5Y-9

GPU による Bezier 曲面の適応的表示

野村　晃†　　吉田　典正†

日本大学†

1. はじめに

CAD ではパラメトリック曲面が広く用いられ，扱われるモデルは非常に複雑なものとなってきている．複雑なモデルを高速に描画するためにはグラフィックハードウェアを効率的に利用すること，ソフトウェアによる工夫によりハードウェアに転送する基本図形の数を減らすことの 2 点が考えられる．

本報告では，上記の 2 点を利用し，GPU(Graphics Processing Unit) を用い，LOD(Level of Detail)と呼ばれる物体の視覚的な重要度に応じてモデルの複雑さを変化させる手法について述べる．

ポリゴンに関する LOD 制御は今まで多くの研究がなされている [2]．パラメトリック曲面を GPU によって効率的かつ高品質に表示する研究も行われているが [1]，本研究では視点からの距離や境界であるかどうかによって適応的に表示する手法を対象とする．

2. GPU を用いた Bezier 曲面の計算

ここでは，GPU 上で双 3 次 Bezier 曲面を表示する方法について述べる．双 3 次 Bezier 曲面式を式(2.1)に，パラメータ u,v による微分ベクトルを式(2.2)に示す．

$$S(u,v)=\sum_{I=0}^{3}\sum_{j=0}^{3}P_{ij}B_i^3(u)B_j^3(v) \qquad (2.1)$$

$$d_u=\frac{\partial}{\partial u}S(u,v) \qquad d_v=\frac{\partial}{\partial u}S(u,v) \qquad (2.2)$$

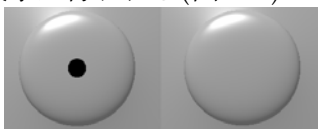
GPU 上での Bezier 曲面の生成の具体的な流れを次に示す．

① CPU から GPU に図 3.3 に示す uv 空間の四辺形群を送る（実際には Display List として保持する）．

② VertexShader で Bezier 曲面上の点および u,v に関する微分ベクトル d_u,d_v を求める．

③ FragmentShader で d_u と d_v の外積を計算し，曲面の法線を算出し，Shading 計算を行う．

この際，法線の計算を VertexShader ではなく FragmentShader で行うことにより，四角形パッチの 1 辺が頂点に縮退する場合に，縮退した点の近傍でより正確に法線を計算することができ，より正しく表示が行われる(図 2.1)．



(a)　　(b)

図 2.1　Teapot の上蓋．(a) VertexShader で法線を求めた場合．(b) FragmentShader で法線を求めた場合．

3. 曲面の LOD

ここでは，物体の視覚的な重要度によって複雑さを変化させる LOD 制御について述べる．

本報告では，視界の後方を向いている曲面パッチを効率的に除去するために法線円錐（Cone of Normals)[3] と呼ばれる手法を用いる．

3.1. 法線円錐と後方曲面パッチの判定

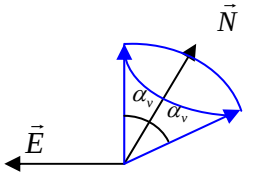
法線円錐は，曲面パッチ上の法線ベクトルの集合を円錐の頂点と角度によって表現する手法である（図 3.1）．また，図 3.2 は曲面パッチに法線円錐を表示した例である．法線円錐を利用した後方面判定のアルゴリズムを次に示す．

① Bezier 曲面の法線 $\vec{n}$ の平均 $\vec{N}$ を求める．

② 曲面パッチ上の様々な法線 $\vec{n}$ に関して，$\vec{N}$ と $\vec{n}$ のなす角の最大値 α_v を求める．

③ 視点 e から Beizer 曲面パッチの四隅の点 $a_{v1}\cdots a_{v4}$ へ向かうベクトルを $\vec{E}_1\cdots\vec{E}_4$ とするときとするとき，式(3.1)がすべての i について成り立つのであれば，曲面パッチは後方を向いているので描画しない．

$$\vec{N}\cdot\vec{E}_i<\sin\alpha_v \quad (i=1\ldots4) \qquad (3.1)$$



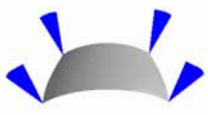


図 3.1　頂点円錐の計算　図 3.2　曲面の四隅の法線円錐

Adaptive Display of Bezier Surfaces using a GPU
† Hikaru Nomura, Norimasa Yoshida, Nihon University

