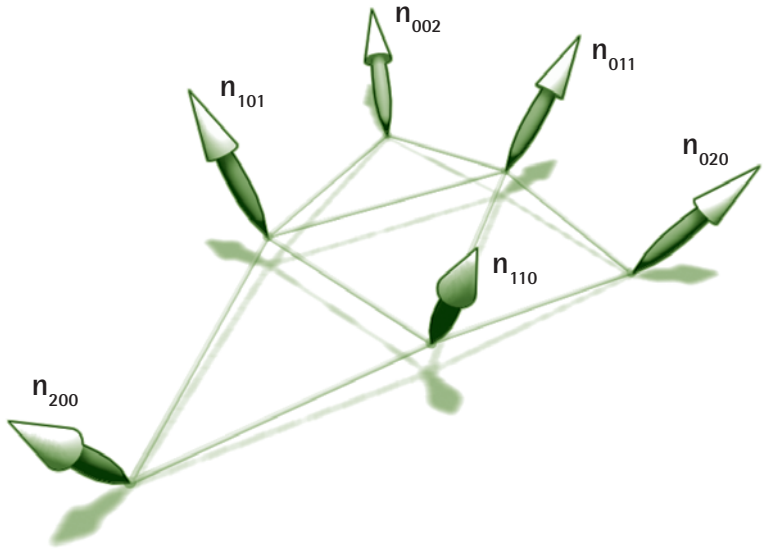
Recent developments for PN-Triangles

Seminar in Geometric Modelling – WS 2006/2007



Recent developments for PN-Triangles

Contents

Recent developments for PN-Triangles

1. Introduction in PN-Triangles

In 3D entertainment software design:

· New techniques added to hardware must have a simple migration path
· A majority of computer software use triangles as their fundamental modelling primitiv
· It is increasingly common for applications to pass normals to the hardware

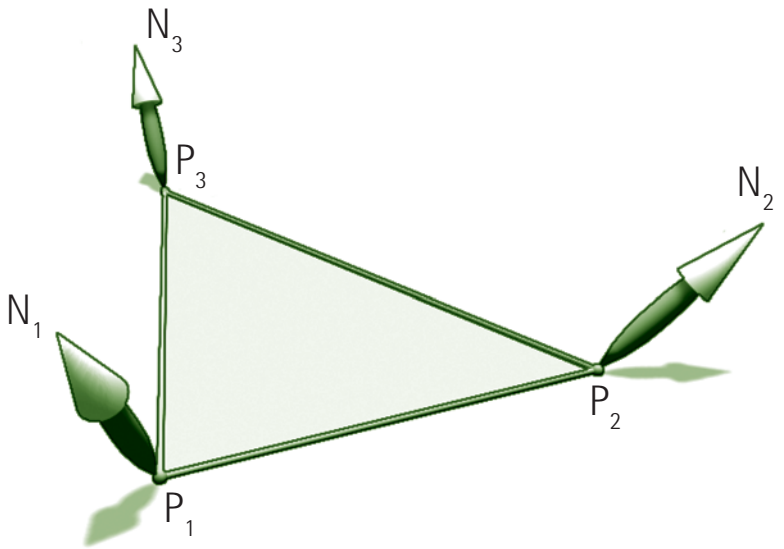
Triangles and normals are the basic components for Point-normal Triangles, short PN-Triangles. Because of the arguments mentioned above, PN-Triangles have become a good way for improving visual quality.

Recent developments for PN-Triangles

1.1 Introduction in PN-Triangles

Standard calculation of PN-Triangles (in Software):

· Input data are points P_i of a given Triangle and the coresponding normals N_i
· Seperate calculation of the geometry (control points) and the normal components of the curved PN-Triangle



Recent developments for PN-Triangles

1.2 Calculation of the Geometry

The geometry of a curved PN-Triangle is defined by a cubic patch b

$b: R^2 \rightarrow R^3$, for $w = 1 - u - v, \quad u,v,w \geq 0$

$$b(u,v) = b_{300}w^3 + b_{030}u^3 + b_{003}v^3 + b_{210}3w^2u + b_{120}3wu^2 + b_{201}3w^2v + b_{021}3u^2v + b_{102}3wv^2 + b_{102}3wv^2 + b_{012}3uv^2 + b_{111}6wuv$$

The control points b_{ijk} are defined as follows:

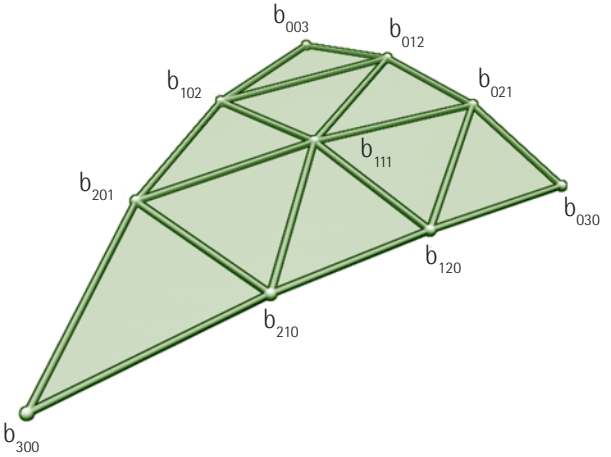
$b_{300} = P_1$
$b_{030} = P_2$
$b_{003} = P_3$

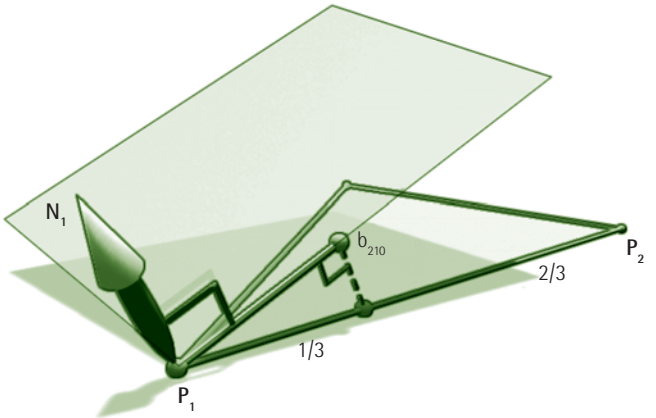
$w_{ij} = (P_j - P_i) \cdot N_i$ (here $\cdot$ is the scalar product)

$b_{210} = (2P_1 + P_2 - w_{12}N_1) / 3$ $\quad$ $b_{120} = (2P_2 + P_1 - w_{21}N_2) / 3$
$b_{021} = (2P_2 + P_3 - w_{23}N_2) / 3$ $\quad$ $b_{012} = (2P_3 + P_2 - w_{32}N_3) / 3$
$b_{201} = (2P_3 + P_1 - w_{31}N_3) / 3$ $\quad$ $b_{102} = (2P_1 + P_3 - w_{13}N_1) / 3$

$E = (b_{210} + b_{120} + b_{021} + b_{012} + b_{201} + b_{102}) / 6$
$V = (P_1 + P_2 + P_3) / 3$

$b_{111} = E + (E - V) / 2$





Recent developments for PN-Triangles

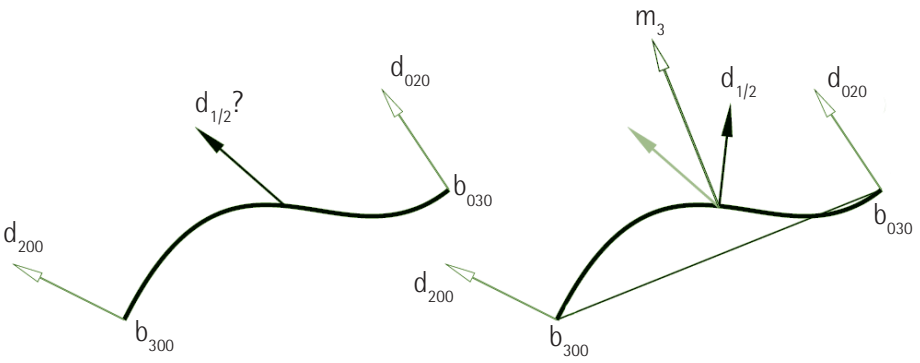
1.3 Quadratically varying normals

Why choose cubically varying geometry and quadratically varying normals?

- The exact surface normals are a challenge to compute
- Tangent continuity does not imply overall better visual impression (smooth interpolating schemes often generate extraneous creases)
- A least cost surface with one polynomial piece per triangle geometric shape is not necessarily normal-continuous, but improves the shading monotonicity by prescribing a separate, continuous normal.

How to calculate the normals:

- Linear interpolation of the normals at the endpoints ignores inflections in the curve while the quadratic normal construction of curved PN triangles picks up such shape variations.



