangle in NTT is updated using the cluster index table CIT to make references to the new vertex table SVT.

When preparing many objects for display, one or more of the foregoing vertex synthesis criteria for computing a representative vertex for a cluster may be employed, depending on the geometry of the particular object being prepared. For each object, it is more preferable to use the same vertex synthesis criteria for generating all its simplified models. This will result in more consistent positions of the representative vertices across different levels of simplified models of the object, and this helps in improving the smoothness in transitions between different levels of simplified models when the object is displayed.

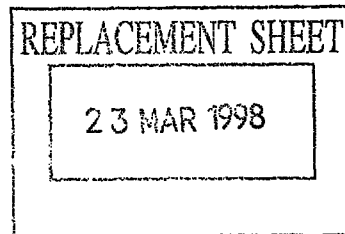


FIG. 8a, FIG. 8b, FIG. 8c and FIG. 8d illustrate two examples of clustering a two-dimensional object and synthesizing the representative vertices. **FIG. 8a** and **FIG. 8c** show the original model, which has already been graded and triangulated, but, for the sake of a simple illustration, the weights of the vertices and the triangles are not shown in the figures. The vertices in the model are intentionally labeled in such a way that vertex V_i has higher weight than V_{i+1} .

Referring to **FIG. 8a**, which shows clustering cells centered at some of the model's vertices. These cells are created in an order according to the order of the weights of their center vertices. In this example, when a vertex V lies in more than one cell, the cell selection criterion used during clustering is to assign V to the cell with the most highly weighted center vertex. As for the vertex synthesis criterion used during synthesis, the center of the clustering cell is selected as the position of the representative vertex. When vertices in the cells are replaced by their representative vertices, the result is the model in **FIG. 8b**.

The example illustrated in **FIG. 8c** and **FIG. 8d** is the same as that in **FIG. 8a** and **FIG. 8b**, except, now, the clustering cells used are larger. The model in **FIG. 8d** is the result of applying the clustering step and the synthesis step to the model in **FIG. 8c**.

(7) Reconstruction of normals

In this step, the vertex-normals of each new triangle have to be reconstructed in an attempt to preserve the smooth shadings on smooth faces and the discontinuities in shading at sharp edges and corners of the original model.

Before describing how normal reconstruction is actually done, the intuitive idea is presented. Here, a smooth surface segment is defined as a polygonal representation of a G^1 continuous surface. So, a sphere has only one smooth surface segment, a cube has six, a cylinder has three, and a cone has two. The normal reconstruction is done by collecting the vertex-normals at all the vertices in

each cluster. Since these vertices may come from different smooth surface segments, the vertex-normals collected for the cluster are further grouped according to which smooth surface segment they come from. For each group of vertex-normals, an average vertex-normal is computed to replace them. This average vertex-normal is the new vertex-normal used by some of the new triangles.

In details, the reconstruction requires, first, the construction of the vertex-normal information of the triangulated original model. The information from the clustering step is then used to update the vertex-normal information which is in turn used, together with the input vertex-normals in the vertex-normal table VNT, to compute new vertex-normals for each new triangle.

To construct the vertex-normal information, a vertex-normal set is initially associated to each vertex V in the vertex table VT of the original model. Each member of a vertex-normal set is a set of indices of the triangles in the triangle table TT. This set of indices of triangles will be called a triangle-index set. In each triangle-index set of V , the indexed triangles use the same vertex-normal (in the vertex-normal table VNT) at the triangle vertices that use V . So, intuitively, each triangle-index set of V represents a vertex-normal that is shared by the indexed triangles at V , and the number of triangle-index sets in the vertex-normal set associated to V corresponds to the number of different vertex-normals at V . For example, if V is in the middle of a smooth surface segment, then there is only one triangle-index set in the vertex-normal set of V , however if V is at a corner of a cube, then there are 3 triangle-index sets in the vertex-normal set of V .

The vertex-normal information constructed is then updated using the information from the clustering step. Assuming that vertices $V_1, V_2, \dots, V_k$ in the vertex table VT are in the same cluster, and their respective vertex-normal sets are $S_1, S_2, \dots, S_k$, the vertex-normal information can be updated as follows:

- (i) let $S \leftarrow S_1 \cup S_2 \cup \dots \cup S_k$,

(ii) if $\exists X, Y \in S$ such that $X \neq Y$ and $X \cap Y \neq \emptyset$ then

$$S \leftarrow (S - \{X\} - \{Y\}) \cup \{X \cup Y\},$$

(iii) go to step (ii) if S is changed, else stop.

The set S is the new vertex-normal set which must be associated to the representative vertex of the cluster. If any triangle-index set becomes empty, the empty set is removed from the corresponding new vertex-normal set. The above vertex-normal information updating step has to be done once for each and every cluster, and each new vertex-normal set is associated to its respective representative vertex of the cluster.

