Adaptive Tessellation of Bézier Surfaces Based on Displacement Maps

F.J. Espino[1] M. Bóo[1] M. Amor[2] J.D. Bruguera[1]

[1]Dept. of Electronic and Computer Eng., Univ. of Santiago de Compostela,
E-mail: {javi, mboo, bruguera}@dec.usc.es

[2]Dept. of Electronics and Systems, Univ. of A Coruña
E-mail: margamor@udc.es

ABSTRACT

Bézier surfaces are widely used in computer graphics applications. Rendering of such surfaces is commonly performed by tessellation. In order to generate less triangles for high quality surfaces, adaptive tessellation algorithms are better. The geometric tests used by these algorithms perform vector computations of high latency that decreases the performance of the algorithm. We propose an adaptive tessellation algorithm that avoids vector computations for tests, replacing them with scalar computations. This way, latency of tests is reduced and therefore, performance improved.

Keywords: Bézier surfaces, tessellation, subdivision, displacement map

1 INTRODUCTION

Bézier surfaces are used in several computer graphics applications. Due to the excellent performance of hardware based methods for triangle meshes, the strategy mostly employed for rendering is the surface tessellation [Kumar96, Moret01]. There are different methods and hardware proposals [Kumar96, Moret01] for Bézier surfaces tessellation. Adaptive tessellation methods increase detail where required [Chung00, Espin03]. However, redundant computation of vertices is made, because every shared vertex is computed once per triangle. The method proposed in [Espin04] improves performance processing groups of triangles instead of individual triangles. The tests employed for adaptive tessellation are based on the computation of all candidate vertices and normal vectors to be inserted and their posterior analysis. This implies much computation, because vertices that do not need to be inserted are computed.

We propose an adaptive tessellation algorithm based on the *layer strips* method [Espin04]. Edge tests are performed using scalar computations over a displacement map [Lee00]. The displacement map is sent together with a coarse tessellation to the the graphics unit. During subdivision, no vertex is computed before a decision is made, improving performance. As a result, good quality meshes are generated with less computations.

WSCG SHORT papers proceedings, ISBN 80-903100-9-5
WSCG'2005, January 31-February 4,2005
Plzen, Czech Republic.

2 ADAPTIVE TESSELLATION BASED ON DISPLACEMENT MAPS

Adaptive tessellation algorithms analyze geometric information of the surface in order to subdivide with high detail the regions of high curvature. This way, the resulting meshes have high quality with less triangles than uniform tes-

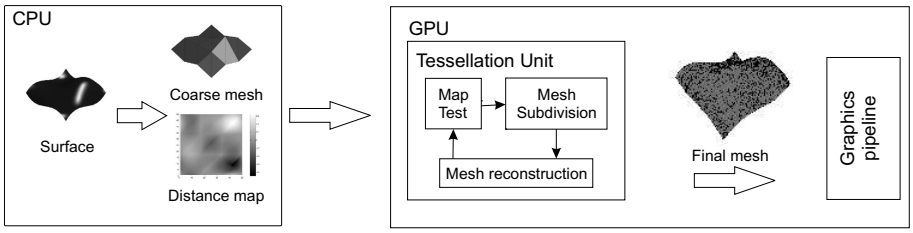


Figure 1: Diagram of the adaptive tessellation algorithm

sellations. The simplest algorithms [Chung00, Espin03] generate an initial coarse tessellation and then processes each triangle iteratively, analyzing each edge for deciding the subdivision pattern of the triangle. Several tests can be used, obtaining different results in number of triangles and quality of the tessellation. The problem of processing triangles individually is that vertices inserted in edges are re-computed. Moreover, tests need the coordinates and normal vector of the vertex that is going to be inserted, so useless computation is performed.

In this section we propose an algorithm that avoids unnecessary vertex computations. The coarse tessellation and its displacement map are computed and sent for rendering. The map is a matrix of distances between the initial coarse tessellation and the surface. This information is employed to generate the adaptive subdivided mesh. The resulting triangle mesh is rendered in the graphics system. This algorithm is a good candidate to be implemented as a first stage of the Graphics Unit (GPU) as shown in Figure 1. This way, communications between CPU and GPU are kept low. Next, the algorithm is detailed.

2.1 Displacement Map Generation

Generation of the displacement map is made through the following steps. First, the surface is sampled in $M \times N$ points and, for each point, a distance value is computed. In general, the distance is computed as the distance between the surface point and the plane of the triangle with a normal vector intersecting the point. In Figure 2 a 2D simplification of a surface and a coarse tessellation is shown. Surface point S_1 represents the general case, where the point is over the normal of only one triangle T_1 and distance d_1 is computed.