Recent developments for PN-Triangles

1.4 Calculation of the normals

The normals of a curved PN-Triangle are defined as follows:

$n: R^2 \rightarrow R^3$, for $w = 1 - u - v, \quad u,v,w \geq 0$

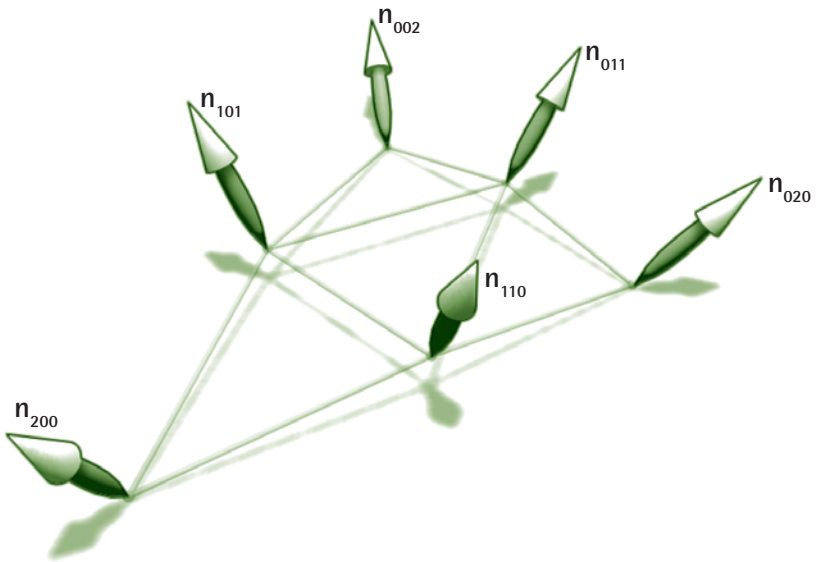
$n(u,v) = n_{200}w^2 + n_{020}u^2 + n_{002}v^2 + n_{110}wu + n_{011}uv + n_{101}wv$

The control points for the normal component are defined as follows for $||N_i|| = 1$:

$n_{200} = N_1$
$n_{020} = N_2$
$n_{002} = N_3$

$$v_{ij} = 2\frac{(P_j - P_i) \cdot (N_i + N_j)}{(P_j - P_i) \cdot (P_j - P_i)}$$

$n_{110} = h_{110} / ||h_{110}||,$ $\quad h_{110} = N_1 + N_2 - v_{12}(P_2 - P_1)$
$n_{011} = h_{011} / ||h_{011}||,$ $\quad h_{011} = N_2 + N_3 - v_{23}(P_3 - P_2)$
$n_{101} = h_{101} / ||h_{101}||,$ $\quad h_{101} = N_3 + N_1 - v_{31}(P_1 - P_3)$



Recent developments for PN-Triangles

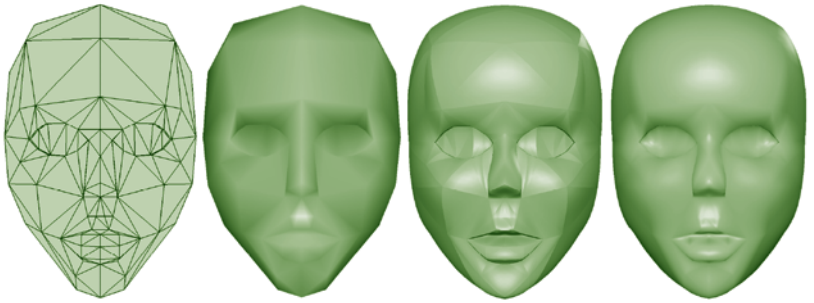
1.5 Pro and Contra of PN-Triangles

Pro:

· Provide a smoother silhouette and a more organic shape for triangle-based models
· Can be used as an triangle-multiplier and can therefore be viewed as a form of geometry compression
· Minimal cost for model preparation, application program modification and rendering performance

Contra:

· Standard algorithm for PN-Triangles smooth around highlights and sharp- /or crease edges of the original model.



Recent developments for PN-Triangles

1.6 Curved Sharp Edges

Curved sharp edges with different normals associated with each side of the (logical) edge between two triangles will in general result in gaps in the surface.

Possible Solution:

- Increase the number of indices in the input stream from three to (at most) nine for triangles with up to six crease halfedges.
- Additional indices point to the point-normal pair of the abutting triangle in the neighboring triangulation
- The algorithm for determining the PN triangle geometry is unchanged except that it projects onto a tangent line to obtain the tangent coefficient of an edge

Recent developments for PN-Triangles

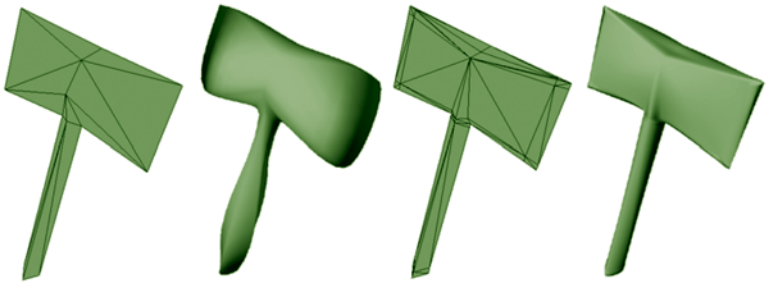
1.7 Problems

Currently a software preprocessing step adds a rim of small triangles along edges intended to be sharp.