Now, for each representative vertex V created in the synthesis step, there is a new (updated) vertex-normal set associated to it. The next step is to compute the vertex-normals for each new triangle (in the triangle table NTT) using the new vertex-normal sets. For each triangle-index set in the vertex-normal set of V , a new vertex-normal is computed, and for each new triangle T (every new triangle has the same index as its original triangle) indexed in the triangle-index set, those triangle vertices which reference V will use the newly computed vertex-normal. The new vertex-normal is computed as follows. For each new triangle T indexed in the triangle-index set, and for each triangle vertex of T which references V , the old unit vertex-normal (from VNT) referenced by the triangle vertex is obtained. These old unit vertex-normals from all the triangles indexed in the triangle-index set are summed together to obtain an average vector. This average vector is the new vertex-normal and it is added to the new vertex-normal table SVNT, and the appropriate vertex-normal index in each indexed triangle's vertex-normal list is updated to reference this new average vector. The updated triangle table NTT is called NTT' in **FIG. 9**.

This method of normal reconstruction is capable of handling non-manifold surfaces. It is able to very well preserve the smooth shadings on smooth faces and the discontinuities in shading at sharp edges and corners of the original model.

In practice, a vertex-normal set may be represented by a list (called a vertex-normal set list), in which each node contains a list (called a triangle-index set list) of indices of triangles in the triangle table TT.

The updating of the vertex-normal set lists for a cluster of vertices $V_1, V_2, \dots, V_k$ is done as follows. First, a new vertex-normal set list Q is created by concatenating all the vertex-normal set lists associated to $V_1, V_2, \dots, V_k$. Then, for each triangle index J in each triangle-index set list L , let T be the triangle indexed by J , and associate to T a pointer to the node of Q containing L . So, a triangle can only be associated with one to three such pointers. For convenience, each of such pointers is called a Q -node pointer.

In the next step of the updating, a head-to-tail traversal of each triangle-index set list in Q is done. For each triangle index J in a triangle-index set list L , let T be the triangle indexed by J . If T is marked 'visited', then delete J from L . Otherwise, when T is not marked 'visited', if T has more than one Q -node pointer, remove the other triangle-index set lists from the Q 's nodes pointed by the Q -node pointers, append these lists to L and mark T 'visited'.

Once all the triangle-index set lists have been traversed and updated, those Q 's nodes which contain empty triangle-index set lists are deleted from Q , and the resulting vertex-normal set list Q is associated to the representative vertex of the cluster.

The new vertex-normal set lists associated to the representative vertices are used to compute the new vertex-normals for each new triangle. These new vertex-normals are stored in a new vertex-normal table SVNT, and the appropriate vertex-normal index in each indexed triangle's vertex-normal list is updated to reference this new vertex-normal table.

(8) Elimination

As a result of merging of vertices in a cluster into one vertex, some triangles degenerate into edges and points. A triangle with three identical vertices will degenerate into a point, while a triangle with exactly two identical vertices will degenerate into an edge. These degenerated triangles are identified from NTT' and for each such triangle, its duplicated vertex indices and the corresponding vertex-normal indices are removed.

The next thing to do is to remove those duplicated triangles, edges and points in the updated NTT'. More specifically, these are removed: (a) duplicated triangles, (b) duplicated edges, (c) duplicated points, (d) edges that are the same as some triangles' edges, and (e) points that fall on the end-points of some edges or on the vertices of some triangles.

If the primitives (triangles, edges and points) are of different colored materials, then they have to be handled separately and independently. To determine if a primitive is duplicated, it is only compared with primitives of the same colored material.

A hash table may be used to efficiently identify duplications of primitives (triangles, edges and points). The number of entries in the hash table is proportional to the number of primitives stored in NTT'. For each primitive P in NTT', if P is a triangle, then its three vertex indices V_1 , V_2 and V_3 , where $V_1 < V_2 < V_3$, are used as a key for hashing. If the triangle (V_1, V_2, V_3) is found in the hash table, then P is deleted from NTT'. On the other hand, if the triangle (V_1, V_2, V_3) is not found in the hash table, then (V_1, V_2, V_3) is added to the hash table and P is retained, and each of (V_1, V_2) , (V_2, V_3) , (V_1, V_3) , (V_1) , (V_2) and (V_3) is also hashed and added to the hash table if it is not already inside. If P is an edge, then its two vertex indices V_1 and V_2 , where $V_1 < V_2$, are used as a key for hashing. If the edge (V_1, V_2) is found in the hash table, then P is deleted from NTT'. On the other hand, if the edge (V_1, V_2) is not found in the hash table, then (V_1, V_2) is added to the hash table and P is retained, and each of (V_1) and (V_2) is also hashed and added to the hash table if it

is not already inside. If P is a point, then its vertex index V_1 is used as a key for hashing. If the point (V_1) is found in the hash table, then P is deleted from NTT'. On the other hand, if the point (V_1) is not found in the hash table, then (V_1) is added to the hash table and P is retained.