3.2. 境界面の判定

境界面（シルエットを含む面）はオブジェクトの視覚的な重要度が高いため他の面より詳細に描画する必要がある（ただし，反射線などによる曲面の評価を行う場合には，境界面でない前方面も詳細にする必要がある）．そのため，法線円錐を用いて，境界面の判定は 3.1 で求めた法線の平均 $\vec{N}$ と視点から曲面の四隅へのベクトル $\vec{E}_1 \cdots \vec{E}_4$ を用い，式(3.2)がいずれかの i について成り立つとき曲面パッチはシルエットを含む可能性がある．シルエットを含む曲面パッチは，より詳細化する．

$$|\vec{N} \cdot \vec{E}_i| < \sin\alpha_v \ (i = 1 \ldots 4) \qquad (3.2)$$

3.3. 視点からの距離に応じた詳細化

視点からの距離に応じて描画するオブジェクトの詳細度を変化させる．詳細度は境界線の見え方を考慮して，$(0,1)\times(0,1)$ の uv 空間を $5^2, 10^2, 15^2, 20^2$ のように分割した（図 3.3）．

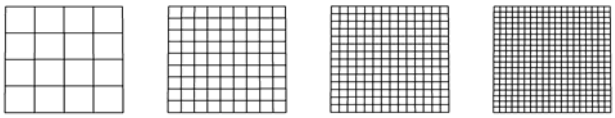


図 3.3 uv パラメータ空間における詳細度の変化

4. 実行結果

CPU には Pentium4 3GHz，GPU には GeForce7800GTX の環境で，表示される結果および効率の度合いを比較した．ただし，効率の測定結果は CPU および GPU の性能のバランスにより変化するため，本結果は一例である．また，Bezier 曲面には，Utah Teapot を利用し，De Casteljau のアルゴリズムにより曲面パッチを分割し，パッチ数を増やした．

表 4.1 に GPU を用いた場合および CPU のみの場合の描画速度の比較を示す．この結果から，曲面パッチ数が多いほど．GPU を用いた場合のほうが CPU のみで描画を行った場合と比べて速度が向上した．

表 4.1 CPU のみで描画を行った場合と GPU を利用した場合の Teapot の表示時間．

	32patch	64patch	128patch	256patch	512patch
CPU (ms)	17.11	25.77	37.64	59.95	100.5
GPU (ms)	14.56	17.73	23.48	35.53	56.81
上昇率(%)	14.9036	31.1991	37.6196	40.7339	43.4726

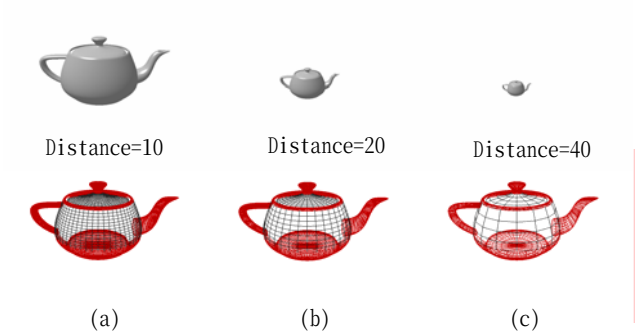
図 4.1 のように GPU を用いた場合のほうが同じ分割数でも法線がより正確に補間されるためハイライトがより正確に描画ができる．

図 4.2 は LOD 制御を行った例である．この結果から，距離が遠くなるほど曲面パッチが粗くなっている様子が分かる．

図 4.3 に反射線を表示した例を示す．(a)は本手法で決めた詳細度で反射線を表示したものであり，(b)は詳細度を低くしたものである．
この結果から，反射線が正確に表示されていることが分かる．

(a) (b)

図 4.1 (a) CPU のみで描画を行った場合，(b) GPU を用いた場合．



(a) (b) (c)

図 4.2 LOD 制御を行った場合の Teapot．赤い線の部分が詳細化された面

(a) (b)

図 4.3 反射線表示(a)通常例，(b)粗くした例

5. おわりに

本報告では，GPU および法線円錐を用いることによって，双三次の Bezier 曲面を，視点からの距離や境界面であるかに応じて適応的に表示する手法を述べた．本手法を用いることで，モデルが大規模であるほど CPU のみで描画を行う場合と比べて高速であり，常に高精度な描画ができることが確認できた．なお，本報告では Bezier 曲面を用いたが，NURBS など他のパラメトリック曲面にも適用することができる．

参考文献

1) 金井崇: GPU による非一様 B スプライン曲面の高速かつ高品質な表示手法．NICOGRAPH/MULTIMEDIA, 2006.
2) D. Luebke et al.: Level of Detail for 3D Graphics, Morgan Kaufmann, 2003.
3) L. Shirman et al.: The cone of normals technique for fast processing of curved patches, Eurographics, 261–272, 1993.