This will result in:

· Higher amount of triangles

· Possible singularities in the added triangles

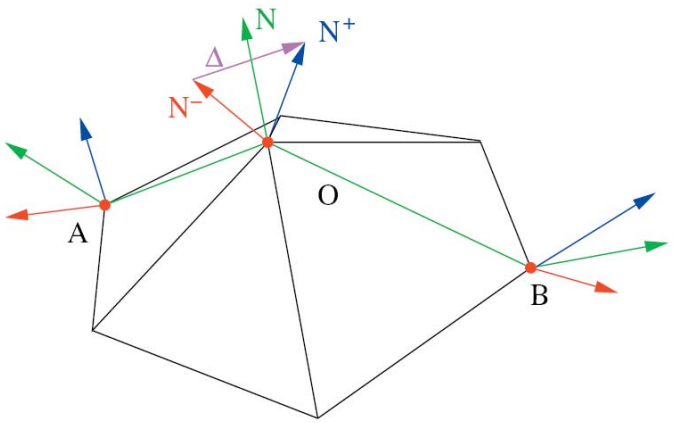


Recent developments for PN-Triangles

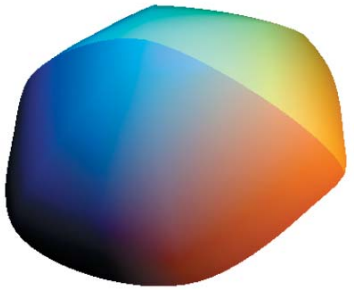
2. Scalar Tagged PN-Triangles

A crease passes through the vertex O
An average normal vector N^+ and N^- can be computed
The sharp crease is then implicitly defined by the three tagged vertices A, O and B

$N = N^+ + N^-$ (normalized to unit length)
$\Delta = N^+ - N^-$ (So $\Delta = 0$ correspondsto a smooth vertex.)
The word „tagged" specifies a vertex with a non-null Δ



(a)



(b)

Recent developments for PN-Triangles

2.1 Basics on STPN-Triangles

Two rules:

1. A tagged vertex can have 2 tagged neighbors at most.
   Ensures that only one crease passes through a given vertex. If not, more than one vector Δ is needed to encode the normal discontinuity

2. A triangle can have 2 tagged vertices at most.
   Does make unambiguous the difference between two distinct creases that are separated by only one triangle, and a crease that loops around a single triangle

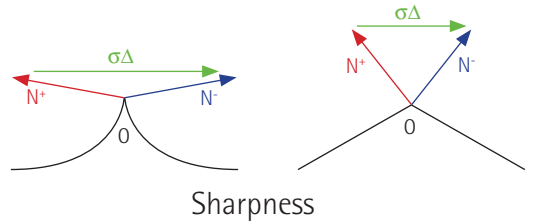
Recent developments for PN-Triangles

2.2 Shape Parameters

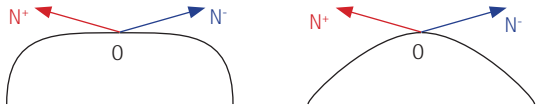
1. $\sigma \in [0;1[$ is called sharpness: it defines the divergence of normal vectors across the two sides of the crease, by interpolating between totally smooth ($\sigma = 0$) and totally sharp ($\sigma = 1$) congurations

2. $\theta \in [-1;1]$ is called tension: it is used to locally control the curvature of all Bézier boundary curves that are starting from a given tagged vertex, and allows to interpolate between three different congurations: tensed Bézier ($\theta > 0$), standard Bézier ($\theta = 0$) and relaxed Bézier ($\theta < 0$).

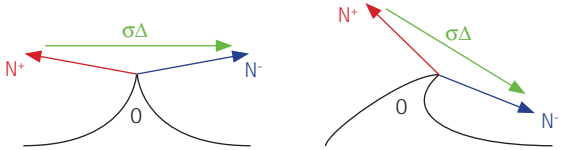
3. $\beta \in [-1;1]$ is called bias: it is used to locally control the direction of all Bézier boundary curves that are starting from a given tagged vertex. Here again three different congurations are interpolated: bias toward N^+ ($\beta > 0$), no bias ($\beta = 0$) and bias toward N^- ($\beta < 0$).



Sharpness



Tension



Bias

Recent developments for PN-Triangles

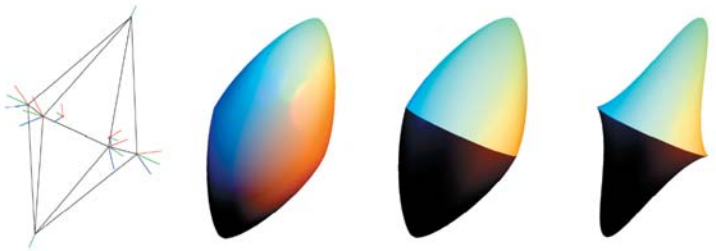
2.3 Mesh generation

In order to obtain a coherent effect of the shape parameters defined above, their influence has to be accounted both for the shading and the geometry of the surface generated during the rendering process.