The use of the above algorithm requires NTT' to be pre-sorted in such a way that all the triangles come before all the edges, and all the edges come before all the points, so that, during the identification of duplicated primitives, all triangles are visited before all edges, and all edges are visited before all points.

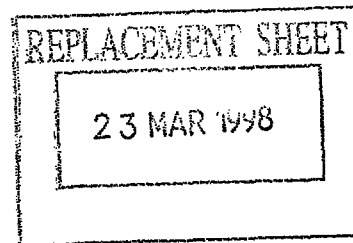
The retained triangles in NTT' are transferred to a triangle table STT, the retained edges, together with their edge thickness from the edge thickness table ETT, are transferred to a edge table SET, and the retained points are transferred to a point table SPT. The primitives in STT, SET and SPT all make references to the vertex table SVT created in Step (6) (synthesis step).

(9) Storing simplified model

The simplified model is described by the vertex table SVT, the vertex-normal table SVNT, the triangle table STT, the edge table SET and the point table SPT. These tables, together with the size of the clustering cell used, are stored in the memory or on other storage media for subsequent use.

No reconstruction of normals

In another embodiment of the preprocessing step, a simplified model is generated without the reconstruction of vertex-normals (Step (7)). In this case, the simplified model does not have the new vertex-normal table SVNT, and for each triangle in the triangle table STT, the triangle's vertex-normal list makes references to the normals stored in the vertex-normal table VNT of the original model. The vertex-normal table VNT is thus shared between the original model and the simplified model.



Next simplified model

For the next level of simplified model of the same object, only Step (4) to Step (9) need to be executed with a new clustering cell size for Step (4). The vertex table VT, the triangle table TT, the weight table WT, the vertex-edge adjacency table VEAT, and the vertex-order table VOT can be reused.

Having generated a plurality of simplified models for the object, the object's original model and the simplified models are stored within a database of graphic models. FIG. 2 illustrates a memory that is coupled to the CPU of FIG. 1. The memory stores a plurality of object models (MODEL 1 to MODEL n) and, in accordance with the invention, also stores one or more simplified models ($S_{1,1}$, $S_{1,2}$, etc.) for each of the object models. The cell sizes ($C_{1,1}$, $C_{1,2}$, etc.) used for the generation of the simplified models are also stored together with the simplified models.

It will be appreciated that instructions for carrying out the calculations described can be supplied to the computer shown in FIG. 1 from a suitable memory and that these instructions can also be carried on any suitable storage medium such as a hard disk, a ROM or a floppy disk. It will also be appreciated that the instructions for the CPU shown in FIG. 1 can be supplied over a data link such as the Internet and this is also shown in FIG. 1.

Choosing clustering cell sizes

To choose the cell sizes for generating the sequence of simplified models of an object, a simple method can be adopted. A minimax box bounding the object is computed first. Then, the clustering cell widths can be some fractions of the length of one of the side of the object's bounding box. For example, if the length of a side of the bounding box is L , the cell widths can be $k_1 \cdot L$, $k_2 \cdot L$, $k_3 \cdot L$, ... and $k_m \cdot L$, where each k_i is a constant of value between 0 and 1 and $k_i < k_{i+1}$, and m is the number of simplified models to be generated. These constant fractions can be adjusted by the user to suit the complexity of the original model, the number of objects in the scene, and the rendering power of the graphics display apparatus. For instance,

very often, for a very complex original model with many small features, a relatively small cell size is enough to significantly reduce its complexity. If there are many objects in the scene or the rendering power of the display apparatus is low, it is more preferable to use some cell sizes which are relatively much bigger.

FIG. 9 shows the dataflow between the steps of the preprocessing for the generation of a single level of simplified model.

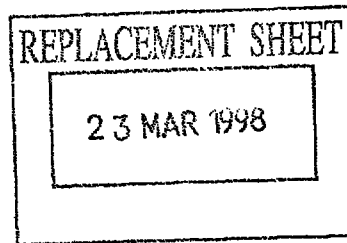
Model Selection Step

The method of the invention further includes a step of selecting either the original model or one of the simplified models for displaying an object. For example, when an object is very far away from the viewpoint, it may be sufficient to choose a very simplified model to display the object. This will save some time in displaying the object compared to always displaying the original model. Whereas, if the object is near to the viewpoint, a more detailed model (or a less simplified model) may be required to reveal the details on the object.

