A system includes plural processors and an interactive display library for interactively displaying a model having a complex surface. The library includes a surface pre-processor for pre-processing the model to provide a plurality of surface primitives; an application program for displaying the model with respect to a first eye-point of a plurality of eye-points and for selecting a second eye-point of the eye-points for display of the model with respect to the second eye-point; and a visibility thread for performing a visibility computation (92). The library further includes a tessellation thread (96) which tessellates each selected surface primitive to provide a plurality of triangles or triangle-strips representative of the selected surface primitive with respect to the second eye-point; and a triangle rendering thread which employs a graphics board to render the triangles or triangle-strips and, thus, display the model with respect to the second eye-point.
<table>
<thead>
<tr>
<th>Code</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL</td>
<td>Albania</td>
</tr>
<tr>
<td>AM</td>
<td>Armenia</td>
</tr>
<tr>
<td>AT</td>
<td>Austria</td>
</tr>
<tr>
<td>AU</td>
<td>Australia</td>
</tr>
<tr>
<td>AZ</td>
<td>Azerbaijan</td>
</tr>
<tr>
<td>BA</td>
<td>Bosnia and Herzegovina</td>
</tr>
<tr>
<td>BB</td>
<td>Barbados</td>
</tr>
<tr>
<td>BE</td>
<td>Belgium</td>
</tr>
<tr>
<td>BF</td>
<td>Burkina Faso</td>
</tr>
<tr>
<td>BG</td>
<td>Bulgaria</td>
</tr>
<tr>
<td>BJ</td>
<td>Benin</td>
</tr>
<tr>
<td>BR</td>
<td>Brazil</td>
</tr>
<tr>
<td>BY</td>
<td>Belarus</td>
</tr>
<tr>
<td>CA</td>
<td>Canada</td>
</tr>
<tr>
<td>CF</td>
<td>Central African Republic</td>
</tr>
<tr>
<td>CG</td>
<td>Congo</td>
</tr>
<tr>
<td>CH</td>
<td>Switzerland</td>
</tr>
<tr>
<td>CI</td>
<td>Côte d’Ivoire</td>
</tr>
<tr>
<td>CM</td>
<td>Cameroon</td>
</tr>
<tr>
<td>CN</td>
<td>China</td>
</tr>
<tr>
<td>CU</td>
<td>Cuba</td>
</tr>
<tr>
<td>CZ</td>
<td>Czech Republic</td>
</tr>
<tr>
<td>DE</td>
<td>Germany</td>
</tr>
<tr>
<td>DK</td>
<td>Denmark</td>
</tr>
<tr>
<td>EE</td>
<td>Estonia</td>
</tr>
<tr>
<td>ES</td>
<td>Spain</td>
</tr>
<tr>
<td>FI</td>
<td>Finland</td>
</tr>
<tr>
<td>FR</td>
<td>France</td>
</tr>
<tr>
<td>GA</td>
<td>Gabon</td>
</tr>
<tr>
<td>GB</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>GE</td>
<td>Georgia</td>
</tr>
<tr>
<td>GH</td>
<td>Ghana</td>
</tr>
<tr>
<td>GN</td>
<td>Guinea</td>
</tr>
<tr>
<td>GR</td>
<td>Greece</td>
</tr>
<tr>
<td>HU</td>
<td>Hungary</td>
</tr>
<tr>
<td>IE</td>
<td>Ireland</td>
</tr>
<tr>
<td>IL</td>
<td>Israel</td>
</tr>
<tr>
<td>IS</td>
<td>Iceland</td>
</tr>
<tr>
<td>JP</td>
<td>Japan</td>
</tr>
<tr>
<td>KE</td>
<td>Kenya</td>
</tr>
<tr>
<td>KG</td>
<td>Kyrgyzstan</td>
</tr>
<tr>
<td>KP</td>
<td>Democratic People’s Republic of Korea</td>
</tr>
<tr>
<td>KR</td>
<td>Republic of Korea</td>
</tr>
<tr>
<td>KZ</td>
<td>Kazakhstan</td>
</tr>
<tr>
<td>LC</td>
<td>Saint Lucia</td>
</tr>
<tr>
<td>LI</td>
<td>Liechtenstein</td>
</tr>
<tr>
<td>LK</td>
<td>Sri Lanka</td>
</tr>
<tr>
<td>LR</td>
<td>Liberia</td>
</tr>
<tr>
<td>LS</td>
<td>Lesotho</td>
</tr>
<tr>
<td>LT</td>
<td>Lithuania</td>
</tr>
<tr>
<td>LU</td>
<td>Luxembourg</td>
</tr>
<tr>
<td>LV</td>
<td>Latvia</td>
</tr>
<tr>
<td>MC</td>
<td>Monaco</td>
</tr>
<tr>
<td>MD</td>
<td>Republic of Moldova</td>
</tr>
<tr>
<td>MG</td>
<td>Madagascar</td>
</tr>
<tr>
<td>MK</td>
<td>The former Yugoslav Republic of Macedonia</td>
</tr>
<tr>
<td>ML</td>
<td>Mali</td>
</tr>
<tr>
<td>MN</td>
<td>Mongolia</td>
</tr>
<tr>
<td>MR</td>
<td>Mauritania</td>
</tr>
<tr>
<td>MW</td>
<td>Malawi</td>
</tr>
<tr>
<td>MX</td>
<td>Mexico</td>
</tr>
<tr>
<td>NE</td>
<td>Niger</td>
</tr>
<tr>
<td>NL</td>
<td>Netherlands</td>
</tr>
<tr>
<td>NO</td>
<td>Norway</td>
</tr>
<tr>
<td>NZ</td>
<td>New Zealand</td>
</tr>
<tr>
<td>PL</td>
<td>Poland</td>
</tr>
<tr>
<td>PT</td>
<td>Portugal</td>
</tr>
<tr>
<td>RO</td>
<td>Romania</td>
</tr>
<tr>
<td>RU</td>
<td>Russian Federation</td>
</tr>
<tr>
<td>SD</td>
<td>Sudan</td>
</tr>
<tr>
<td>SE</td>
<td>Sweden</td>
</tr>
<tr>
<td>SG</td>
<td>Singapore</td>
</tr>
<tr>
<td>SI</td>
<td>Slovenia</td>
</tr>
<tr>
<td>SK</td>
<td>Slovakia</td>
</tr>
<tr>
<td>SN</td>
<td>Senegal</td>
</tr>
<tr>
<td>SZ</td>
<td>Swaziland</td>
</tr>
<tr>
<td>TD</td>
<td>Chad</td>
</tr>
<tr>
<td>TG</td>
<td>Togo</td>
</tr>
<tr>
<td>TJ</td>
<td>Tajikistan</td>
</tr>
<tr>
<td>TM</td>
<td>Turkmenistan</td>
</tr>
<tr>
<td>TR</td>
<td>Turkey</td>
</tr>
<tr>
<td>TT</td>
<td>Trinidad and Tobago</td>
</tr>
<tr>
<td>UA</td>
<td>Ukraine</td>
</tr>
<tr>
<td>UK</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>US</td>
<td>United States of America</td>
</tr>
<tr>
<td>UZ</td>
<td>Uzbekistan</td>
</tr>
<tr>
<td>VN</td>
<td>Viet Nam</td>
</tr>
<tr>
<td>YU</td>
<td>Yugoslavia</td>
</tr>
<tr>
<td>ZW</td>
<td>Zimbabwe</td>
</tr>
</tbody>
</table>
SYSTEM AND METHOD FOR INTERACTIVELY DISPLAYING
A MODEL HAVING A COMPLEX SURFACE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to an improved display system and, more specifically, it relates to a system for interactively displaying a model and, most specifically, is particularly advantageous in interactively displaying a geometric model having a complex surface from a plurality of eye-points. The invention also relates to an improved display method.

2. Description of the Prior Art

Non-Uniform Rational B-Spline (NURBS) surfaces are routinely used to represent large-scale models (e.g., without limitation, geometric models representing automobiles, submarines, airplanes, building architectures, sculptured models, mechanical parts, other models for applications involving surface fitting over scattered data or surface reconstruction) for computer graphics, geometric modeling, computer-aided design (CAD), computer-aided manufacturing (CAM), visualization, and animation applications. Many applications like interactive walkthroughs and design validation need to interactively visualize these complex surface models.

This class of representation includes tensor-product and triangular Bézier surfaces. Often these surfaces are approximated with triangles for efficient display on current graphics systems. Bézier surfaces defined over a triangular domain are often called Bézier triangles. Bézier triangles were later generalized to tensor-product surfaces. The theory of triangular NURBS is disclosed by W. Dahmen, et al., Blossoming begets B-spline bases built better by B-patches, Mathematics of Computation, 59(199):97-115, 1992.