(a) Coarse mesh with a ring of vertices tagged as sharp ($\sigma = 0.7$)
(b) Result obtained with standard PN-Triangles
(c) Result with sharpness only in shading
(d) Result with sharpness both in shading and geometry

Results:

· Sharpness value σ mainly acts on the shading
· Bias β and the tension θ mainly act on the silhouette of the object, and so on the underlying geometry.

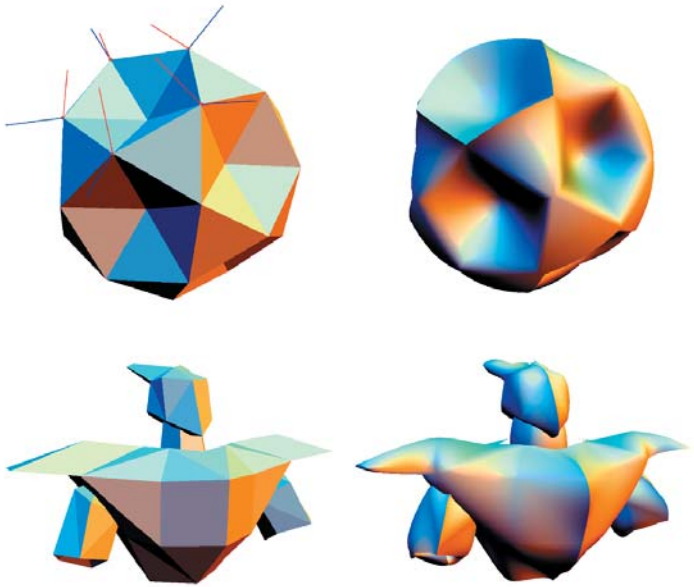


Recent developments for PN-Triangles

2.4 Conclusion

Scalar Tagged PN-Triangles:

- allows the user to design more complex shapes at a coarse level that will be dynamically refined preserving this shape parameters, which is an interesting property for real-time applications and compression
- need further investigation for the case of multiple sharp edges meeting in the same vertices, and how to locally and efficiently encode this kind of features



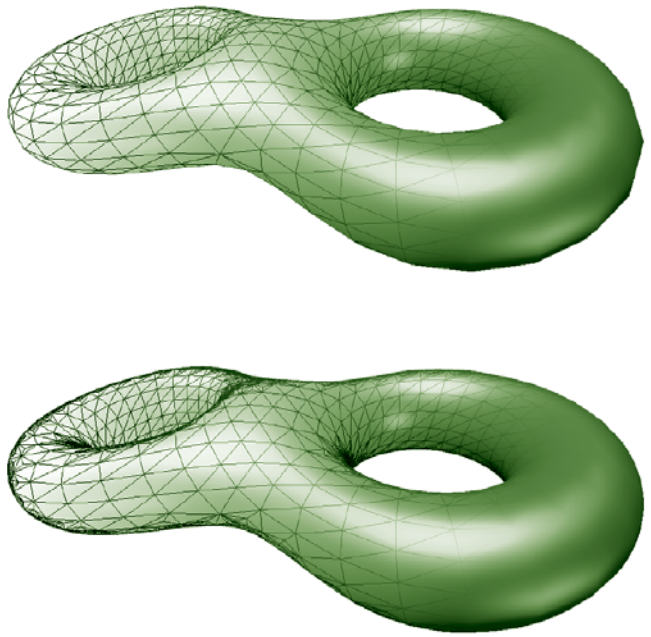
Recent developments for PN-Triangles

3. Real-time linear silhouette enhancement

Definition of a silhouette:

The silhouette separates an object from the background and is characterised by a sharp change in contrast and texture.

- In standard rendering methods, a sharp silhouette of a given mesh is a problem according to the underlying basic structure of triangles.
- Replace linear silhouette edges with smooth curves and refine the local geometry accordingly during rendering.
- Define a continuous "silhouetteness" test and use this to blend flat and curved geometry in order to avoid transitional artifacts.