The step of selecting a model includes a step of determining the size of the object space covered by a pixel on the display screen, and this will be called the object-space-to-pixel-space-ratio or R for short. The value of R of an object with a distance D from the viewpoint is

$$R = (2D \tan (FOV/2)) / H$$

where FOV is the vertical field of view angle of the camera, and H is the height of the viewing window in pixels. The distance D of an object from the viewpoint can be readily estimated using the shortest distance between the viewpoint and a pre-computed simple bounding volume enclosing the object. An enclosing sphere of the object is a convenient bounding volume for computing D . If the object or the viewpoint moves, such that D changes, or FOV or H is changed, then R has to be recomputed.



For displaying the object, the model with the largest clustering cell width which is less than or equal to R will be chosen. Here, it is said that the original model has clustering cell of width 0. Let M be the model chosen. There is no benefit in choosing a model more detailed than M for display, since at this distance from the viewpoint, each cell width in the object space will be mapped to approximately one or less pixel width. This model selection is done independently for each object, and each has its own value of R .

However, to maintain real-time interactive frame rates, a model less detailed than M may need to be selected for display. This model is the most detailed model of the object whose rendering time, including the calculation of perspective transformation, lighting computation, scan conversion, etc., is consistent with providing smooth motion of the object on the display.

An adaptive model selection technique can be integrated to select a model for each object in the scene for display such that the total display time of all the objects is consistent with providing a good frame rate. An example of such adaptive model selection technique is described by Funkhouser and Séquin in "Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments", Computer Graphics (SIGGRAPH '93 Proceedings), pp. 247-254, 1993.

Usually, when there are many complex objects in the scene, the rendering workload will generally be high almost all the time. To speed up the adaptive model selection step, an attempt is made to reduce the number of models of each object that need to be considered by the optimization algorithm in the model selection step. This is done by choosing a model which is not more detailed than M' where M' is the model with the largest cell width less than or equal to $R \cdot X$, and $X > 1$. X can be adjusted, automatically by the adaptive model selection algorithm or manually by the user, according to the general rendering workload. Note that

the use of $X > 1$ does not stop the most detailed model of an object from being selected.

Referring again to FIG. 2, the system includes user input devices such as a keyboard, mouse or trackball. During use, an entire scene may be moved according to a user's command. The user inputs these commands using the input devices. In response, the CPU may access a simplified model to be used for each object within the scene. The system automatically switches to the simplified models when the user initiates the motion of the objects in the scene. When motion stops, the system returns to the full resolution mode, triggered by an explicit user action, or by itself if there is no user intervention for a predefined interval of time. The goal is to select from the memory the most detailed model for each object, while also permitting real-time interactive motion to be achieved. As the model selection step requires the clustering cell size used for generating each simplified model, the clustering cell size is kept together with the simplified model in the memory.

Rendering a model

When a model is to be displayed, the Host processor will send the necessary commands and data from the model to the graphics subsystem.

To display a triangle, its vertices and its vertex-normals are sent to the graphics subsystem. For rendering a thick-edge using a thick-line, besides the vertices at the two endpoints, the pixel-width of the thick-line primitive must also be sent to the graphics subsystem. The pixel-width of the thick-line is computed as T / R , where T is the estimated thickness of the thick-edge, and R is the object-space-to-pixel-space-ratio of the object as described above.

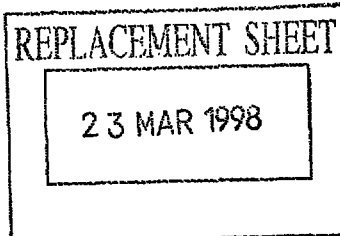
The shading or intensity of a thick-line which renders a thick-edge can, in general, be improved by using a dynamic normal instead of a fixed one for the thick-line. This dynamic normal is only computed during rendering. The improvement is particularly good if that the elongated part approximated by the edge is cylinder-

like. The dynamic normal is the vector perpendicular to the edge, in the plane containing the edge and the camera, and the angle it makes with the vector from any point on the edge to the camera is acute. The dynamic normal is calculated using the formula $(V_1 - C) \times (V_2 - C) \times (V_2 - V_1)$ where V_1 , V_2 are the edge's endpoints and C is the camera's position.

Operation Of The Method Of Invention

FIG. 4 is a flowchart that depicts the operation of the method of the invention. First, the system loads or generates a database of objects appearing in a scene to be displayed. These objects may be elements of, for example, a CAD database or a virtual environment. For each of these object, one or more simplified models are generated by the preprocessing step depicted in **FIG. 3**. The original models and the simplified models of all objects are stored in the memory as shown in **FIG. 2**. At this point, the system is ready to begin an interactive display session with a user.

In response to a command from the user to change the position of the displayed object, the system accesses the memory to obtain a suitable model. As was previously described, it is within the scope of the invention for the system to select an appropriate simplified model for display. The selected model is passed to the graphics subsystem, resulting in the display of the selected model on the display screen.



CLAIMS

Thus, while the invention has been particularly shown and described with respect to preferred embodiments thereof, it will be understood by those skilled in the art that changes in form and details may be made therein without departing from the scope and spirit of the invention.

Having thus described our invention, what we claim as new, and desire to secure by Letters Patent are:

1. A method for execution by a data processor to prepare an object for display on a display means wherein the object is represented by an original model which is described by a plurality of vertices and planar faces each having boundaries defined in terms of the plurality of vertices, the method comprising the steps of:

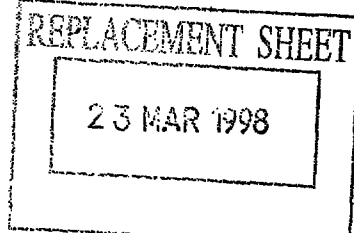
preprocessing the original model to produce at least a first simplified model with the first simplified model having fewer faces and vertices than the original model, and wherein the step of preprocessing to produce the first simplified model comprises the steps of:

grading each vertex in the original model by computing a weight for the vertex to reflect its visual importance;

clustering vertices in the original model by using a first plurality of clustering cells to group vertices into a plurality of clusters wherein the order of creations and the positions of the clustering cells are determined by at least the weights and the positions of the vertices in the original model; and

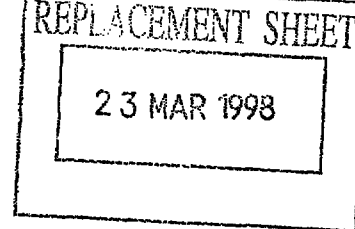
utilizing the clustering cells so produced to generate a simplified model; model selecting to select either the original model or a simplified model for display as a function of at least the size of the object space covered by a pixel and the size of the clustering cells used in generating the simplified model.





2. A method as set forth in claim 1 wherein the step of preprocessing to produce the first simplified model includes the step of computing the thickness of each group of triangles which are to be degenerated into the same edge.
3. A method as set forth in claim 1 wherein the step of preprocessing to produce the first simplified model includes the step of reconstructing vertex-normals for each new triangle, wherein the step of reconstructing vertex-normals comprises the steps of:
 - constructing the vertex-normal information of each vertex in the original model;
 - updating the vertex-normal information for each cluster of vertices to derive the vertex-normal information of the representative vertex; and
 - computing new vertex-normals for each new triangle using the vertex-normal information of representative vertices and the input vertex-normals in the vertex-normal table VNT.
4. A method as set forth in claim 1 wherein the step of preprocessing to produce the first simplified model includes the steps of:
 - sorting vertices in the original model by ordering them according to the order of their weights;
 - triangulating each face in the original model by partitioning it into one or more non-overlapping triangles;
 - synthesizing a representative vertex for each cluster created in the step of clustering as set forth in claim 1 to replace all the vertices in the cluster, and thus simplifying some triangles into edges and points;
 - eliminating duplicated triangles, edges and points in the simplified model; and
 - storing the first simplified model for subsequent use.
5. A method as set forth in claim 1 wherein the step of grading includes a step of using the value of L as the estimation, for each vertex V , for the size of the faces bounded by V , where L is either (a) the length of the longest edge among all the edges incident to V , or (b) the average length of all the edges incident to V .





6. A method as set forth in claim 1 wherein the step of grading includes a step of using the value of $\cos(\theta/2)$ as the estimation, for each vertex, for the probability of seeing the vertex on the silhouettes of the object from any arbitrary viewing direction, where θ is the maximum angle between all pairs of edges incident to the vertex.

7. A method as set forth in claim 1 wherein the step of grading includes a step of using the value of $\cos(\phi)$ as the estimation, for each vertex V , for the probability of seeing V on the silhouettes of the object from any arbitrary viewing direction, where ϕ is either (a) the maximum angle among all the angles between the average edge of V and all the edges incident from V , or (b) the average of all the angles between the average edge of V and all the edges incident from V .

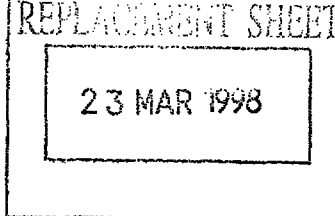
8. A method as set forth in claim 1 wherein the step of grading includes a step of using the value of M as the estimation, for each vertex V , for the probability of seeing V on the silhouettes of the object from any arbitrary viewing direction, where M is equal to $((\text{area of sphere} - 2 * A) / \text{area of sphere})$ and $A = \min \{ \text{area of spherical polygon}, \text{area of sphere} - \text{area of spherical polygon} \}$.

9. A method as set forth in claim 1 wherein the step of grading includes a step of using (a) the product of L and $\cos(\theta/2)$, (b) the product of L and $\cos(\phi)$, or (c) the product of L and M , as the weight of each vertex, where L has the value as set forth in claim 5, $\cos(\theta/2)$ has the value as set forth in claim 6, $\cos(\phi)$ has the value as set forth in claim 7, and M has the value as set forth in claim 8.

10. A method as set forth in claim 1 wherein the step of clustering comprises:

- letting the candidate vertex V be the most highly weighted vertex in the original model which has not been assigned to any cell,
- determining if V does not lie in any existing cell, then creating a new cell centered at V and assigning V to this new cell,
- else if V lies in exactly one existing cell, then assigning V to the cell,

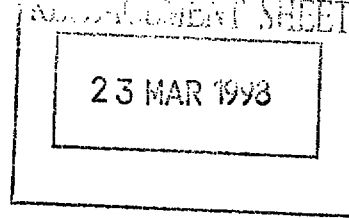




else if V lies in two or more existing cells, assigning V to one of these cells according to a cell selection criterion, and
repeating the above steps if necessary, until all the vertices have been assigned to some cells where the vertices assigned to each cell form a cluster.

11. A method as set forth in claim 10 wherein the cell selection criterion includes a step of selecting the cell with the most highly weighted center vertex.
12. A method as set forth in claim 10 wherein the cell selection criterion includes a step of selecting the cell whose center vertex is nearest to the candidate vertex.
13. A method as set forth in claim 10 wherein the cell selection criterion includes a step of selecting the cell C whose average length of the edges incident to the candidate vertex to the vertices already assigned to C is the shortest.
14. A method as set forth in claim 10 wherein the cell selection criterion includes a step of selecting the cell which has the largest number of vertices adjacent to the candidate vertex.
15. A method as set forth in claim 10 wherein the step of identifying the cells the candidate vertex lies in includes the step of using a plurality of three-dimensional grids to partition the object space, so as to identify the cells the candidate vertex lies in.
16. A method as set forth in claim 1 wherein the step of clustering comprises:
letting the candidate vertex V be the most highly weighted vertex in the original model which has not been visited,
determining if V does not lie in any existing cell, then creating a new cell centered at V and assigning V to this new cell,
repeating the above steps when there is still unvisited vertex in the original model, thereafter





for each vertex V in the original model which has not been assigned to any cell, assigning it to the cell whose center vertex is nearest to V , so that all the vertices assigned to a cell form a cluster.

17. A method as set forth in claim 16 wherein the step of identifying the cells the candidate vertex lies in includes the step of using a plurality of three-dimensional grids to partition the object space, so as to identify the cells the candidate vertex lies in.

18. A method as set forth in claim 2 wherein the step of computing the thickness includes a step of finding each cluster-pair which contains vertices which are used by a group of triangles that are to be degenerated into the same edge.

19. A method as set forth in claim 18 wherein those cluster-pairs containing some triangles are stored in a hash table.

20. A method as set forth in claim 2 wherein the step of computing the thickness includes a step of estimating the thickness of each group of triangles in a cluster-pair as a function of the perpendicular distances of all the vertices of these triangles to the line joining the two average positions of these vertices in the two clusters of the cluster-pair.

21. A method as set forth in claim 3 wherein the step of constructing the vertex-normal information includes the step of building a vertex-normal set for each vertex V in the vertex table VT of the original model.

22. A method as set forth in claim 3 wherein the step of updating the vertex-normal information for a cluster of vertices $V_1, V_2, \dots, V_k$ (in the vertex table VT) comprises:

letting $S \leftarrow S_1 \cup S_2 \cup \dots \cup S_k$,

determining if $\exists X, Y \in S$ such that $X \neq Y$ and $X \cap Y \neq \emptyset$ then

$S \leftarrow (S - \{X\} - \{Y\}) \cup \{X \cup Y\}$,



repeating the above steps if S is changed, else stop;
where $S_1, S_2, \dots, S_k$ are the respective vertex-normal sets of the vertices $V_1, V_2, \dots, V_k$ and the final S is the vertex-normal set of the representative vertex of the cluster.

23. A method as set forth in claim 4 wherein the step of synthesizing includes a step of selecting the center of the clustering cell as the position of the representative vertex.

24. A method as set forth in claim 4 wherein the step of synthesizing includes a step of computing the weighted-average position of the vertices within a cluster, and a step of selecting the weighted-average position as a position of the representative vertex.

25. A method as set forth in claim 4 wherein the step of synthesizing includes a step of computing the weighted-average position of the vertices within a cluster, and a step of selecting as the representative vertex the vertex in the cluster that is nearest to the weighted-average position.

