Per-Pixel Evaluation of Parametric Surfaces on GPU

Takashi Kanai
Keio University SFC
kanai@sfc.keio.ac.jp

Yusuke Yasui
ASTOM Inc.
yasui@astom.co.jp

http://graphics.sfc.keio.ac.jp/project/gpusubd/

1 Introduction

In the rendering of parametric surfaces, in general, we often use a polygon to draw the approximation of its original surface. However, such rendering of a polygon does not represent its original surface exactly. This occurs notably in the case that a polygon is coarse or that a view position becomes closer to a surface.

In this paper, we propose a per-pixel evaluation method of parametric surfaces using GPU which is effective for rendering. For the evaluation of surfaces on GPU, [Bolz and Schröder 2003] have proposed a method for rendering subdivision surfaces. Unfortunately, the final rendering is done by subdivided polygons in this approach. The characteristic of our scheme is to perform the evaluation in the fragment program from parameters of each fragment. This ensures a (at least pixel-based) high quality surface rendering. We also introduce an example of using our per-pixel surface rendering scheme.

2 Overview

We describe here an overview of our per-pixel evaluation of paramtric surfaces. We prepare a polygon as input. This polygon can be created with approximating the original surface. In addition, we also prepare a face id in each face and parameters (u,v) in each vertex. Each face of the input polygon is decomposed into the set of fragments in the rasterization process. Each fragment has an inherited face id and linearly interpolated parameters u and v from the input polygon. These are once stored in the texture, and are used in the fragment program to compute positions or their derivatives.

Positions or their derivatives are computed by drawing a rectangle polygon on p-buffer. The viewport size of p-buffer is the same as the transferred texture. Positions are computed in the fragment program as follows. Firstly, we fetch the parameters and face id from the transferred texture at the current fragment. By using them, we select a set of control points from the texture stored in advance. Also, we compute the basis functions with these parameters. Then, the position is computed by linear combination of the basis function with control points. The result is once copied to a vertex array. When rendering, it is also transferred as the texture to another fragment program which computes the derivatives and normals.

A method for computing basis functions and for storing control points are different by various forms of parametric surfaces. We describe each evaluation method for two types of parametric surfaces.

Bézier, B-spline, Hermite Surfaces. These linear parametric forms are in general represented as the following formula:

$$s(u,v) = \sum_i b_i^n(u,v) C_i, \qquad (1)$$

where C_i, $b_i(u,v)$, n denote a control point, a basis function and the degree, respectively. In this case, it is effective that control points are stored in a texture, and coefficients of basis functions are directly provided in a fragment program. Furthermore, the computation of basis functions becomes simpler when degrees are fixed. It is also possible to evaluate non-linear parametric forms of surfaces such as NURBS as linear cases.

Subdivision Surfaces. Stam proposed in [Stam 1998] that the evaluation of subdivision surfaces at arbitrary parameter value is regarded as the linear combination of eigenbasis functions and control points. A point on a surface $s(u,v)$ is represented as follows:

$$s(u,v) = \hat{C}_0^T \Lambda^{n-1} X_k b(t_{k,n}(u,v)). \qquad (2)$$

Each position is evaluated on Ω_k^n, where n and k are determined by the two parameters u and v. $\hat{C}_0$ in Equation (2) is represented as $\hat{C}_0 = V^{-1} C_0$, where C_0 is a column vector representing the set of control points around a rectangle. N denotes the valence of an extraordinary point. V is an invertible matrix whose columns are the eigenvectors of subdivision matrix. Λ is a diagonal matrix containing the eigenvalues of the subdivision matrix.

Using Equation (2) directly is too costly even on the current graphics hardware. Consequently, we calculate $\hat{C}_0^T \Lambda^{n-1} X_k$ in advance and store it to the texture instead of storing each $\hat{C}_0^T, \Lambda^{n-1}$ and X_k, respectively.

3 Results and Conclusion

Frame rates of our scheme are approximately 2-8 fps (measured on Athlon 2600^+ + Geforce FX 5900 Ultra). The computation for each frame involves a position of subdivision surface, two first order derivatives in both u, v directions and a normal. This depends on the resolution of display, because the number of processed fragments can be changed according to such resolutions. As one of applications for our per-pixel evaluation method, [Yasui and Kanai 2004] introduces a high quality surface assessment method of subdivision surfaces in combined with the computation of reflection lines.

References

BOLZ, J., AND SCHRÖDER, P. 2003. Evaluation of subdivision surfaces on programmable graphics hardware. submitted for publication.

STAM, J. 1998. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In *Computer Graphics (Proc. SIGGRAPH 98)*, ACM Press, New York, 395–404.

YASUI, Y., AND KANAI, T. 2004. Surface quality assessment of subdivision surfaces on programmable graphics hardware. In *International Conference on Shape Modeling and Applications 2004*, IEEE CS Press, Los Alamitos, CA, 129–136.