Recent developments for PN-Triangles

3.1 Detection of silhouettes

Definition of front- and back-facing:

· View point $o \in \mathbb{R}^3$ is the position of the observer
· For a point p on the mesh, the view direction vector is $p - o$

Let n be the surface normal in p, then the definition is:

The mesh is front-facing for $(p - o) \cdot n \leq 0$ and back-facing otherwise

The silhouette of a triangle mesh is the set of edges where one of the adjacent faces is front-facing while the other is back-facing. A function to determine if an edge e_{ij} (shared by two triangles T_s and T_t) is a silhouette edge uses the definiton above, the normals n_s and n_t and the midpoint p_{ij} of the given edge:

$$f_{ij}(x) := \left(\frac{(p_{ij} - x)}{||(p_{ij} - x)||} \cdot n_s \right) \left(\frac{(p_{ij} - x)}{||(p_{ij} - x)||} \cdot n_t \right)$$

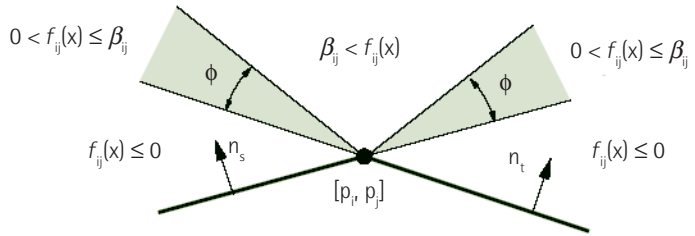
The edge is now a silhouette edge if $f_{ij}(o) \leq 0$.

Recent developments for PN-Triangles

3.2 Rendering silhouettes

Because "silhouetteness" is a binary function of the view-point, a naive rendering implementation leads to transitional artifacts; the rendered geometry depends discontinuously on the view-point. Propose instead to make the rendered geometry depend continuously on the view-point. Define the silhouetteness of e_{ij} seen from $x \in R^3$ to be:

$$\alpha_{ij}(x) := \begin{cases} 1 & f_{ij}(x) \le 0 \\ 1 - f_{ij}(x)/\beta_{ij} & 0 < f_{ij}(x) \le \beta_{ij} \\ 0 & \beta_{ij} < f_{ij}(x) \end{cases}$$



where $\beta_{ij} \ge 0$ is a constant

Heuristic showed that $\beta_{ij} = 0.25$ worked well in practice

Recent developments for PN-Triangles

3.3 View dependent geometry

Define for each edge e_{ij} in the mesh a smooth edge curve on the Bézier form

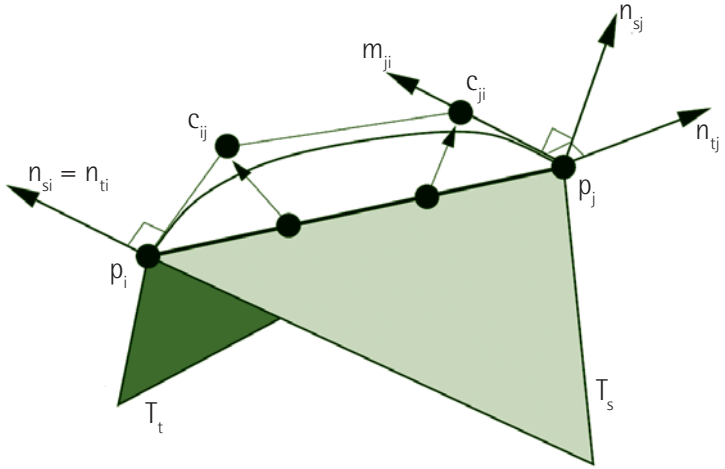
$$C_{ij}(t) = p_i B_0^3(t) + c_{ij} B_1^3(t) + c_{ji} B_2^3(t) + p_j B_3^3(t)$$

This cubic curve interpolates its end points p_i and p_j , and it remains to determine the two inner control points c_{ij} and c_{ji}. If the two shading normals n_{si}, n_{ti} are equal, the edge end p_i is smooth, otherwise it is a feature edge end.

Define $l_{kl} = (2p_k + p_l) / 3$ and $m_{ij} = \dfrac{(n_{ti} \times n_{si})}{\|(n_{ti} \times n_{si})\|}$ then c_{ij} is defined as follows:

For a smooth edge end $c_{ij} = l_{ij} - \dfrac{(p_j - p_i) \cdot n_{si}}{3} n_{si}$

For a feature edge end $c_{ij} = p_i + \dfrac{(p_j - p_i) \cdot m_{ij}}{3} m_{ij}$