Exceptional cases can be found. For these cases we propose the following. In Figure 2, point S_2

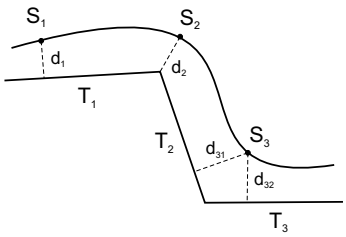


Figure 2: Cases in distance map generation

has no triangle with a normal vector intersecting it, so the distance is computed as the distance d_2 to the nearest edge (edge between triangles T_1 and T_2). Finally, both triangles T_2 and T_3 have normal vectors that intersect point S_3, so the nearest triangle is chosen for computing distance d_3.

The necessary resolution ($M \times N$) of maps can be computed through a Fourier analysis, using Nyquist criteria to select the minimum map resolution. However, the simulations of our algorithm show that lower resolutions do not decrease the quality of the generated meshes. This is due to the fact that maps are not used for surface reconstruction, but only for test purposes.

2.2 Distance Map Test

The edge analysis is performed comparing distances associated to vertices and inserting vertices in those regions where the distance differences are greater than a threshold value. Parametric coordinates are used to read the map values. Due to the limited resolution of the map, the distances are linearly interpolated from the available distances.

Three tests are distinguished depending on how many points are used for distance comparisons: two-point test ($Map2p$), four-point test ($Map4p$) and eight-point test($Map8p$). Figure 3 represents the data used for these tests: d_1 and d_2 are the

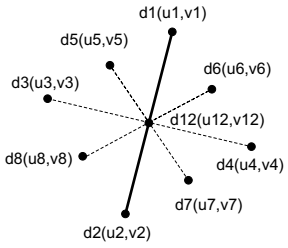


Figure 3: Distance values for additional test

Figure	T	cmesh(KB)	map(KB)	Error
Surface1	18	0.84	1.00	0.2566
Surface2	18	0.84	1.00	0.1604
Teacup	468	21.94	26.00	0.0273

Table 1: Initial tessellations

distances associated to the vertices of the edge; d_{12}, the distance of the candidate vertex to be inserted if the test is positive; and d_3 to d_8, the additional points for tests $Map4p$ and $Map8p$. The parameters u and v are the parametric coordinates of the vertices, used for accessing the distance map.

The test $Map2p$ only uses edge related information, i.e. points d_1, d_2 and the tested vertex d_{12}. Differences are computed and compared to a threshold value according to:

$$Map2p = (|d_1 - d_{12}| > th)\ OR\ (|d_2 - d_{12}| > th) \quad (1)$$

The $Map4p$ test uses two additional distance values (d_3 and d_4 in Figure 3) to improve the quality of the tessellation. The parametric coordinates needed for accessing the map are computed with:

$$\begin{array}{l}(u_3, v_3) = (u_{12} - v_e,\ v_{12} + u_e) \\ (u_4, v_4) = (u_{12} + v_e,\ v_{12} - u_e)\end{array} \quad (2)$$

where $(u_e, v_e) = (u_1 - u_{12}, v_1 - v_{12})$. The differences are compared together with differences of $Map2p$ to the threshold value using:

$$\begin{array}{rcl} Map4p & = & Map2p\ OR\ (|d_3 - d_{12}| > th) \\ & & OR\ (|d_4 - d_{12}| > th) \end{array} \quad (3)$$

where $Map2p$ are the comparisons of Eq. (1). This test is extended to $Map8p$ where four additional distances (d_5, d_6, d_7 and d_8) are considered. These tests have been selected due to their simplicity of computation and the good results obtained during simulation.

3 RESULTS

The algorithm has been simulated in C++ language and tested for several figures. Results are shown for three figures: two individual surfaces and a teacup.

Table 1 shows the memory requirements and mesh error of the coarse tessellations for the three figures. Column *cmesh* shows the memory requirement of the coarse mesh (18 triangles per surface[1] where *teacup* has 26 surfaces). Column *map* shows the requirements of the displacement map with a resolution of 16x16 points. Keeping these sizes, the communications requirements are kept very low (1.84KB in total). The last column of Table 1 shows the mesh error of the coarse meshes.

Table 2 shows the mesh error for the subdivided meshes with map tests (columns $Map2p$, $Map4p$ and $Map8p$). The error of flat test [Espin04] is also included in column $flat$. Results for different number of triangles of final mesh (columns T) are shown. Of course, any of the adaptive methods reduces the error respect to the coarse mesh.

Simulation data shows that using more distances in test improves quality of subdivided meshes. The difference in quality between $Map4p$ and $Map8p$ is small and the cost, larger. Therefore, $Map4p$ have the best quality and cost trade-off.

Quality of meshes subdivided with map tests are very close to those generated with flat test. For example, for $surface2$ tessellated with around 5000 triangles quality is 0.00164 and 0.00064 for $Map4p$ and $flat$ respectively; quality is even better for more complex figures, like *teacup*.

Final considerations about mesh quality can be made by looking at Figure 4. Figures 4(a), 4(b) and 4(c) shows the subdivided meshes using the $Map4p$ test with approximately 5000 triangles. As mentioned above, quality of the coarse tessellation is improved with higher detail in curved regions.

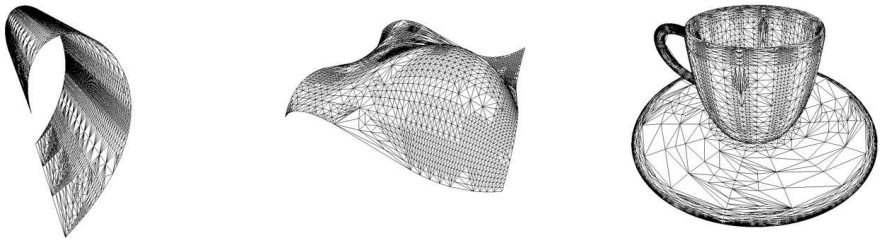
With the flat tests [Espin04] when a vertex is computed and not inserted, useless computations are performed. In our proposal the decisions are computed first, so useless computations are avoided and performance is improved.

In summary, we propose to reduce the computations associated to vertices not finally inserted using high quality subdivided meshes and low communication and memory requirements.

[1] Layer Strip [Espin04] requires 36 Bytes per vertex.

Figure	Map2p		Map4p		Map8p		Flat	
	T	E	T	E	T	E	T	E
surface1	5034	0.00025	5846	0.00167	5848	0.00168	4374	0.00025
	4388	0.00047	4695	0.00182	4697	0.00183	3600	0.00053
	2070	0.00160	2037	0.00438	2043	0.00437	1926	0.00132
surface2	5831	0.00344	6010	0.00162	5898	0.00164	5645	0.00064
	4384	0.00213	4045	0.00212	4306	0.00168	4003	0.00092
	2041	0.00404	2093	0.00403	2009	0.00375	2074	0.00183
teacup	20100	0.00175	23196	0.00145	17987	0.00238	20604	0.00242
	13380	0.00239	14859	0.00297	14589	0.00318	12220	0.00363
	7835	0.00429	7284	0.00406	7207	0.00427	9630	0.00390

Table 2: Tessellation results



(a) Surface1 (Map4p test)

(b) Surface2 (Map4p test)

(c) Teacup (Map4p test)

Figure 4: Tessellated meshes

4 CONCLUSIONS

In this paper an adaptive tessellation algorithm for Bézier surfaces is presented. It combines the methods of previous algorithms with the displacement map concept in order to obtain more efficient and implementable tests.

The method we propose is based on displacement maps and performs simpler tests that do not need the precomputation of vertices, increasing performance. Quality results for subdivided meshes are very close to previous algorithms, keeping communication requirements low.

The different tests based on using a set of distances (from 2 to 8) have been analyzed and compared, concluding that the use of 4 distances for test has the best trade off between quality and computation load.

ACKNOWLEGMENTS

This work was partially supported by the Ministry of Science and Technology of Spain under contract MCYT-FEDER TIC2001-3694-C02-01 and by the Secretaría Xeral I+D of Galicia (Spain) under contract PGIDIT03TIC10502PR.

REFERENCES

[Chung00] A.J. Chung and A.J. Field. A Simple Recursive Tessellator for Adaptive Surface Triangulation. *Journal of Graphics Tools: JGT*, 5(3):1–9, 2000.

[Espin03] F. J. Espino, M. Bóo, M. Amor, and J. D. Bruguera. Adaptive Tessellation of NURBS Surfaces. *Journal of WSCG*, 11(1):133–140, 2003.

[Espin04] F. J. Espino, M. Bóo, M. Amor, and J. D. Bruguera. Hardware Support for Adaptive Tessellation of Bézier Surfaces Based on Local Tests. Technical report, www.ac.usc.es, 2004.

[Kumar96] S. Kumar, D. Manocha, and A. Lastra. Interactive Display of Large-Scaled NURBS Models. *IEEE Trans. on Vis. and Comp. Graphics*, 2(4):323–336, 1996.

[Lee00] Aaron Lee, Henry Moreton, and Hugues Hoppe. Displaced Subdivision Surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 85–94, 2000.

[Moret01] H. Moreton. Watertight Tessellation Using Forward Differencing. In *ACM Siggraph/Eurographics Workshop on Graphics Hardware*, pages 25–32, 2001.