26. A method as set forth in claim 4 wherein the step of eliminating includes a step of identifying duplications of triangles, edges and points with hash table.

27. A method as set forth in claim 1 wherein the step of preprocessing further including the steps of:

clustering vertices in the original model by using a second plurality of clustering cells, wherein said second plurality of clustering cells has a dimension different from a corresponding dimension of said first plurality of clustering cells, to group vertices into a plurality of clusters wherein the order of creations and the positions of the clustering cells are determined by at least the weights and the positions of the vertices in the original model; and



23 MAR 1998

utilizing the clustering cells so produced to generate a second simplified model.

28. A method as set forth in claim 27 wherein the step of preprocessing to produce the second simplified model further includes the steps of:

- computing the thickness of each group of triangles which are to be degenerated into the same edge;
- synthesizing a representative vertex for each cluster created in the step of clustering to replace all the vertices in the cluster, and thus simplifying some triangles into edges and points;
- reconstructing vertex-normals for each new triangles;
- eliminating duplicated triangles, edges and points in the simplified model; and
- storing the second simplified model for subsequent use.

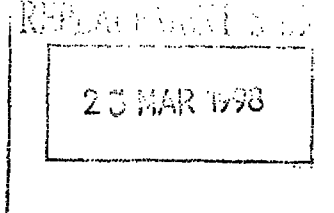
29. A method as set forth in claim 27 wherein the steps of:

- clustering vertices in the original model;
- computing the thickness of each group of triangles which are to be degenerated into the same edge;
- synthesizing a representative vertex for each cluster created in the step of clustering to replace all the vertices in the cluster;
- reconstructing vertex-normals for each new triangles;
- eliminating duplicated triangles, edges and points in the simplified model, and
- storing the second simplified model for subsequent use;

may be repeated as many times as needed provided each time the clustering step is done using a plurality of clustering cells with dimension different from the preceding ones.

30. A method as set forth in claim 1 wherein the step of model selecting includes a step of determining the object space covered by a pixel on the display means.





31. A method as set forth in claim 1 further including the step of computing the pixel-width of each thick-line primitive which renders a thick-edge, during display of the object using a simplified model.

32. A method as set forth in claim 1 further including the step of computing the dynamic normal for each thick-line primitive which renders a thick-edge, during display of the object using a simplified model.

33. Apparatus for preparing an object for display on a display means wherein the object is represented by an original model which is described by a plurality of vertices and planar faces each having boundaries defined in terms of the plurality of vertices, the apparatus comprising:

means for processing the data representing the original model to produce at least a first simplified model with the first simplified model having fewer faces and vertices than the original model, and wherein the apparatus for processing to produce the first simplified model comprises:

means for grading each vertex in the original model by computing a weight for the vertex to reflect its visual importance;

means for clustering vertices in the original model by using a first plurality of clustering cells to group vertices into a plurality of clusters wherein the order of creations and the positions of the clustering cells are determined by at least the weights and the positions of the vertices in the original model; and

means for utilizing the clustering cells so produced to generate a simplified model;

means for model selecting to select either the original model or a simplified model for display as a function of at least the size of the object space covered by a pixel and the size of the clustering cells used in generating the simplified model.

34. The apparatus as set forth in claim 33 further includes means for storing clustering cell sizes used in generating the said models, and wherein said means



for model selecting is responsive to a determined size of the object space covered by a pixel on the display means, and to said stored clustering cell sizes, to select one of the said models for display.

35. A storage device storing computer readable instructions for causing a programmable processing apparatus to perform a method of preparing an object for display on a display means wherein the object is represented by an original model which is described by a plurality of vertices and planar faces each having boundaries defined in terms of the plurality of vertices, the method comprising the steps of:

preprocessing the original model to produce at least a first simplified model with the first simplified model having fewer faces and vertices than the original model, and wherein the step of preprocessing to produce the first simplified model comprises the steps of:

grading each vertex in the original model by computing a weight for the vertex to reflect its visual importance;

clustering vertices in the original model by using a first plurality of clustering cells to group vertices into a plurality of clusters wherein the order of creations and the positions of the clustering cells are determined by at least the weights and the positions of the vertices in the original model; and

utilizing the clustering cells so produced to generate a simplified model;

model selecting to select either the original model or a simplified model for display as a function of at least the size of the object space covered by a pixel and the size of the clustering cells used in generating the simplified model.

36. A signal conveying instructions for causing a programmable processing apparatus to perform a method of preparing an object for display on a display means wherein the object is represented by an original model which is described by a plurality of vertices and planar faces each having boundaries defined in terms of the plurality of vertices, the method comprising the steps of:



23 MAR 1998

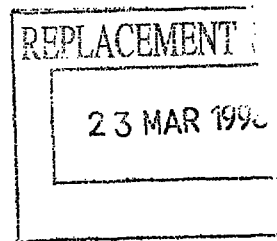
preprocessing the original model to produce at least a first simplified model with the first simplified model having fewer faces and vertices than the original model, and wherein the step of preprocessing to produce the first simplified model comprises the steps of:

grading each vertex in the original model by computing a weight for the vertex to reflect its visual importance;

clustering vertices in the original model by using a first plurality of clustering cells to group vertices into a plurality of clusters wherein the order of creations and the positions of the clustering cells are determined by at least the weights and the positions of the vertices in the original model; and

utilizing the clustering cells so produced to generate a simplified model; model selecting to select either the original model or a simplified model for display as a function of at least the size of the object space covered by a pixel and the size of the clustering cells used in generating the simplified model.



**ABSTRACT****MULTI-RESOLUTION GRAPHIC REPRESENTATION GENERATED BY
WEIGHT-CONTROLLED VERTEX CLUSTERING FOR INTERACTIVE
VISUALIZATION APPLICATIONS**

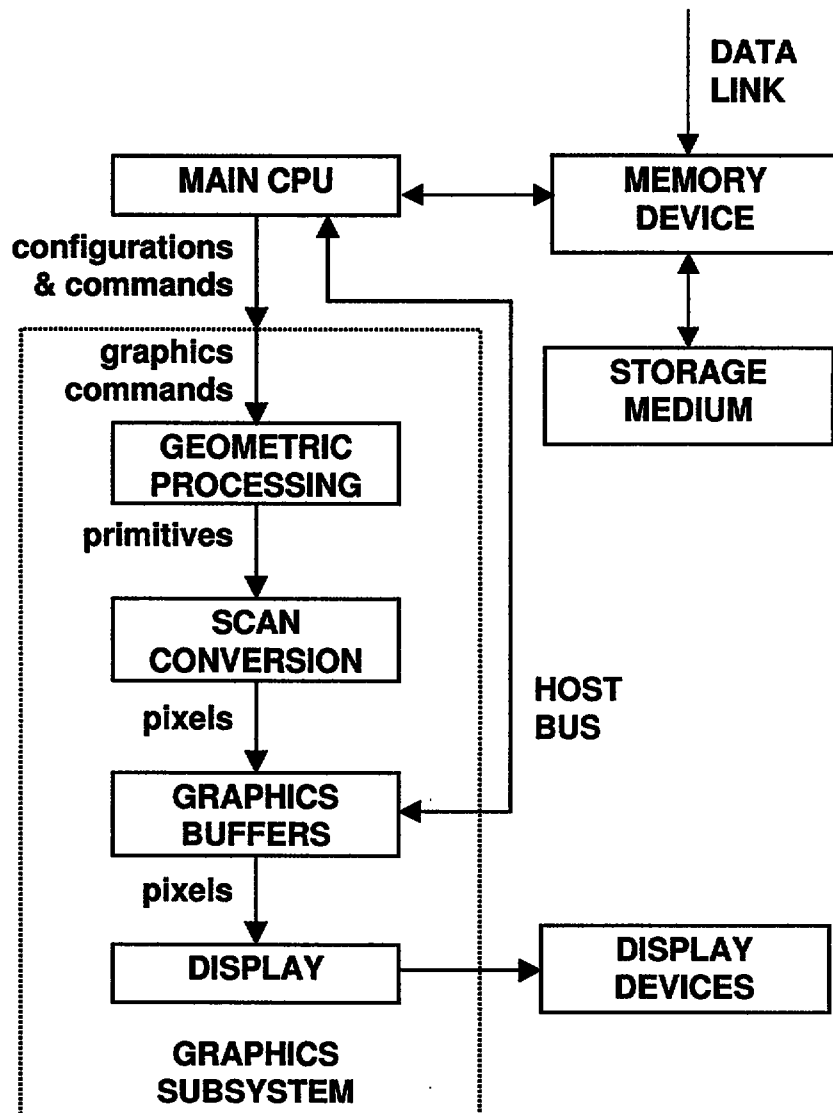
A method for execution by a data processor that prepares an object for display during interactive visualization. The method includes a step of preprocessing an original model, which is a polygonal boundary representation of the object, so as to produce a plurality of simplified models with each simplified model having fewer faces and vertices than the original model. The step of preprocessing includes the steps of grading, sorting vertices, triangulation, clustering, thick-edge computation, synthesis, reconstruction of normals, elimination, and storing the simplified model. During display of the object, the method further includes a step of selecting either the original model or one of the simplified models for display so as to achieve real-time interactive motion of the object.

Fig 8a & Fig.8b are to be published.

23 MAR 1998

1

FIG. 1



23 MAR 1998

10

FIG. 9