Recent developments for PN-Triangles

3.4 Implementation with PN-Triangles

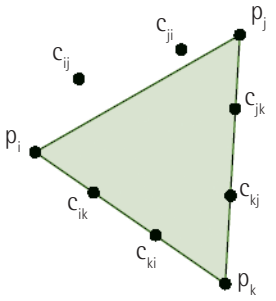
$$b_{300} = p_i$$
$$b_{030} = p_j$$
$$b_{003} = p_k$$

$$b_{201} = \alpha_{ik} c_{ik} + (1-\alpha_{ik}) l_{ik}$$
$$b_{102} = \alpha_{ki} c_{ki} + (1-\alpha_{ki}) l_{ki}$$
$$b_{120} = \alpha_{ji} c_{ji} + (1-\alpha_{ji}) l_{ji}$$
$$b_{210} = \alpha_{ij} c_{ij} + (1-\alpha_{ij}) l_{ij}$$
$$b_{012} = \alpha_{kj} c_{kj} + (1-\alpha_{kj}) l_{kj}$$
$$b_{021} = \alpha_{jk} c_{jk} + (1-\alpha_{jk}) l_{jk}$$

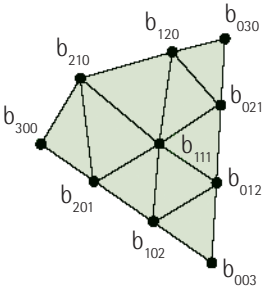
$$b_{111} = (3/12)(b_{201} + b_{120} + b_{012} + b_{102} + b_{210} + b_{021}) - (1/6)(b_{300} + b_{030} + b_{003})$$

If $\alpha_{ij} = 1$ for all edges, the resulting patches equals the ones in the PN-triangle construction.

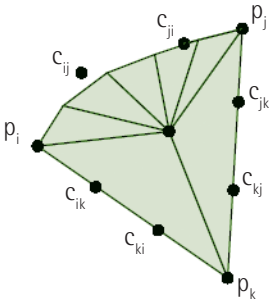
This construction yields for a given viewpoint a spline surface that is a blend between the geometry of the PN-triangle construction and the mesh itself. The blend is such that the surface is smooth near a silhouette and flat elsewhere



(a) The edge curve control points of a triangle with one silhouette edge



(b) The control points of the corresponding patch

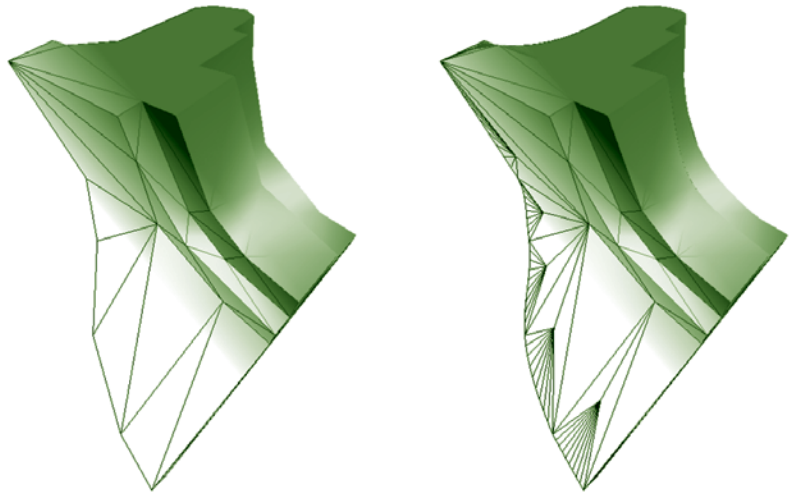


(c) A possible rendering strategy for a curved patch

Recent developments for PN-Triangles

3.5 Conclusion

This algorithm should be a strong candidate for GPU implementation. However, determining silhouetteness requires connectivity information and refinement generates new geometry. These operations are not supported by current versions of OpenGL or DirectX GPU programs



Recent developments for PN-Triangles

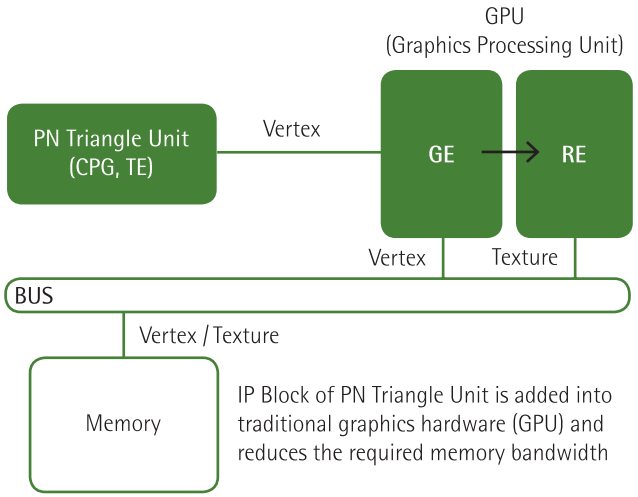
4. Hardware implementation of PN-Triangles

Reducing the required memory bandwidth is a main issue in 3D computer graphics. PN triangle solves the memory bandwidth problem by using curved surface representation and tessellation.