In order to exploit recent advances in triangle rendering capabilities of graphics systems, it is common to generate a one-time polygonal approximation of the surfaces of a model and discard the analytic representation. Unfortunately, such polygonal approximations require an unduly large number of polygons, thereby necessitating the need for polygon simplification. What is more, due to discretization of geometry, even a dense tessellation is sometimes inadequate for zoomed up views. On the other hand, recent advances in efficient tessellation of surfaces allows the dynamic generation of an appropriate number of triangles from the analytic representation based on the user’s location or eye-point in a virtual environment.
Curved surface rendering has been an active area of research for more than two decades. A number of techniques for rendering splines based on polygonization, ray-tracing, scan-line conversation and pixel-level subdivision have been proposed. The fastest known algorithms are based on polygonization.

Algorithms based on view-dependent polygonization have been proposed to render spline surfaces. However, the fastest algorithms and systems can only render models composed of a few thousand spline patches at interactive rates on high-end graphics systems (e.g., SGI Onyx with a RealityEngine graphics accelerator). None of these approaches are suitably efficient on current graphics systems. In order to exploit fast triangle rendering hardware, recent research has focused on generating polygonal approximations of surfaces. See, for example, S. Kumar, et al., Interactive display of large NURBS models, *IEEE Transactions on Visualization and Computer Graphics*, 2(4):323-336, Dec. 1996.

A one-time static tessellation of many real-world models usually requires the generation of millions of triangles. Most current graphics systems are not able to render these many triangles at interactive speeds (e.g., more than about 5-10 times a second).


The prior approaches often require management of large amounts of data and are still unable to generate more detail than the initial tessellation. Moreover, it seems unnatural that in order to render surfaces, one first generates a large number of triangles and then finds a way to reduce the count.

On the other hand, efficient techniques for dynamic surface tessellation have made it possible to approximate surfaces with a small number of triangles in the first place. See, for example, A. Rockwood, et al., Realtime rendering of trimmed surfaces, *ACM Computer Graphics*, 23(3):107-117, 1989; S. Abi-Ezzi, et al., Tessellation of curved surfaces under highly varying transformations, *Proceedings of Eurographics*, pp. 385-97, 1991; S. Kumar, et al., Interactive display of large NURBS

S. Kumar, et al., Accelerated walkthrough of large spline models, discloses surface simplification. Most surface triangulation algorithms produce at least two triangles for each tensor product Bézier patch. Furthermore, each trimming curve must be tesselated into at least one edge and this adds further to the triangle count. For example, a relatively small part like a "Utah teapot" consists of 32 Bézier patches. However, 64 triangles are not required to represent it when its size is, say, a few pixels on screen. As part of pre-processing, polygonal simplifications of a mesh of trimmed Bézier patches are computed. Adjacent surfaces are combined and polygonal approximations for super-surfaces are computed. For a super-surface with \( n \) Bézier patches, polygonal simplifications with less than \( 2n \) triangles are generated. Surface patches and their simplifications are represented using spatial hierarchies. Given a view-point, an appropriate polygonal approximation is computed based on surface simplification and incremental triangulation. Since two adjacent super-surfaces may be rendered at different detail, algorithms prevent cracks between these super-surfaces.

Algorithms which approximate surfaces with a small number of triangles provide much finer control on the update of detail as compared to approaches which reduce the count of the triangles as in J. Xia, et al., A dynamic view-dependent simplification for polygonal models, *Proc. IEEE Visualization*, pp. 327-34, 1996; and H. Hoppe, View dependent refinement of progressive meshes. Accordingly, the dynamic tessellation approach also promises to be more scalable and efficient, in general.

A combination of on-line tessellation (for high-detail objects) and off-line tessellation combined with multiple levels of detail (for lower detail objects) has also been used (see S. Kumar, et al., Accelerated walkthrough of large spline models).

While model sizes in new applications will continue to grow, the representation for a model that must be processed every frame need not grow linearly. In other words, more information may be compressed in a small number of higher-order primitives like surfaces and super-surfaces (see S. Kumar, et al., Accelerated walkthrough of large spline models).
For these reasons, there remains a very real and substantial need for an improved system for interactively displaying a model having a complex surface at still faster speeds.

**SUMMARY OF THE INVENTION**

The present invention has met this need by providing a system for interactively displaying a model having a complex surface. The system comprises: means for pre-processing the model to provide a plurality of surface primitives; means for displaying the model with respect to a first eye-point of a plurality of eye-points and for selecting a second eye-point of the eye-points for display of the model with respect to the second eye-point; means for providing a value for each of the surface primitives with respect to the first eye-point; means for selecting surface primitives for display with respect to the second eye-point as a function of the first and second eye-points and the value with respect to the first eye-point for each of the surface primitives; means for tessellating each selected surface primitive to provide a plurality of triangles or triangle-strips representative of the selected surface primitive with respect to the second eye-point; and means for displaying the triangles or triangle-strips to display the model with respect to the second eye-point.

As another aspect of the invention, a method interactively displays a model having a complex surface. The method comprises: pre-processing the model to provide a plurality of surface primitives; displaying the model with respect to a first eye-point of a plurality of eye-points; selecting a second eye-point of the eye-points for display of the model with respect to the second eye-point; providing a value for each of the surface primitives with respect to the first eye-point; selecting surface primitives for display with respect to the second eye-point as a function of the first and second eye-points and the value with respect to the first eye-point for each of the surface primitives; tessellating each selected surface primitive to provide a plurality of triangles or triangle-strips representative of the selected surface primitive with respect to the second eye-point; and displaying the triangles or triangle-strips to display the model with respect to the second eye-point.

The invention will be more fully understood from the following detailed description of the invention on reference to the illustrations appended hereto.
BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a software block diagram illustrating three separate layers of an interactive display library in accordance with the invention;

Figure 2 is a hardware block diagram illustrating four separate processors for use with the interactive display library of Figure 1;

Figure 3 illustrates a triangular domain and barycentric coordinates;

Figures 4A-4C illustrate a super-surface hierarchy in which three levels of the hierarchy of a "Utah teapot" are represented;

Figure 5A shows a super-surface construction for a uniform tessellation (dashed lines) of a single super-surface in its domain;

Figure 5B shows four adjacent super-surfaces and their constituent patches;

Figure 6 shows a back-patch computation in which the bounding volume of a patch defines a back-space such that if the normal direction lies entirely in the back-space, then all points on the patch are back-facing;

Figures 7A-7C illustrate triangulation in the triangular domain;

Figure 8 is a plot of model size versus average memory size per patch;

Figure 9 is a plot of model size versus average time to render;

Figure 10A is a flowchart of a visibility test in accordance with the invention; and

Figure 10B is a flowchart of a tessellation test in accordance with the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring to Figure 1, an interactive display library 1 is divided into three exemplary layers including a systems layer 2, a surface layer 4 and a utility layer 6. The exemplary library 1 interactively displays models with trimmed NURBS surfaces. The library 1 includes a suite of algorithms and is based on dynamic triangulation of groups of surface primitives.

The five major components of the library 1 are: (1) surface pre-processing as provided by the model hierarchy manager 7 of the systems layer 2, and the surface triangulator algorithm 8 and the trimming curve processor algorithm 10 of the surface layer 4; (2) pipeline management provided by the pipeline manager 12; (3)
visibility computation by the visibility thread 14; (4) tessellation by the tessellation thread 16; and (5) triangle rendering by the rendering thread 18.

Referring to Figures 1 and 2, the pipeline manager 12 monitors the performance of the processors 20,22,24,26 and the graphics board 27 to tune the system for performance improvement. The library 1 provides users with the ability to configure the rendering pipeline based on the available hardware (i.e., the number of processors and the quality of the graphics board). It will be appreciated that while reference has been made to the exemplary four processors 20,22,24,26, a wide range and count (e.g., 1 or more) of other processors such as, for example, mainframe computers, mini-computers, workstations, personal computers, microcomputers, and other microprocessor-based computers, central processing units (CPUs) or networked processors may be employed.

The library 1 incorporates all software components of a surface display system except the application program 43 and user interface (e.g., OpenGL layer 28), for efficient rendering of parametric surfaces. The library 1 may be used to read a list of tensor-product or triangular NURBS or Bézier surfaces and render them. The internal representation is based on the Bézier form. More importantly, the library 1 seamlessly integrates the different algorithms to obtain rendering speeds not obtained before. In addition, the library 1 performs run-time performance monitoring to dynamically rearrange the surface rendering pipeline.

Each module, Mₙ, of the library 1 is preferably designed such that if it needs the results of another module, Mₙ, the appropriate data structures of Mₙ are accessible in its local space. If Mₙ is disabled, Mₙ can be configured to either disable itself automatically or enable Mₙ via a call-back. The library 1 also provides support for multi-processor systems. A lock-free synchronization scheme (see S. Kumar, et al., Scalable algorithms for interactive visualization of curved surfaces, *Supercomputing*, 1996) is employed to reduce the number of system calls needed for mutual exclusion by appropriately ordering read-write operations.

A function in any of the layers 2,4,6 has access to all functions in the layers below it. However, most calls are only to functions in the layer immediately below. Functions in the system layer 2 typically call other functions of that layer via call-backs. The current scheme provides a number of plug-in opportunities for exploring trade-offs via function pointers. To avoid the overhead associated with
function pointers, the library 1 also allows hard-wiring of function calls using conditional compiling.

As a part of pre-processing by the model hierarchy manager 7, a hierarchical representation of the model is generated, adjacencies between surfaces are computed, and surface properties like maximum magnitudes of derivatives and bounds on Gauss map are generated. NURBS representation is also converted to Bézier at this stage. Complex Bézier surfaces are further subdivided into simpler parts, as well. Adjacent Bézier patches are hierarchically combined into super-surfaces as discussed below in connection with Figures 5A-5B.

The pipeline manager 12 functions like a job scheduler, except jobs are not separate processes. The default initial order of jobs may be changed via calls to library routines. This order dynamically changes at rendering time when a system bottleneck is discovered. In order to keep the system stable, reconfigurations are not performed unless a bottleneck persists. Also, the application program 43 in the applications layer 29 may disable the dynamic reconfiguration altogether.

The library 1 employs the visibility thread 14 to determine the visibility of surface primitives (e.g., Bézier patches or super-surfaces). A visibility primitive is the smallest entity which may be deemed by the library 1 to be visible or invisible. A visibility primitive is a node in the hierarchical representation of the model. Not all surface primitives need be at the same level at a time. For example, one surface primitive could be a single Bézier patch, while another is a super-surface (i.e., a collection of patches). The library 1 includes a variety of view-frustum and back-patch culling routines. Different types of bounding box hierarchies are available and one (or more) of them may be activated. Additionally, the application program 43 may provide its own visibility processor.

The tessellation thread 16 triangulates one surface primitive at a time. The library 1 tessellates a surface primitive based on a number of different quality constraints (as discussed below in connection with Figures 5A-5B and 6), one or more of which may be enabled. Using the current viewing parameters and the stored surface properties (generated by the model hierarchy manager 7), the library 1 computes the sampling size required for each surface primitive. These samples are subsequently triangulated in a crack-free manner. The library 1 directly generates triangle-strips, using an incremental algorithm that exploits coherence.
The triangle rendering thread 18 is the interface to the OpenGL layer 28. It passes triangle-strips to the graphics board 27. The exemplary embodiment employs OpenGL, although it can easily be modified for a wide variety of other graphics standards (e.g., without limitation, DIRECT-3D, PHIGS).

Any of the foregoing five major components may be turned on or off. The exemplary library 1 renders more than 50,000 tensor-product and triangular Bézier patches up to 5 times a second on the exemplary Onyx2 30 which employs the four processors 20, 22, 24, 26 and the exemplary InfiniteReality graphics board 27. The library 1 also outperforms simple triangle rendering on a variety of other platforms such as, for example, without limitation, Silicon Graphics O2 employing IRIX (i.e., a version of Unix), INDIGO2 Extreme, Octane, other Onyx systems, and PCs running System V or BSD Berkeley Unix, Windows 98 or Windows NT. The library 1, CPU 30 including OpenGL layer 28, and application program 43 form a system 44 for interactively displaying a model having a complex surface.

Referring to Figure 3, a triangular domain and barycentric coordinates are illustrated. A rational tensor-product Bézier patch, $F(u,v)$, of degree $m \times n$, defined for $(u,v) \in [0,1] \times [0,1]$, is specified by a mesh of control points, $p_{ij}$, and their weights, $w_{ij}$:

$$F(u,v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} P_{ij} B_{i}^{m}(u) B_{j}^{n}(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} B_{i}^{m}(u) B_{j}^{n}(v)} \quad (1)$$

wherein $0 \leq i \leq m$;

$0 \leq j \leq n$; and

the Bernstein function is:

$$B_{i}^{m}(u) = \binom{m}{i} u^{i} (1-u)^{m-i}$$

The normal direction is given by $N(u,v) = F_{u} \times F_{v}$, where $F_{u}$ and $F_{v}$ are the partial derivatives of $F(u,v)$.

The library 1 employs the barycentric coordinates of the domain triangle as shown, for example, by the triangle abc of Figure 3. Each point on the domain is represented by three coordinates, $u$, $v$, and $w$. The coordinates of a point $q$ are:
\[ u = \frac{A(q,b,c)}{A(a,b,c)}, \quad v = \frac{A(a,q,c)}{A(a,b,c)}, \quad w = \frac{A(a,b,q)}{A(a,b,c)} \]  

(2)  

wherein \( A \) is the area function; and  

\( u + v + w = 1 \), and, hence, there are only two independent variables

A triangular Bézier patch of degree \( n \) is given by:

\[
F(U) = \sum_{|I| = n} w_I p_I B_I^n(U) = \frac{P(U)}{W(U)}
\]  

(3)  

wherein \( U \) is a point on the domain represented by some \((u,v,w)\);

\( I \) is a vector of three numbers \((i,j,k)\);

\(|I| = i + j + k;\)

\[
B^n_I(U) = \frac{n!}{i!j!k!} u^i v^j w^k;
\]  

(4)

\( i + j + k = n; \)

\( \Sigma \) is over enumeration of \((i,j,k)\) that sum to \( n \) \((i.e., i+j+k = n)\);

\( p_I \) is a control point; and

\( w_I \) is the weight of the control point

The control points \( p_I \) form a triangular mesh of control points. The three boundary curves of a triangular Bézier patch, \( u=0, v=0, \) and \( w=0, \) are all Bézier curves. Instead of partial derivatives with respect to the two axes of the rectangular domain, directional derivatives are computed. The cross product of two directional derivatives produces the normal direction at a point on the surface.

The directional derivatives of a Bézier surface are also triangular Bézier surfaces themselves. In general, the polynomial derivative in the direction \( d' = (d,e,f) \) is:

\[
D_{d'}(P(U)) = n \sum_{|I| = n-1} q_I B_I^{n-1}(U)
\]  

(5)  

wherein \( q_I = [d w_{i+1} p_{i+e} + e w_{i+e} p_{i+e} + f w_{i+e} p_{i+e}]; \)

\( eI = (1,0,0); \)
\[ e2 = (0,1,0); \text{ and} \]
\[ e3 = (0,0,1) \]

In particular, the derivatives along the \( u \), \( v \) and \( w \) axes, respectively, are:

\[ D_u(p(U)) = n \sum_{|I|=n-1} (p_{I+e1} - p_{I+e3}) B_I^{n-1}(U) \quad (6) \]

\[ D_v(p(U)) = n \sum_{|I|=n-1} (p_{I+e2} - p_{I+e3}) B_I^{n-1}(U) \quad (7) \]

\[ D_w(p(U)) = n \sum_{|I|=n-1} (p_{I+e3} - p_{I+e1}) B_I^{n-1}(U) \quad (8) \]

wherein the quotient rule for derivatives of rationals is employed.

As employed by the library 1 during rendering, "super-surfaces" are a dynamically selected collection of patches, rather than a static cluster as shown by the prior art. The exemplary library 1 tessellates super-surfaces at rendering time and treats them as first class primitives, thereby obtaining a better control on the triangulation generated for interactive update of the frames. In addition to reducing the complexity of crack-prevention, this parametric scheme also allows the library 1 to build a hierarchy of super-surfaces. The library 1 performs simplification of a small number of surfaces directly, as opposed to that of a potentially large number of polygons.

While S. Kumar, et al., Interactive display of large NURBS models, and S. Kumar, et al., Accelerated walkthrough of large spline models, report that the most expensive operation in their system is testing if the tessellation of a surface patch needs to change from the current frame to the next, the exemplary library 1 improves the performance of this test by using a fractional condition framework. Instead of executing the loop "for each patch, if processing needed, then process patch", the library 1 computes the change required in viewing parameters to warrant re-processing of a given surface primitive. Note that the viewing parameter changes just once a frame. In this manner, the library 1 reduces both the time for testing if a surface primitive is visible, and the time for testing if the surface primitive needs re-tessellation.
Other important contributions of the library include: (1) dynamic super-surface construction and tessellation; (2) computation of patch adjacencies and construction of a hierarchical surface representation; (3) an improvement of the back-patch culling algorithm of S. Kumar, et al., Hierarchical visibility culling for spline models, *Proc. Graphics Interface*, pp. 142-50, 1996; (4) triangle reduction when trimming curves are specified in a piece-wise linear form; (5) extension of algorithms for tensor product Bézier surfaces to triangular Bézier surfaces; and (6) crack-free tessellation of models with both triangular and tensor-product surfaces (also, a simple extension of the triangular scheme works for S-patches with n-sided domains as well).

As shown in Figures 4A-4C, a super-surface hierarchy in which three levels of the hierarchy of the exemplary "Utah teapot" are represented. Each super-surface is colored in a single color. Figure 4C is the highest level, Figure 4B is the intermediate level, and Figure 4A is lowest level of the exemplary hierarchy.

The input to the library is specified as an unordered list of triangular Bézier or NURBS surface patches. Optionally, each surface patch may include trimming curves specified in Bézier, NURBS, or piece-wise linear forms. While representing smooth curves as polygonal chains is contrary to the exemplary design of the library, many real-world models do maintain a dense sampling of trimming curves. For example, constructive solid geometry modelers generate curves of intersection as the trimming curves. It is often not possible to find an exact rational parametrization of these curves in the domains of the corresponding intersecting surfaces.

NURBS curves and surfaces are subdivided into Bézier form using knot insertion as shown by G. Farin., *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, 1993. Additionally, each Bézier trimming curve is also subdivided at patch boundaries. For the following, the input to the library is assumed, for convenience of description, to be in the Bézier form.

Referring to Figures 5A-5B, the super-surface construction of Figure 5A shows a uniform tessellation (dashed lines) of a single super-surface 31 in its domain. Domains of its exemplary eight constituent patches are marked with solid lines in Figure 5A. Figure 5B shows four adjacent super-surfaces 32,34,36,38 and their constituent patches (*e.g.*, each of the super-surfaces 32,34 has 6 patches). Special points are marked in bold.
For each Bézier surface, the model hierarchy manager 7 pre-computes the bounding boxes of its Gauss maps and the maximum magnitudes of its partial derivatives. Such data are termed surface properties herein. The volume of the bounding box of a Gauss map is referred to as its magnitude.

Each complex Bézier surface is subdivided. The complexity of a Bézier patch, F, is defined as complex by the following algorithm:

First, subdivide F into F₁ and F₂, at u (or v) = 1/2.

Next, if:

\[
\frac{1}{2} < \left| \frac{\mathcal{P}(F) - \mathcal{P}(F_i)}{\mathcal{P}(F)} \right| < 2
\]

wherein \( \mathcal{P} \) denotes a property; and

\[ i \in \{1,2\} \]
then, output F since it is not-complex.

Otherwise, since F is complex, output F₁ and F₂.

The library 1 performs uniform subdivision of the domain. This subdivision of complex patches ensures that the simpler half is not over-tessellated to satisfy quality constraints on the complicated half.

Before the library 1 constructs the surface hierarchy, it derives adjacencies between surface patches. The library 1 employs an extension of the scheme proposed in G. Barequet, et al., Repairing CAD models, *Proc. IEEE Visualization*, pp. 363-70, 1997. If boundary control polygons p₁ and p₂ of patches P₁ and P₂, respectively, match exactly, then the library 1 draws an edge between P₁ and P₂ in the adjacency graph. Otherwise, if the curves do not match exactly, then the library 1 employs an adjacency score (based on relative distances to neighboring boundary curves) to assign adjacencies.

The surface hierarchy is constructed in a bottom-up fashion. At each level, i, the library 1 employs a technique similar to S. Kumar, et al., Accelerated walkthrough of large spline models, to construct a *super-surface* (i). First, grow *super-surfaces* (i-1) into larger *super-surfaces* (i). Then, continue to merge adjacent super-surfaces of level -1 until the number of super-surfaces reduces approximately by a factor of log.
\[ |\text{super-surface}(i)| \leq \frac{|\text{super-surface}(i-1)|}{U_k \log |\text{super-surface}(i-1)|} \]

wherein \( U_k \) is a small constant (e.g., 2)

Mergers between candidate pairs of super-surfaces proceed in the order of increasing relative difference in the magnitude of their properties. In practice, the library 1 typically constructs about five to eight levels for the best performance. More levels only add extra overhead. At rendering time, this hierarchy is traversed. Based on the viewing parameter, a super-surface is either further expanded, or else is output as a primitive for subsequent tessellation and rendering.

The properties of a super-surface are obtained by a union of the properties of the component surfaces. Note that when super-surfaces are merged, their trimming curves with overlapping edges are also merged. Since not all super-surfaces are tessellated at the same level, it is important to maintain a valid representation of the boundary curves between super-surfaces. In particular, there are special points on the boundary curves of a super-surface that must be included in any sampling: points adjacent to more than two super-surfaces are special points. These are exactly the domain corners, such as 40,42, of the super-surfaces of Figure 5B.

S. Kumar, et al., Accelerated walkthrough of large spline models, discloses a technique to combine adjacent Bézier patches into super-surfaces. Each super-surface is triangulated and various levels of detail (LODs) (see J. Cohen et al., Simplification envelopes) of the triangulation are generated. At rendering time, the appropriate level of detail is displayed for each super-surface based on its distance from the viewer. If the highest level of detail of a super-surface is insufficient, then each Bézier patch of the super-surface is triangulated separately. Extra processing is required to prevent cracks between super-surfaces at different levels of detail.

Incremental tessellation and rendering of Bézier patches is discussed in S. Kumar, et al., Interactive display of large NURBS models. In brief, referring to Figure 6, view-frustum and back-face culling are first performed, after which, by using the bounding volume of a Bézier patch, \( F \), the back-space, an open cone \( S_B \), is computed. If \( N = F_u \times F_x \) is contained in the back-space \( S_B \), then \( F \) is back-facing. It will be appreciated that this approach uses bounding volumes for both \( F \) and \( N \), while, as discussed below in connection with Equations 11-14 and the fractional
condition test, the library 1 reduces this to just one bound and improves the accuracy of the test.

Continuing to discuss the prior incremental tessellation and rendering of Bézier patches, the sampling is tested to satisfy one or more of the screen-space quality constraints: bounded triangle size, bounded normal deviation, bounded surface deviation, and bounded normal difference. The number of samples needed is computed as $s_r(B_r) \times \mathcal{P}(F)$, where $s_r$ is the scaling factor for the bounding box of $F$ and $\mathcal{P}(F)$ is a pre-computed property of the surface. The scaling factor is the maximum ratio between any unit vector in $B_r$, and its projection on-screen. Finally, if a sampling different from the previous frame is required, the interior of the patch and the boundary curves are re-sampled and the new set of samples are incrementally re-triangulated. While the prior art shows that up to 30% of processor time is spent in computing $s_r$ at rendering time, as discussed below in connection with the improved back-patch computation, the library 1 reduces this time.

For the improved back-patch computation disclosed herein, a rational Bézier function, $F(u,v)$, is denoted as $\frac{f(u,v)}{w(u,v)}$. For simplicity of presentation, the parameters $(u,v)$ are eliminated from the subsequent disclosure, as they are implicit from the context. A Bézier patch, $F$, is back-facing if:

$$\mathbf{N} \cdot (\mathbf{F} - \mathbf{E}) > 0 \quad (11)$$

wherein $\mathbf{N} = \mathbf{F}_u \times \mathbf{F}_v$ for all $(u,v) \in [0,1] \times [0,1]$; and $\mathbf{E}$ is the eye-point as determined by application program 43 of Figure 1.

In turn, the application program 43 displays a model with respect to the eye-point $\mathbf{E}$ and, then, selects another eye-point $\mathbf{E}'$ to display the model with respect to that eye-point $\mathbf{E}'$. The exemplary visibility test may also be written as:

$$\mathbf{N} \cdot \mathbf{F} > \mathbf{N} \cdot \mathbf{E} \quad (12)$$

wherein $\mathbf{F} = \frac{f}{w}$ is set forth in Equation 1;
\[ N \cdot F = F_u \times F_v \cdot F = \frac{w f_u - f w_u}{w^2} x \frac{w f_v - f w_v}{w^2} \cdot \frac{f}{w}; \] (13)

\[ F_u \times F_v \cdot F = \frac{w f_u x f_v - w f u x f - w f u x f_v}{w^3} \cdot \frac{f}{w} = \frac{f_u x f_v}{w^3}; \] (14)

\( f_u \) is \( \delta f/\delta u; \)
\( f_v \) is \( \delta f/\delta v; \)
\( w_u \) is \( \delta w/\delta u; \) and
\( w_v \) is \( \delta w/\delta v \)

Therefore, if \( F \) is an \( m \times n \) rational Bézier patch, and \( N \) is a \( 3m \times 3n \) rational Bézier patch, then \( N \cdot F \) is a \( 3m-1 \times 3n-1 \) rational Bézier patch. At rendering time, the library 1 checks if \( N \cdot E \) is greater than \( m_F \), a pre-computed surface property equal to the minimum value of \( N \cdot F \), such that the test of Equation 12 is safely met. This test can be performed if \( p_q \cdot E \) is greater than \( m_F \), where \( p_q \) are the control points of rational Bézier patch \( N \). In practice, the library 1 just tests against the bounding box of the control points. Since the library 1 pre-computes the exact \( m_F \) and employs a bound just once in the exemplary technique, this visibility test is about 20\% more accurate in practice than that of known previous attempts.

It will be appreciated that both the test if a Bézier patch is visible and the test if a patch needs to be re-tesselated are applicable directly to super-surfaces. In particular, the pre-computed surface visibility property, \( m_p \), for any super-surface, \( P \), is the minima of \( m_F \) of all its constituent patches. Similarly, the scaling factor, \( s_p \), of a super-surface, \( P \), is the maxima of the scaling factors, \( s_F \), of all its constituent patches. The bounding volume of \( N \) is the union of the bounding volumes of all the constituent patches. Hence, the library 1 does not need to compute an analytical representation for a super-surface.

To perform an \( n_u \times n_v \) tessellation of a super-surface, \( P \), the library 1 simply considers the union of the domains of all its constituent patches, arranged in an arbitrary plane as shown in Figure 5A. Each sample point of \( P \) is contained in exactly one of the constituent domains. The library 1 employs the representation of the corresponding patch to raise the sample to the three dimension Euclidean space (\( \mathbb{R}^3 \)). Again, boundary curves are tessellated separately. Triangulation between the interior of a super-surface and the boundary curves continues to be a planar
triangulation just as in the simple Bézier patch case. The rest of the algorithm remains the same.

As mentioned earlier, two of the most important operations in prior art surface rendering are the tests: (1) whether a primitive, P, is visible; and (2) whether the current sampling of P is sufficient. As compared to these prior tests, the library introduces a significant speed-up by changing the "whether" to "how much". Instead of Boolean answers, the library employs a fractional response and, hence, expresses "visibility" and "re-tessability" as real numbers. In this manner, a high magnitude of the visibility value, \( d_v \), of a surface primitive implies that it is highly invisible and need not be re-tested for small changes to the eye-point (e.g., view-point) as provided by application program 43. Similarly, a high magnitude of the re-tessability value, \( d_t \), of a surface primitive implies that the primitive need not be re-tested for small changes in the eye-point.

The visibility value, \( d_v \), of a surface primitive, P, is defined as the minimum distance the eye-point must move until visibility of P needs to be re-computed. If the eye-point moves less than \( d_v(P) \), then P continues to be either visible or hidden.

Re-tessability, \( d_t \), is the minimum distance the eye-point must move until the primitive must be retested to check if re-tessellation is needed.

The computation of fractional back-patch visibility is set forth below. It will be appreciated that the other fractional conditions are computed in an analogous manner. For example, the visibility is computed for frame \( i \) by comparing:

\[
v_i = \mathbf{N} \mathbf{E} \text{ to } m_f
\]

wherein \( m_f \) is a pre-computed surface property equal to minimum value of \( \mathbf{N} \mathbf{E} \); \( N = F_x \times F_y \) is a 3m x 3n rational Bézier patch; and \( E \) is the eye-point for frame \( i \).

For frame \( i+1 \), the visibility is computed by comparing:

\[
v_{i+1} = \mathbf{N} \mathbf{E}' \text{ to } m_f
\]

wherein \( E' \) is the eye-point for frame \( i+1 \).

As discussed above, the visibility need not be re-computed for frame \( i+1 \) if:

\[
\mathbf{N} \mathbf{E}' < m_f
\]

or if:

\[
\mathbf{N} \mathbf{(E}' - E) < m_f - \mathbf{N} \mathbf{E}
\]
\[ N \cdot (E' - E) < m_p - v_i \] 

(19)

Hence, while the maximum value of \( N \cdot (E' - E) \) remains less than \( m_p - N \cdot E \), the visibility need not be re-computed.

The maximum value of \( N \cdot (E' - E) \) is \(| N | \cdot | E' - E | \). This implies that re-testing is not required for:

\[ | E' - E | < d_r = \frac{m_p - v_i}{|N|} \] 

(20)

wherein \(| E' - E |\) is movement (i.e., change in distance) of the eye-point; and \( m_p \) and maximum value of \(| N |\) are pre-stored properties of primitives.

S. Kumar, et al., Accelerated walkthrough of large spline models, discloses a sampling of all trimming curves to obtain a piecewise linear representation. However, if the trimming curves are already specified in a highly tessellated form, it is difficult to reduce the number of points on the trimming curves without introducing cracks in the triangulation. Since the exemplary library 1 computes adjacencies, it simplifies the common boundary curve separately and uses the simplified version on both adjacent patches, thereby avoiding cracks. The library 1 performs this simplification by deleting points on the curve as long as each resulting curve segment is less than \( \Delta \) in length, where \( \Delta \) is a user-controlled bound on the maximum triangle-edge length. A problem with this approach is that resulting trimming curve polygon may be self-intersecting.

In order to avoid this problem, the library 1 selects special points on the curve that cannot be deleted. The special points of a trimming curve on a patch domain are selected as follows. First, choose the point \( p_m \) with the minimum internal angle. Second, mark \( p_m \) as special and add it to a polygonal chain which is, thereby, being constructed. Next, add point \( p_m + i \) to the polygonal chain (in which addition is modulo \( n \)). Finally, if a segment of the polygonal approximation of the trimming curve (C) intersects the Convex Hull, as understood by those skilled in the art, of the chain, then \( p_m + i \) is special, set \( m = m + i \), and proceed recursively.

A point on the common boundary curve is marked as special if it is found to be special in either of the two adjacent surface domains. It can be proven that if special points are included, the resulting trimming polygon is guaranteed to be simple.
Often, surfaces are not rectangular in shape. A tensor-product parametrization of these surfaces is degenerate. It is possible to reparametrize a triangular domain onto a rectangular domain (see K. Iino, et al., Subdivision of triangular Bézier patches into rectangular Bézier patches, *Transactions of the ASME*, 1992) and then apply the algorithms designed for tensor-product surfaces, but that defeats the purpose of using triangular Bézier surfaces.

The quality criteria are specified in screen space, whereas sampling is performed in the parametric domain \((u,v)\) of the surface. The relationship between the step size on the domain and the resulting triangle size on the screen is of interest. For the domain \(D\) of a three-dimensional surface \(F\), and points \(p_1\) and \(p_2\) on \(D\), it is assumed that vector \(T(F(p))\) maps the point \(F(p)\) to screen space, that \(|T(F(p_1))T(F(p_2))| = f_1(|F(p_1)F(p_2)|)\), and that \(|F(p_1)F(p_2)| = f_2(|p_1p_2|)\). The function \(f_3 = f_1 \circ f_2\) is of interest, wherein \(f_2\) depends only on the control points of \(F\) and is independent of the viewing parameters. The function \(f_3\) may be computed in two ways: (1) by transforming the surface (control points) into screen space and using transformed representation to compute the bounds on derivatives, and the step sizes as a function of these bounds; or (2) by determining bounds on the surface in object space as a pre-processing step, and at run time, computing the step size as a function of the pre-computed bounds and viewing parameters.

The first method finds \(f_3\) in one step and all computation must be performed at rendering time. On the other hand, the second method factors \(f_3\) into two parts, \(f_1\) and \(f_2\). First, the relationship between the domain step size and the step size in the object space, which is independent of the viewing parameters, is pre-computed and stored. Second, the triangle sizes in the object space are mapped to the screen space to compute the sampling size at rendering time. \(f_1\) depends on how much the viewing and perspective transformations scale the lengths of vectors. The minimum scaling over all vectors that lie in the intersection of the view-volume and the Convex Hull of the surface is computed. For parallel projection, and a linear transformation matrix, the relationship is quite simple, as the scaling is given by the norm of the matrix. For perspective projection, the minimum scaling occurs at one of the hull points. A bounding box, instead of a convex hull, may be employed to reduce computational costs.
Given $\Delta_s$, the step size in the object space, $\Delta_s$, the maximum step size in screen space, is computed as follows. For each point $P_i$ on the hull, $r_i$, the length of the major axis of the ellipse projected onto the screen by the circle, $c_i$, with center $\bar{P}_i$ and radius $\Delta_s$, is computed. $c_i$ lies in a plane perpendicular to $\bar{P}_i E$, where $E$ is the eye-point. Note that a vector perpendicular $\bar{P}_i E$ scales the least on projection, $\Delta_s = \min_i (r_i)$.

The library 1 employs an algorithm, which generalizes to arbitrary sided domains, to directly tessellate triangular Bézier surfaces. The library 1 extends the prior art sampling algorithm of S. Kumar, et al., Interactive display of large NURBS models, to triangular surfaces. The present sampling algorithm satisfies the minimum triangle-edge length quality constraint, with other constraints being similarly extended. This constraint restricts the screen size of each resulting triangle to be no greater than $\Delta$, a user supplied parameter. Given $\Delta$, the domain step-size, $\delta$, is computed in two stages. First, as an off-line process, $\delta_0 (\delta_0 = \Delta / m)$, the number of domain steps needed to ensure that each resulting triangle has no edge larger than $\Delta$, in object space, is determined. Then, at run time, $s_F$, the scaling factor of vectors due to viewing transformation, is determined. A step size of $\delta = \delta_0 s_F$ guarantees that the screen size of each edge of the resulting triangle is less than $\Delta$.

To determine $\delta_0$, the library 1 finds a bound on the maximum magnitude of directional derivatives of a Bézier triangle. The library 1 finds the maxima in all three directions: $u=0$, $v=0$ and $w=0$. This is equivalent to finding the maxima of pair-wise partial derivatives $M_u = |F_u - F_v|$, $M_v = |F_u - F_w|$ and $M_w = |F_v - F_w|$. These maxima are computed using the method of resultants (see, for example, D. Manocha, et al., Algorithms for intersecting parametric and algebraic curves i: simple intersections, ACM Transactions on Graphics, 13(1):73-100, 1994). A domain step-size of $\delta_0$ in the direction parallel to $u=0$ results in an object space step-size of no more than $M_u \delta_0$ (by the mean value theorem). In order to perform the uniform tessellation, the library 1 discards the minimum of the three maxima and picks isoparametric lines parallel to the axes corresponding to the other two maxima.

Interestingly, by generalizing this technique to tensor-product surfaces and computing $F_d = \min(|F_{u+v} \max, |F_{u-v} \max, |F_v \max)$ in addition to $|F_u \max$ and $|F_v \max$, the library 1 reduces the total number of triangles generated by prior
proposals. Instead of using $\frac{M_u}{\sqrt{2}}$ and $\frac{M_v}{\sqrt{2}}$, respectively, for the step-sizes along the u and v axes, $\max(\frac{M_u}{\sqrt{2}}, F_d)$ and $\max(\frac{M_v}{\sqrt{2}}, F_d)$, respectively, are employed.

Similarly to the tensor-product case, to avoid cracks, the library 1 samples the interior separately from the boundary curves and generates filling triangles at three boundaries.

Once the library 1 determines the interior and boundary step sizes, the surface is correspondingly sampled and triangulated. For the step-size needed on the three boundaries being $\delta_1$, $\delta_2$, and $\delta_3$, respectively, and the interior step-size needed along the three directions being $\delta_u$, $\delta_v$, and $\delta_w$, respectively, the library 1 picks the minimum two of the interior step-sizes for uniform tessellation.

The main steps of the algorithm are as follows. For convenience of discussion, but without loss of generality, assume $\delta_u < \delta_w$ and $\delta_v < \delta_w$. Also, let $n_i = \frac{1}{\delta_i}$. First, $n_i$ equi-spaced samples 50 are selected on the $i^{th}$ boundary of boundary 52 (boundary 0), boundary 54 (boundary 1) and boundary 56 (boundary 2) as shown in Figure 7A. Second, offset curves 58 and 60 ($u = \frac{\delta_v}{2}$ and $v = \frac{\delta_w}{2}$, respectively) are generated as shown in Figure 7A, with the factor of $\frac{1}{2}$ being employed to guarantee bounds on edge lengths at the boundary. The strips 62,64 are then triangulated between the two offset curves 58,60 and the corresponding boundary curves 52,54 (i.e., $u=0$ and $v=0$). Next, iso-parametric lines 66,68,70 ($u = (i+\frac{1}{2})\delta_u$ and $72,74,76$ ($v = (j+\frac{1}{2})\delta_v$) are generated, with $i > 1$ and $j > 1$. Then, the iso-parametric lines 66,68,70,72,74,76 are delimited by the offset curve and the boundary 56 ($w=0$) as shown in Figure 7B. Finally, as shown in Figure 7C, diagonals, such as 78, are added to each parallelogram, such as 80, generated in the preceding delimiting step and the envelope of these parallelograms is triangulated with curve 54 ($v=0$).

By tessellating each boundary curve independently of the interior and by avoiding introduction of extra sample points on the curve at the triangulation stage,
the library 1 allows two surfaces adjacent to a boundary curve to be independently parameterized, as long as the common curve is consistent. In particular, one of these could be a triangular surface while the other is a tensor-product surface.

Just as some surfaces are inherently triangular, and are better parameterized on a triangle, there are others that are inherently pentagonal, or hexagonal and so on. There exist parametrizations defined over multi-sided convex domains. J. Gregory, \( C^1 \) rectangular and non-rectangular surface patches, *Surfaces and CAGD*, pp. 25-33, 1983, discloses some of the pioneering work on multi-sided patches. Although previous proposals disclose an algorithm to decompose general multi-sided patches into triangular and rectangular patches, it is more efficient to directly render multi-sided patches.


The exemplary tessellation algorithm for triangular Bézier patches easily extends to polygonal domains. The library 1 still chooses two independent (the independent boundaries of an n-gon have no more than \( \frac{n-1}{2} \) edges between them) boundary curves and generates iso-parametric curves corresponding to the two boundary curves. Offsets are first generated for each of those two curves and the corresponding boundary strips are triangulated. Finally, the envelope of the generated mesh is triangulated with the rest of the boundaries. Interestingly, this general algorithm reduces exactly to the ones discussed above for the cases of triangular and tensor-product patches.

Referring again to Figures 1 and 2, the pipeline of the library 1 has four major stages. In the exemplary embodiment, the pipeline is not a static sequential pipeline and, hence, the stages are termed threads. The four threads are surface-tree construction (C) by the model hierarchy manager thread 7, visibility computation (V) by the visibility thread 14, surface tessellation (T) by the tessellation thread 16, and rendering (R) by the rendering thread 18. The exemplary threads may interact with each other through in-memory data structures, or through files. The file interface may
be extended to use inter-process communication or Unix sockets. The interactions may be synchronous or asynchronous.

The V, T, and R threads are primarily discussed, since the C thread runs just once during pre-processing and, hence, must be completed before the other threads start. In turn, the three threads V, T, R repeat every frame.

Each thread is designed to traverse an associated data structure. During one execution, the library 1 instances a thread one time, although different instances of a thread may be used to compare competing design schemes. Similarly, the data structure traversed by a thread may have several instances.

Each corresponding traverse method may output a primitive that the thread decodes by employing its header. The format for a primitive is extensible. Each thread also provides call-back hooks (i.e., routines that are called just before or just after a thread processes a primitive). There also are hooks for two call-backs, respectively, as the first and the last step of each frame of the thread. Similarly, each data structure provides a call-back hook at each primitive.

For the concept of autonomy, the pipeline manager 12 schedules only autonomous entities for execution. Other routines employed for rendering are called via call-backs by autonomous entities. Any thread may be marked as autonomous. The pipeline manager 12 does not pre-empt any autonomous entity. The library 1 lets the application program 43 controls preemption by allowing an entity to explicitly relinquish control by returning. Before returning, a thread saves its state and sets its active flag to request re-scheduling.

The frame loop, which repeats for each frame, of the pipeline manager 12 is as follows: (1) set all autonomous routines active; (2) while there exists an active autonomous routine R, call R; and (3) if tuning is enabled, then fine tune the pipeline.

The exemplary pipeline manager 12 allows the library 1 to re-arrange the pipeline in a variety of ways. For example, the Surface Tree, as previously constructed by the model hierarchy manager thread 7 (as discussed above in connection with Figures 5A-5B and Equations 9-10), may be declared autonomous.

At each node, the pipeline manager 12 performs the visibility computation, tessellation and rendering functions, in that order, via call-backs.

Alternatively, each thread may be declared autonomous, with thread V producing structure L₁, a list of visible primitives, that thread T, in turn, traverses to
produce \( L_2 \), a list of triangle-strips, which thread \( R \), in turn, traverses and finally sends to the OpenGL server for display by the graphics board 27. In this example, each of the respective data structures provide concurrency control for simultaneous reads and writes.

For an asynchronous rendering scheme, the library 1 disables the concurrency control and allows thread \( T \) and thread \( R \) to maintain consistency by their order of accesses to list \( L_2 \).

As an example of one initial configuration of the pipeline (i.e., the default set-up), \( n \) processes are created, in which \( n \) is user specified and is typically equal to the number (e.g., \( n=4 \) in the exemplary embodiment) of processors in the system.

A default initial 1-process configuration is as follows: (a) the visibility computation thread \( V \) is marked as autonomous; (b) the associated data structure is set to the Surface Tree; (c) the PrimitiveEnd call-back is set to call procedures Tessellate (Primitive) and Render (Primitive); and (d) the pipeline manager 12, PipeManager, is started.

As another example, a default initial 2-process configuration is as follows. For Process 0: (a) the rendering thread \( R \) is marked as autonomous; (b) the associated data structure is set to Render List which is the list of triangles to display; and (c) the PipeManager is started, and for Process 1: (a) the visibility computation thread \( V \) is marked as autonomous; (b) the associated data structure is set to Surface Tree; (c) the PrimitiveEnd call-back is set to call procedures Tessellate(Primitive) and add to Render List; and (d) the PipeManager is started.

All unspecified call-backs are set to a null. If more than two processors are available, then all the other processors start with thread \( T \) as the autonomous entity with the Surface Tree. In Figure 2, there are two processors 20,22 for thread \( T \). The model hierarchy manager 7 and the pipeline manager 12 are distributed and do not reside on any one processor.

In the exemplary embodiment, Surface Tree is actually a forest with a few tens of trees. Load balancing is easily achieved by starting with an equitable distribution of these trees, followed by work stealing as proposed in S. Kumar, et al., Scalable algorithms for interactive visualization of curved surfaces, Supercomputing, 1996.
The exemplary library 1 preferably provides for: (1) increasing the scheduling priority of a bottlenecked autonomous thread; (2) creating an autonomous instance of bottlenecked thread (on a process where it is not autonomous); (3) changing the maximum traversal depth of *Surface Tree*, thereby obtaining either less detailed or more detailed primitives; and (4) dynamic configuration of the pipeline.

For example, if thread A spends more than 10% of its time waiting at a synchronization barrier with thread B, thread B is declared to be bottlenecked. In the exemplary embodiment, a thread must be bottlenecked for at least three frames in any block of five frames to cause an adjustment in the pipeline.

As another example, if the graphics board 27 is found to be the bottleneck, the library 1 changes the maximum traversal depths of threads V and T to offset the bottleneck.

An event recorder module 45 collects statistics on bottlenecks. Preferably, a high precision hardware counter is employed as is available on a Silicon Graphics platform.

The library 1 is preferably implemented using a suitable language (*e.g.*, C/C++) and is independent of the platform. Preferably, shared memory 46 for the processors 20, 22, 24, 26 is controlled by shared memory manager 48 which is also employed to cache triangle-strips.

The library 1 stores the Bézier surface representation (*i.e.*, control points) separately from the tessellated triangles, since the control points remain static, but the cache sizes for primitives may change between frames. Caches are allocated in multiples of 1 Kbytes to avoid fragmentation. The library 1 stores the surface properties with the triangle cache such that if re-tessellation is not required, then no access to the control points’ memory page is needed. Since the exemplary surface properties are less than 50 bytes, it does not impose severe overhead to move them when the cache a primitive requires relocation.

As an alternative, the control points are stored with the cache, although that embodiment is slower to relocate, as the representation of a trimmed Bézier patch may take a few hundred bytes. In addition, a primitive may be a super-surface and may comprise multiple sets of control points.

It is believed that the exemplary library 1 is the first system of its kind for rendering large surface models. While prior proposals for efficient rendering
management are available, none are designed for dynamic super-surface tessellation, nor are they flexible enough to allow the desired types of rendering. Also, it is believed that most other surface rendering systems are more than a hundred times slower than the library 1. Still other prior art systems are little more than a collection of functions for evaluating properties of NURBS surfaces and for triangulating them, but do not form a comprehensive display system.

Compared to the performance of S. Kumar, et al., Accelerated walkthrough of large spline models, the library 1 achieves a speed-up of about 1.5 to 2.5 times. Approximately 40% of this improvement may be attributed to the fractional condition scheme and another approximately 20% to the improved visibility algorithm. Most of the rest of the improvement comes from using parametric super-surfaces. Furthermore, the tessellations produced by using parametrically constructed super-surfaces also satisfy quality constraints not satisfied by the static LODs of S. Kumar, et al., Accelerated walkthrough of large spline models.

To test the performance of the library 1, a variety of Bézier surface models are rendered on a variety of platforms. Speed-ups of 5 to 15 times over a simple polygonal rendering are obtained by the exemplary system. In this comparison, multiple LODs for the polygonal rendering are not employed. The best performance on each system (modulo pipe-line changes) is provided in which all numbers are for synchronous rendering. Asynchronous rendering is two to three times faster, with a degradation in image quality.

On average, the space consumption was observed to be 800 to 900 bytes per Bézier patch (including cached triangles). On some executions, however, the average space per patch exceeded 10 Kbytes for some frames. These were highly closed-up views and a number of non-visible patches still had their caches active and were thereby contributing to the total count. Furthermore, the average space required per patch went down as the size of models went up as shown in Figure 8, since larger models typically have smaller patches on screen.

As shown in Figure 9, the rendering slow-down curve is well below linear with increasing model size. As model sizes continue to grow, primitives processed per frame by the library 1 consist of more Bézier patches. From these observations, it is believed that the approach of dynamic tessellation scales well.
One of the surprising discoveries was the effect of interrupt overhead on the rendering thread R, especially when it is the only autonomous thread on its process. The rendering processor is often blocked waiting for the OpenGL FIFO buffer to be emptied, signifying a rendering bottleneck. This block can cause multiple interrupts on an IRIX system. The rendering thread R on its associated processor waited for OpenGL up to 50% of the time on a dual processor Octane. A significant part of this was attributed to processing interrupts associated with a full FIFO. The total idle time actually went down by about 15 to 20% when sleeps in thread R (sleep times were counted as idle) were artificially introduced. This occurred due to a decrease in the number of interrupts. This result suggests always running a second autonomous thread on the processor running thread R, although at reduced priority.

Figure 10A is a flowchart of the visibility test 82. At 84, the magnitude of the view change vector (\( |E'-E| \)) is determined and, at 85, \( v_i = N \cdot E \) is determined. Next, at 86, if the test of Equation 20 is not met, then, at 88, the current visibility is employed. On the other hand, if that test is met, then, at 90, the visibility is computed as discussed above in connection with Figure 6. After either 88 or 90, if the surface primitive is not visible, then it is discarded at 94. Otherwise, if the surface primitive is visible, then it is tessellated and displayed as discussed below in connection with Figure 10B.

Figure 10B is a flowchart of the tessellation test 96. At 98, the change \((\Delta s_p)\) in the scaling factor \((s_p)\) from the previous frame to the current frame is determined. Next, at 100, if the change in scaling factor \((\Delta s_p)\) is not greater than \(\delta_o\) \((\delta_o = \Delta s_p/m_p)\), then, at 102, the current tessellation is again employed, after which it is rendered at 104. On the other hand, if the change in scaling factor \((\Delta s_p)\) is greater than \(\delta_o\), then, at 106, the tessellation step size is recomputed. Next, at 108, if the current tessellation step size is equal to the previous tessellation step size, then step 102 is executed. Otherwise, at 110, the tessellation is recomputed, after which it is rendered at 104.

At 106, the tessellation step size may be computed by various methods, such as by employing a difference criteria or a deviation criteria. For example, in the difference criteria, the tessellation step size \(\delta\) is chosen per Equation 21:
\[
\delta \leq \frac{\Delta}{|\vec{X}'(t), \vec{Y}'(t), \vec{Z}'(t)|}
\]

wherein \( F(u, v) = (X(u, v), Y(u, v), Z(u, v), W(u, v)) \) is a rational Bézier surface;
\( \delta = 1/n_u \);
for Bézier surface, \( F(u, v) \), the tessellation parameters are computed in
object space as
\[
n_u = \frac{1}{\Delta} \begin{bmatrix}
X(u, v) & Y(u, v) & Z(u, v) \\
W(u, v)_{\alpha} & W(u, v)_{\beta} & W(u, v)_{\gamma}
\end{bmatrix},
\]

\( \frac{X(u, v)}{W(u, v)} \) corresponds to the maximum magnitude of the partial
derivative of \( \frac{X(u, v)}{W(u, v)} \) with respect to \( u \) in the domain \([0, 1] \times [0, 1]\); and
\[
\vec{X}'(t), \vec{Y}'(t), \vec{Z}'(t)
\]
represent the maximum magnitude of \( X'(t), Y'(t), Z'(t) \), respectively, in the domain
\([0, 1]\), at eye-point \( E' \).

The exemplary library 1 is a configurable system useful for interactive
rendering of trimmed tensor product or triangular NURBS surfaces. It also forms a
general test-bed for quickly prototyping new algorithms. In addition, various system
trade-offs can be systematically explored using the library 1, by changing just the part
being profiled and leaving all else constant. As a case in point, a network of
workstations may be employed to enable interactive triangulation and rendering of
large scale models. The library 1 framework has been quite useful in quick testing of
preliminary ideas.

A novel feature of the library 1 is its ability to re-configure the
rendering pipeline to best suit the relative processing powers of the available
processor(s) and graphics board.

The exemplary library 1 may be successfully employed to obtain faster
rendering rates than known prior systems. The library 1 also improves over known
prior art visibility and tessellation algorithms. In addition, the fractional condition
framework of estimating when a primitive needs to be re-evaluated for visibility or for
triangulation completely eliminates the need to process about 20-25% of the model being rendered. In summary, it is believed that the library 1 achieves a speed-up of about 2-3 times over the previously best known algorithms. It is also believed that the library 1 scales almost linearly with increased processing power and deteriorates only sublinearly with increased model size.

Also, algorithms may be employed to efficiently distribute the visibility, computation, and tessellation threads over a network of workstations. In addition, the library 1 provides a flexible framework for designing and testing surface rendering algorithms.

Whereas particular embodiments of the present invention have been described above for purposes of illustration, it will be appreciated by those skilled in the art that numerous variations in the details may be made without departing from the invention as described in the exemplary claims which are appended hereto.
I CLAIM:

1. A system for interactively displaying a model having a complex surface, said system comprising:
   means for pre-processing said model to provide a plurality of surface primitives;
   means for displaying said model with respect to a first eye-point of a plurality of eye-points and for selecting a second eye-point of said eye-points for display of said model with respect to the second eye-point;
   means for providing a value for each of the surface primitives with respect to the first eye-point;
   means for selecting surface primitives for display with respect to the second eye-point as a function of the first and second eye-points and said value with respect to the first eye-point for each of said surface primitives;
   means for tessellating each selected surface primitive to provide a plurality of triangles or triangle-strips representative of said selected surface primitive with respect to the second eye-point; and
   means for displaying the triangles or triangle-strips to display said model with respect to the second eye-point.

2. The system of Claim 1 including said means for pre-processing includes means for providing a Bézier patch as one of said surface primitives.

3. The system of Claim 1 including said means for pre-processing includes means for providing a plurality of Bézier patches adjacent to each other as one of said surface primitives.

4. The system of Claim 1 including said means for pre-processing includes means for providing a plurality of Bézier patches adjacent to each other as a super-surface.

5. The system of Claim 1 including said means for pre-processing includes means for providing a plurality of Bézier patches adjacent to each other as a first super-surface, a plurality of Bézier patches adjacent to each other as a second super-surface, a plurality of Bézier patches adjacent to each other as a third super-surface, and means for
identifying a point adjacent to at least the first, second and third super-surfaces as a special point.

6. The system of Claim 5 including each of said super-surfaces has an interior and a boundary; and said means for pre-processing includes means for processing the interior of each of said super-surfaces, and means for separately processing the boundary between adjacent ones of said super-surfaces.

7. The system of Claim 1 including said means for pre-processing includes means for processing a trimming curve associated with one of said surface primitives.

8. The system of Claim 7 including said trimming curve is selected from the group consisting of Bézier, NURBS, and piece-wise linear forms.

9. The system of Claim 1 including said means for pre-processing includes means for providing a plurality of Bézier patches adjacent to each other as a first super-surface, and a plurality of Bézier patches adjacent to each other as a second super-surface, with said first and second super-surfaces being surface primitives and having a boundary therebetween; and said means for selecting a surface primitive to be displayed includes means for selecting the boundary for display.

10. The system of Claim 1 including said means for tessellating includes means for tessellating Bézier triangles to provide triangular Bézier surfaces.

11. The system of Claim 1 including said means for providing a value includes means for calculating visibility for said first and second eye-points; and said means for selecting a surface primitive to be displayed includes means for determining that recalculation of visibility for the second eye-point is not required when:

$$| E' - E | < \frac{m - v_i}{|N|},$$
wherein \( |E' - E| \) is movement of the first eye-point, \( N \) is a rational Bézier patch, \( m_p \) and \( |N| \) are constants, and \( v \) is \( N \cdot E \) for the first eye-point.

12. The system of Claim 11 including said means for pre-processing includes means for predetermining said constants.

13. The system of Claim 12 including said means for pre-processing includes means for providing a plurality of Bézier patches adjacent to each other as a first super-surface and a plurality of Bézier patches adjacent to each other as a second super-surface, and said means for predetermining said constants includes means for predetermining at least one constant for each of said super-surfaces.

14. The system of Claim 13 including each of said Bézier patches includes at least one pre-computed surface property; and said super-surface which corresponds to said Bézier patches includes at least one pre-computed surface property which is the minima of said at least one pre-computed surface property of said Bézier patches.

15. The system of Claim 13 including each of said Bézier patches includes at least one pre-computed scaling factor; and said super-surface which corresponds to said Bézier patches includes at least one pre-computed scaling factor which is the maxima of said at least one pre-computed scaling factor of said Bézier patches.

16. The system of Claim 1 including said means for displaying said model includes means for displaying said model with respect to a sequence of said eye-points; said means for providing a value includes means for providing said values for each of said eye-points; and said means for selecting surface primitives includes means for selecting said surface primitives for display with respect to a subsequent one of said eye-points of said sequence as a function of the previous one of said eye-points and a value with respect to said previous eye-point for each of said surface primitives.
17. The system of Claim 1 including
said complex surface is a complex parametric surface; and
said means for pre-processing said model provides a plurality of
hierarchical parametric surface primitives for said complex parametric surface.

18. A method for interactively displaying a model having a complex
surface, said method comprising:
pre-processing said model to provide a plurality of surface
primitives;
displaying said model with respect to a first eye-point of a
plurality of eye-points;
selecting a second eye-point of said eye-points for display of said
model with respect to the second eye-point;
providing a value for each of the surface primitives with respect
to the first eye-point;
selecting surface primitives for display with respect to the second
eye-point as a function of the first and second eye-points and said value with respect
to the first eye-point for each of said surface primitives;
tessellating each selected surface primitive to provide a plurality
of triangles or triangle-strips representative of said selected surface primitive with
respect to the second eye-point; and
displaying the triangles or triangle-strips to display said model
with respect to the second eye-point.
**FIG. 2**

**FIG. 3**

*PRIOR ART*

SUBSTITUTE SHEET (RULE 26)
FIG. 5A

FIG. 5B

SUBSTITUTE SHEET (RULE 26)
FIG. 10A

GET MAGNITUDE OF VIEW CHANGE VECTOR (|E' - E|)

\[ |\mathbf{E}' - \mathbf{E}| \]

\[ \mathbf{v}_c = \mathbf{N} \cdot \mathbf{E} \]

USE CURRENT VISIBILITY

\[ \begin{cases} \text{N} & \frac{|\mathbf{E}' - \mathbf{E}|}{|\mathbf{N}|} \leq \frac{|\mathbf{M}_F - \mathbf{v}_c|}{|\mathbf{N}|} \\ \text{Y} & \text{COMPUTE FULL VISIBILITY} \end{cases} \]

DISCARD

VISIBLE?

Y

TESELLATE AND DISPLAY

FIG. 10B

GET CHANGE IN SCALING FACTOR (S_F)

\[ \Delta S_F > S_0 \]

RE-USE CURRENT TESSELLATION

RECOMPUTE TESSELLATION STEP SIZE

IS CURRENT STEP SIZE = PREVIOUS STEP SIZE?

RECOMPUTE TESSELLATION

RENDER
## INTERNATIONAL SEARCH REPORT

**International application No.**

PCT/US99/19024

### A. CLASSIFICATION OF SUBJECT MATTER

- **IPC(6):** G06T 15/40
- **US CL:** 345/423
- According to International Patent Classification (IPC) or to both national classification and IPC

### B. FIELDS SEARCHED

- **Minimum documentation searched (classification system followed by classification symbols):**
  - **U.S.:** 345/423, 421, 427, 441, 442

- **Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched**

- **Electronic data base consulted during the international search (name of data base and, where practicable, search terms used):**
  - ACM Digital Library: viewpoint surface visibility

### C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>US 5,696,837 A (GREEN) 09 December 1997, whole document</td>
<td>1-18</td>
</tr>
<tr>
<td>Y</td>
<td>US 5,261,029 A (ABI-EZZI ET AL.) 09 November 1993, whole document</td>
<td>1-18</td>
</tr>
<tr>
<td>Y</td>
<td>US 5,255,352 A (FALK) 19 October 1993, whole Document</td>
<td>14</td>
</tr>
<tr>
<td>Y</td>
<td>US 5,381,521 A (BALLARD) 10 January 1995, whole document</td>
<td>15</td>
</tr>
</tbody>
</table>

**X** Further documents are listed in the continuation of Box C.  

**See patent family annex.**

<table>
<thead>
<tr>
<th>*</th>
<th>Special categories of cited documents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;A&quot;</td>
<td>document defining the general state of the art which is not considered to be of particular relevance</td>
</tr>
<tr>
<td>&quot;E&quot;</td>
<td>earlier document published on or after the international filing date</td>
</tr>
<tr>
<td>&quot;L&quot;</td>
<td>document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)</td>
</tr>
<tr>
<td>&quot;O&quot;</td>
<td>document referring to an oral disclosure, use, exhibition or other means</td>
</tr>
<tr>
<td>&quot;P&quot;</td>
<td>document published prior to the international filing date but later than the priority date claimed</td>
</tr>
</tbody>
</table>

**Date of the actual completion of the international search:**  

05 OCTOBER 1999

**Date of mailing of the international search report:**  

03 NOV 1999

**Authorized officer:**

- **Name and mailing address of the ISA/US Commissioner of Patents and Trademarks:**
  - Box PCT
  - Washington, D.C. 20231
  - Facsimile No. (703) 305-3230

- **Telephone No.:** (703) 306-2903

Form PCT/ISA/210 (second sheet)(July 1992)
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>MULMULEY, K. Hidden Surface Removal with respect to a moving view point, 1991 ACM 089791-397-3/91/0004/0512</td>
<td>1-18</td>
</tr>
</tbody>
</table>