Predictive fixed-frame rate tessellation of NURBS surfaces

H. Sánchez,[1] A. Moreno[2] and A. García-Alonso[3]

[1] Department of Computer Science, University of Extremadura, Mérida, Badajoz, Spain
[2] VICOMTech, San Sebastián, Gipuzkoa, Spain
[3] Department of Computer Science and Artificial Intelligence, University of the Basque Country, San Sebastián, Gipuzkoa, Spain

Abstract
In this work we propose a predictive fixed-frame rate scheme that shows how to control the visualization frame rate using uniform dynamic tessellation. Similar to previous works[5, 6], tessellation quality criteria is based on the triangle edge length.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling

1. Motivation

Cooperative visualization makes possible the virtual design and maintenance of engineering products through its 3D replica. In collaborative applications, the speed in the making of the following operations is vitally important: the transmission of the entire model, the transmission of messages that update the visualized scene and the generation of the updated final image. Managing to accelerate these operations makes possible a real-time interaction with the 3D model.

3DShared[1, 2] is a freeware cooperative visualization tool developed in the Computer Science School of San Sebastián - Spain. It is totally independent of the platform and it uses the Internet protocols to communicate with the hosts that participate in a cooperative session.

At the moment, 3D models supported by 3Dshared are described by polygonal meshes. As the density of polygons in meshes is pre-calculated, it is possible to find difficulties in model transmission. Furthermore, large CAD models can stretch the capability of some hosts whose graphical features are limited. This fact complicates keeping a similar frame rate in all hosts that take part in a collaborative session.

In many applications of CAD/CAM, virtual reality, animation and scientific visualization, object models are described by NURBS surfaces. This representation allows to define exactly both algebraic geometries and abstract surfaces standardizing the representation by a single mathematical equation. Moreover, thanks to their growing use in CAD/CAM, NURBS surfaces appear in the main neutral file formats for interchanging geometric data, like IGES and STEP. Pielg and Tiller[3] have studied NURBS surfaces in depth.

In Internet oriented applications, because of its implicit compression, NURBS surfaces are more convenient than polygonal meshes because the amount of information to be transferred among hosts is reduced. With regard to the interactive visualization, its description allows to select the optimal level of detail in runtime. Thus, it is possible to manage the quality of the generated image in function of either the degree of interactivity to achieve or the hardware features of a particular host. This means that, in different hosts participating in a collaborative session over the same 3D model, its polygonal approximation can be made up of a different number of polygons.

2. Related Work

2.1. Uniform dynamic tessellation

Because of its importance in computer graphics applications, the rendering of NURBS surfaces has been researched intensively in last three decades. Approximation based rendering algorithms are generally much faster due to recent advances in graphics hardware. In these algorithms, more simple primitives like polygons or points[4] approximate surfaces.

In dynamic tessellation algorithms, meshes are generated in the interactive stage of the rendering pipeline. As they

© The Eurographics Association 2003.

have to run as fast as possible, the tessellations they produce usually are of lower quality than those generated in preprocessing stage. However, there are two interesting facts. In first place, storing NURBS surfaces instead of storing polygonal representations saves a large percentage of available memory. This is especially interesting when using large models where many surfaces are not visible and their incorporation to the potentially visible surfaces set will be gradual. In second place, an appropriate tessellation can be obtained in runtime as the function of actual viewing conditions, the wanted interactivity degree or available graphical hardware features in a local computer. So, this kind of algorithms is very advisable for interactive visualization.

In dynamic tessellation, uniform decomposition is generally used because they are less time consuming than adaptive decomposition. Uniform decomposition tessellates the surface using a regular grid defined in the parametric domain. This does not guarantee that the resulting tessellation will be uniform. Anyway, it is possible to determine a parametric grid size that may produce polygons that meet certain restrictions. For instance, polygons that projected onto the screen will be within specific size bounds: Rockwood[5] and Kumar[6] dynamically and uniformly tessellate rational Bezier surfaces bounding triangle edge lengths in screen space. We pretend to extend these works adding a fixed-frame rate control under triangle edge length quality criteria.

2.2. Predictive fixed-frame rate LOD selection

A predictive fixed-frame-rate LOD selection algorithm estimates the complexity of the frame to be rendered and selects levels of detail to ensure that the update deadline is never exceeded.

Funkhouser and Sequin[7] presents a predictive LOD selection algorithm that adapts the discrete level of detail of visible objects to satisfy a constant frame rate. They used a cost/benefit paradigm that attempted to optimize the perceptual benefit of a frame against the computational cost. Mason and Blake[8] describe a hybrid of Funkhouser and Sequin's predictive LOD selection algorithm and imposters technique. Gobbetti and Bouvier[9]'s predictive LOD selection algorithm works with continuous LOD models. Zach et al.[10] presents a predictive LOD selection algorithm that incorporates discrete and continuous representations of each object.

3. Outline of the algorithm

Given a set of NURBS surfaces, the interactive rendering stage of the proposed tessellation algorithm consist of two main operations:

- culling of invisible surfaces
- fixed frame rate predictive tessellation of visible surfaces

In interactive visualization, the speed of interaction with the 3D model (quantified by the frame rate) has priority over the image quality. One form of controlling the frame rate consists on explicitly bounding the number of polygons which forms tessellations of visible surfaces while maintaining as much quality as possible.

With uniform tessellation it is possible to implement a fixed frame rate predictive algorithm because step sizes computation is made once per surface. So, it allows predicting beforehand with negligible cost a estimation of the number of vertices and polygons that will be generated.

4. Visibility Control

We implement two different culling techniques at surface level: view frustum and back-face culling.

The view frustum culling works by testing the bounding sphere of each surface against the current view-frustum which is defined by the four sides of a truncated pyramid.

The back-face culling algorithm is based on the clustered back-face culling used by Zhang and Hoff III[11] for polygonal models. We extend that idea using a discretization of the normal surface of each NURBS surface. Unitary normal space is partitioned in small pyramids named clusters. The apex of all pyramids is the center of a unitary cube. Each face of the cube is subdivided in regular cells. The base of each pyramid is one of the cells. In preprocess, we characterize each surface based on the clusters that intersect its normal vectors. In each frame, front-facing clusters are determined in constant time based on the representative normals of each cluster (normals of its corners). If the cluster is front-facing, surfaces with normals intersecting this cluster are visible.

5. Tessellation quality

We propose a new tessellation criteria based on the projection into screen space of the bounding sphere projection that encapsulates a surface but extended with geometric information of the surface. With this tessellation criteria, triangles edge length in screen space is bounded.

To obtain the step sizes (distance among two parametric points) in each parametric direction, maximum lengths of the iso-edges of the control polygon in each parametric direction are calculated, (lu_{max}, lv_{max}). The aspect ratio among these lengths ($ar = lu_{max}/lv_{max}$) is constant. If lu_{max} is the diameter of a sphere with center the nearest point of the minimum bounding sphere encapsulating the surface to the point of view, step sizes in each parametric direction (n_u, n_v) are:

$$n_u = \frac{u_{max} - u_{min}}{\frac{lu_{max,scr}}{\lambda} - 1}$$

$$n_v = \frac{v_{max} - v_{min}}{\frac{lu_{max,scr}}{\lambda \cdot ar} - 1}$$

where (u_{max}, u_{min}) and (v_{max}, v_{min}) are the dimensions of the parametric domain in each parametric direction, $lu_{max,scr}$ is the screen space projection of lu_{max} and λ is the triangle edge length bound.

A tessellation of a surface is determined by its $lu_{max,scr}$ and λ. $lu_{max,scr}$ is function of lu_{max}, the center of the minimum bounding sphere encapsulating the surface, the position of the point of view and the position of the projection plane. For a constant value of λ, the number of polygons of a surface tessellation will be reduced or increased if the distance among the point of view and the surface increases ($lu_{max,scr}$ smaller) or decreases ($lu_{max,scr}$ larger).

The value of λ is a global quality bound. So, the effect of changing λ maintaining constant the rest of parameters implies: with increases of λ, step sizes decrease and, therefore, less quality tessellations are generated; inversely, reducing λ involves more quality tessellations. Consequently, if the system detects that the frame rate is degradating or the memory cost is excessive, the logical action will be increasing the value of λ.

From a quality point of view, retessellation of a surface is made only when the following happens:

$$|lu_{max,scr,current} - lu_{max,scr,required}| > K \cdot lu_{max,scr,current}$$

where $K > 0$ controls the frequency of tessellations.

The behavior of K is simple and intuitive. If K is large, the quality difference among two consecutive tessellations will produce the popping effect. On the other hand, if K is small, it is possible that the increasing of quality of the new tessellation do not compensate the effort of generating it.

Independently of the value of K, the frequency of tessellations decreases as the distance among the surface and the point of view decreases. Equally, if the value of K increases, the tessellations are more frequent.

6. Computational Cost of Tessellating and Rendering

If the value of λ is constant during the visualization, the quality of tessellations increases as the distance among the model and the point of view decreases. Consequently it is possible that the amount of polygons exceeds the memory or the rendering capability of the graphics hardware.

The total computational cost of generating and rendering the tessellation of potentially visible surfaces is approximated by:

$$C_{total} = \sum_{i=1}^{n} C(S_i)$$

where N is the set of surfaces and $C(S_i)$ is the cost of generating and rendering the tessellation of a particular surface, S_i. This time is estimated as:

$$C(S_i) = C_{gen}(S_i) + C_{rend}(S_i)$$

The time required to generate a tessellation of a surface depends on its number of vertices and the time required to evaluate a vertex (a vertex is composed of the threedimensional surface point and its associated normal vector): given $lu_{max,scr}$, ar and λ, the temporal cost of generating the tessellation of the surface S_i is:

$$C_{gen}(S_i) = \left(\frac{lu_{max,scr}(S_i)}{\lambda} - 1\right) \cdot \left(\frac{lu_{max,scr}(S_i)}{\lambda \cdot ar(S_i)} - 1\right) \cdot t_{eval}$$

where t_{eval} is the required time to evaluate a vertex with a particular surface evaluation method.

Respect the rendering of surfaces, Funkhouser and Sequin predict the temporal cost of the *Geometry stage* for a polygonal object as a lineal combination of the number or polygons and vertices that compose the object weighted by coefficients that depends on the hardware features and the rendering algorithm (wireframe, Gouraud, Phong,...). For the *Rasterizer stage*, the temporal cost is proportional to the number of pixels covered by the screen projection of the object.

While calculating the number of polygons and vertices is straightforward, determining the number of pixels that covers a projected surface is computationally expensive. It requires calculating silhouette curves and their projection to screen space in runtime. This value can be approximated projecting and calculating the projected area of a bounding volume of the surface. Another possibility is precomputing a set of values of the projected area as function of the surface orientation and obtain the number of pixels as function of the distance.

7. Predictive Fixed-Frame Rate Tessellation

We propose a predictive fixed-frame rate tessellation algorithm based on the parameters exposed in previous sections.

We define $\mathbf{tr_{max}}$ as the time available for the generation of one frame. The user or the application sets the value of this parameter (e.g. 50 msec.). It should be noted that in each frame all the tessellations are rendered to create one image, but only some surfaces require a new tessellation. So, tr_{max} must be distributed between those two tasks, plus the culling phase. However, sometimes might happen that there is not enough time within a frame to render the image and to create all the new tessellations needed. The algorithm predicts the time required to perform the different tasks and gives priority to the generation of a new image per frame. However, this is not an absolute policy, if required, it reserves a minimum time fraction per frame for the generation of tessellations that will reduce the display cost. This policy should lead to less complex tessellations, reduce the time required for the generation of one frame and a new equilibrium. In first place we define some variables then we propose the algorithm:

- $\mathbf{tr_v}$ is the consumed time of the culling operation
- $\mathbf{te_f}$ is the estimated time for the generation of one frame.

This time is the sum of tessellation time and rendering time ($\mathbf{te_f} = \mathbf{te_t} + \mathbf{te_r}$)

- $\mathbf{tr_t}$ is the real time that is used to generate tessellations and rendering them

With respect to parameter λ we use the following notation:

- $\lambda_{\mathbf{obj}}$ is the objetive quality to assign to all surfaces
- λ_{Δ} is the increment of λ that is applied to λ_{obj} when it is necessary to increase or reduce the quality
- $\lambda_{\mathbf{S_i}}$ is the current quality associated to the tessellation of the surface S_i
- $\lambda_{\mathbf{max}}$ is the current largest λ_{S_i}. Its possible values are λ_{obj} or $\lambda_{obj} + \lambda_{\Delta}$

Rendering a frame consists of the following operations:

- Determine which surfaces requires a new tessellation
- Predict the computational cost of tessellating and rendering those surfaces, te_f.
- Determine the consumed time by these two tasks, tr_v

If the available time to generate the frame is larger than the estimated time ($te_f < tr_{max} - tr_v$), the following operations will be done (render + increase quality):

- Tesselate surfaces that require a new tessellation.
- Render.
- If the up to now consumed time is less than tr_{max}, surfaces with $\lambda_{S_i} > \lambda_{obj}$ are retessellated with more quality ($\lambda_{S_i} = \lambda_{obj}$). If the available time is over, this process is interrupted. If the quality of all surfaces is λ_{obj}, this value is decremented in λ_{Δ}: the objetive quality is increased

Otherwise, if the available time to generate the frame is smaller than the estimated time ($te_f > tr_{max} - tr_v$), the following operations will be done (render + increase quality + allow for lower frame rates):

- If $\lambda_{max} = \lambda_{obj}$, the objetive quality is reduced in λ_{Δ}.
- Decide how much time will be dedicated to generate coarser tessellations, tr_t. This election is made as function of the estimated time of rendering, te_r. The following expression is suggested:[†]

$$te_r > tr_{max} : tr_t = tr_{max}/2$$
$$te_r \leq tr_{max} : tr_t = tr_{max} - te_r * 0.5$$

- During this time, tr_t, surfaces with $\lambda_{S_i} < \lambda_{obj}$ are retessellated reducing its quality ($\lambda_{S_i} = \lambda_{obj}$). If the available time is over, this process is interrupted and the rendering phase begins. This simplification will continue in the next frame.
- If the previous operation has been completed and the available time is not over, surfaces that require a new tessellation will be retessellated.
- Render.

8. Conclusions

We propose a predictive fixed-frame rate scheme that shows how to control the visualization frame rate using uniform dynamic tessellation. We are researching about this tessellation algorithm now. In the future it must be extended to avoid cracks among adjacent surfaces and to support trimmed surfaces.

References

1. D. Borro, I. Recio, H. Sánchez, A. García-Alonso and L. Matey. CSCW for foundry design using Java3D. *ACM 2000 Conference on Computer Supported Cooperative Work*, Philadelphia, December 2000. 1
2. D. Borro, I. Recio, C. Pedrinaci, H. Sánchez and A. García-Alonso. Peer-to-peer techniques applied to the Cooperative Visualization of CAD Models. *Proceedings of MICAD 2003*, Paris, pp. 8–13, April 2003. 1
3. L.A. Piegl and W. Tiller. The NURBS Book. Monographs in Visual Communication. Springer, 2° edition, 1997. 1
4. J. Chhuganni and S. Kumar. Budget Based Sampling of Parametric Surfaces. *To appear in ACM 3D Interactive Graphics*, 2003. 1
5. A. Rockwood, K. Heaton and T. Davis. Realtime Rendering of Trimmed Surfaces. *ACM Computer Graphics (SIGGRAPH Proceedings)*, **23**(3), pp. 107–117, 1989. 1, 2
6. S. Kumar. Interactive Display of Parametric Spline Surfaces. *PhD Thesis*, University of North Carolina, 1996. 1, 2
7. T.A. Funkhouser and C.H. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. *Computer Graphics (SIGGRAPH 93 Proceedings)*, **27**, pp. 247–256, August 1993. 2
8. A. Mason and E.H. Blake. Automatic Hierarchical Level of Detail Optimization in Computer Animation. *Computer Graphics Forum*, **16**(3), pp. 191–199, 1997. 2
9. E. Gobbetti y E. Bouvier. Time-critical multiresolution scene rendering. *IEEE Visualization*, pp. 123–130, 1999. 2
10. C. Zach, S. Mantler and K. Karner. Time-critical Rendering of Discrete and Continuous Levels of Detail. *Eurographics Workshop on Rendering*, pp. 1–8, 2002. 2
11. H. Zhang and K.E. Hoff III. Fast Backface Culling Using Normal Mask. *Symposium on Interactive 3D Graphics*, pp. 103–106, 1997. 2

[†] Although in this work, a simple scheme is suggested, this election is not obvious: a iterative-predictive loop that estimates what surfaces should be tessellated to maximize the rendering time saving could be use.