· Reduces bandwidth consumption.
· Simple dedicated hardware executing complex operations of tessellation by adopting the de Casteljau algorithm.
· The proposed PN triangle generation unit can be constructed with only one linear interpolator, a number of temporal registers and control logic.

The PN triangle generation unit implemented in graphics hardware achieves several benefits: fast execution, power saving, less memory bandwidth consumption, and etc. A hardware architecture of PN triangle generation unit is divided by two parts:

Control Point Generation (CPG) and Tessellation (TE)



Recent developments for PN-Triangles

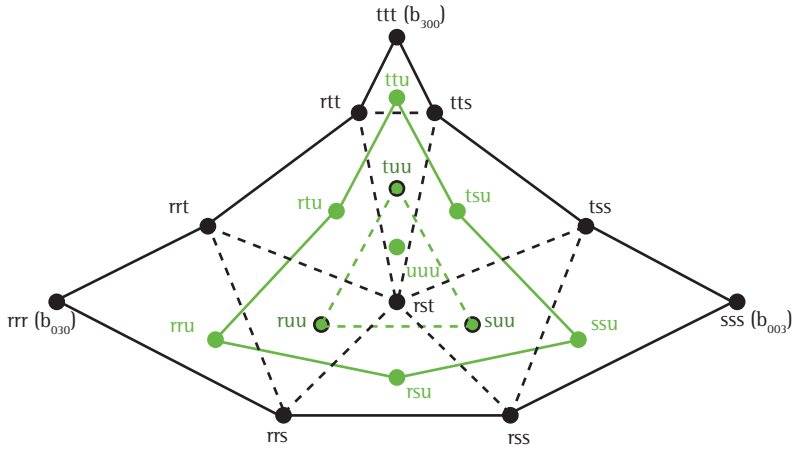
4.1 Control Point Generation and Tessellation

Control Point Generation (CPG):

· Operations for CPG are also executed in geometry processing engine
· Use remained clock cycles of the geometry processing engine for CPG
· Evaluations are performed only one time for each flat triangle
· The CPG part consists of only registers for storing control points

Tessellation (TE):

· Use of the de Casteljau algorithm to evaluate Bézier surface functions
· De Casteljau does not make the problem of numerical instability and is appropriate for hardware implementation because of its repetitive property
· 10 linear interpolations are performed for the evaluation
· TE is simply constructed with a number of linear interpolator (INT) blocks and temporal registers storing intermediate values.



de Castlejau algorithm for triangular cubic Bézier surface

Recent developments for PN-Triangles

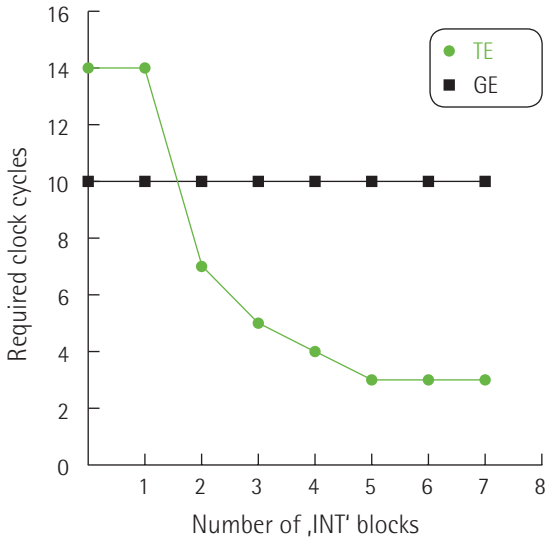
4.2 Load Balancing

The number of 'INT' blocks implemented in TE depends on a hardware designer's choice.

- If a single 'INT' block is used in TE, hardware cost is minimal but the geometry processing engine must wait for TE every time until a new vertex is generated.
- If a lot of 'INT' blocks are used, the tessellation is executed quickly but TE must wait for the geometry processing engine to finish the geometry operations before generating a new vertex.

Load Balancing:

- If two 'INT' blocks are implemented in TE, the performance of TE catches up with the performance of the geometry processing engine
- For the best performance and the least hardware cost, the optimal number of 'INT' blocks in TE is two.



Recent developments for PN-Triangles

5. References

1. Alex Vlachos, Jörg Peters, Chas Boyd and Jason L. Mitchell. Curved PN Triangles
2. Gerald Farin. PN Patches. February 16, 2003
3. Tamy Boubekeur, Patrick Reuter and Christophe Schlick. Scalar Tagged PN Triangles. EUROGRAPHICS 2005 / J. Dingliana and F. Ganovelli
4. Christopher Dyken and Martin Reimers. Real-time linear silhouette enhancement
5. Kyusik Chung and Lee-Sup Kim. A PN Triangle Generation Unit for Fast and Simple Tessellation Hardware. Department of EECS, KAIST, 373-1, Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea