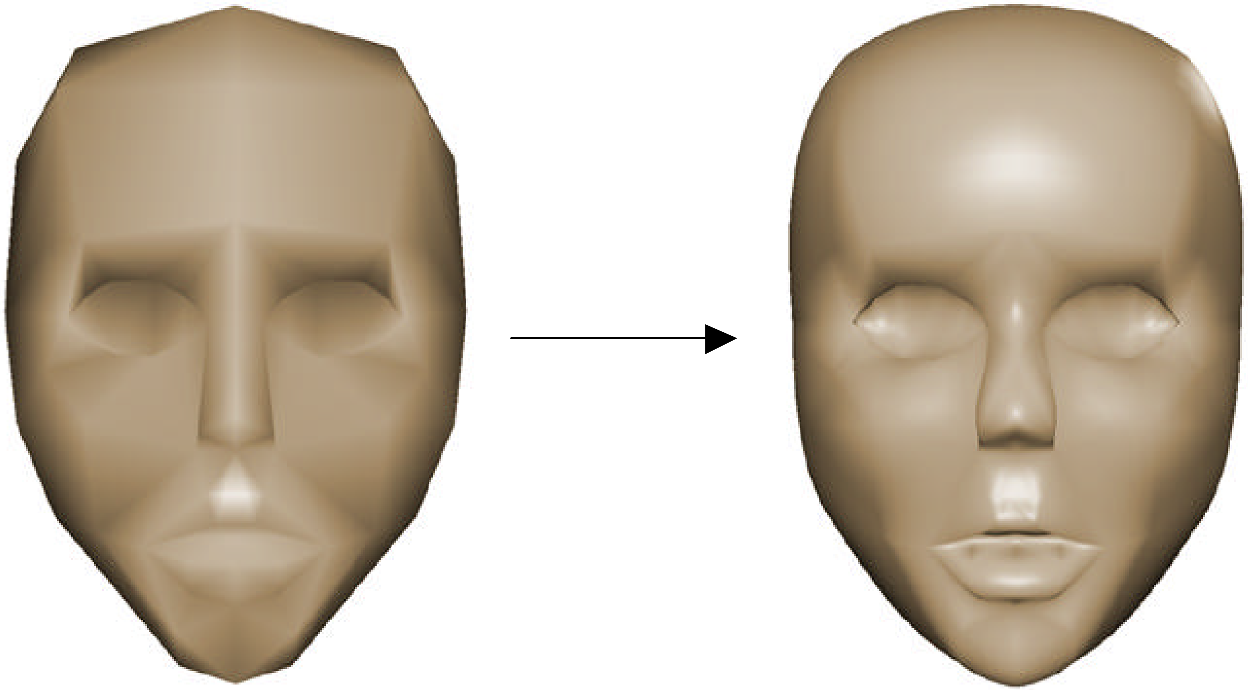
Curved PN Triangles

Alex Vlachos
AVlachos@ati.com

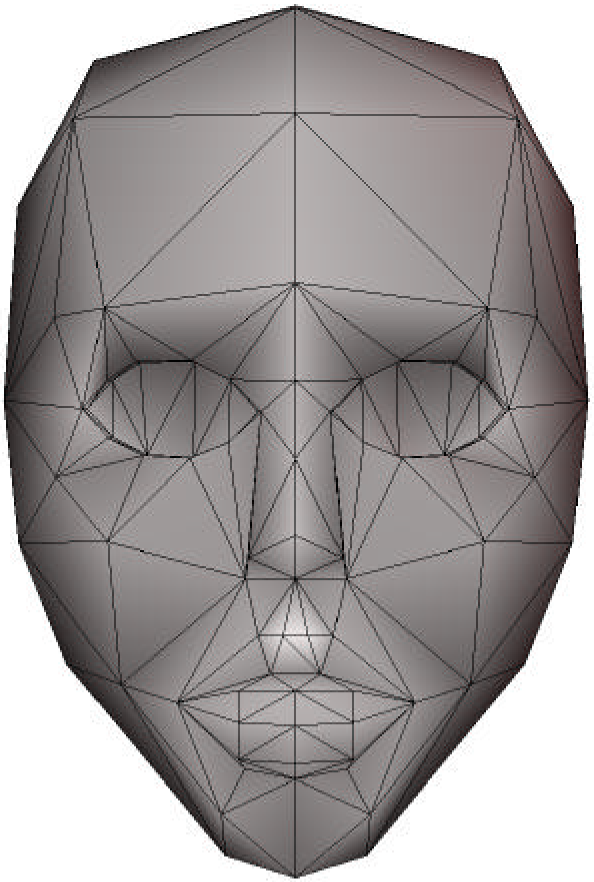
Jörg Peters
jorg@cise.ufl.edu

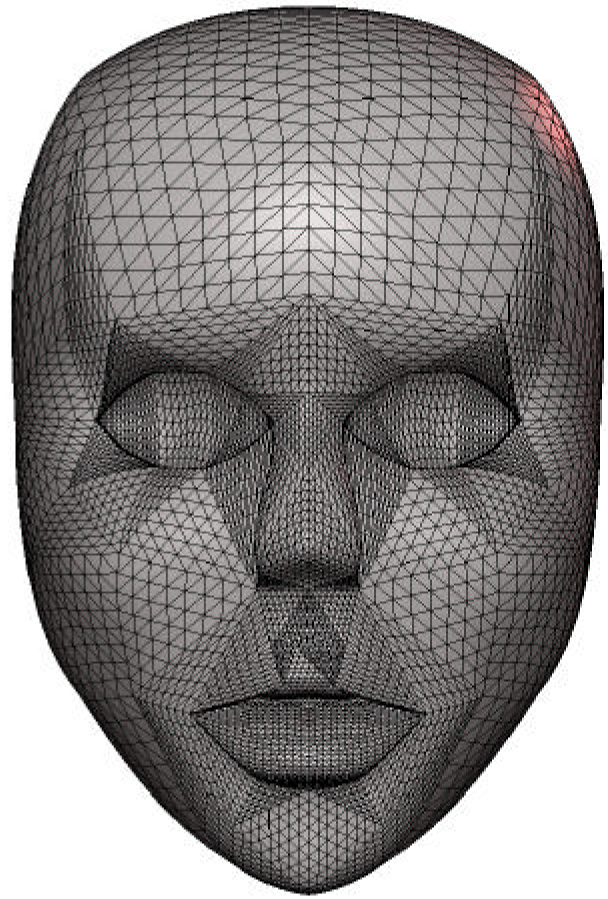


Outline

- Motivation
- Constraints
- Surface Properties
- Performance
- Demo

Quick Demo





Constraints

- Software Developers
  - Must fit into their art and engineering pipelines
  - Must be compatible with work already in progress
  - Must be backward compatible
- API's: OpenGL & Microsoft's Direct3D
  - Expressible through the API concisely
  - Minimal change (vertex buffers and index buffers)
- Hardware
  - Performance
  - Cost
  - Fits current architecture

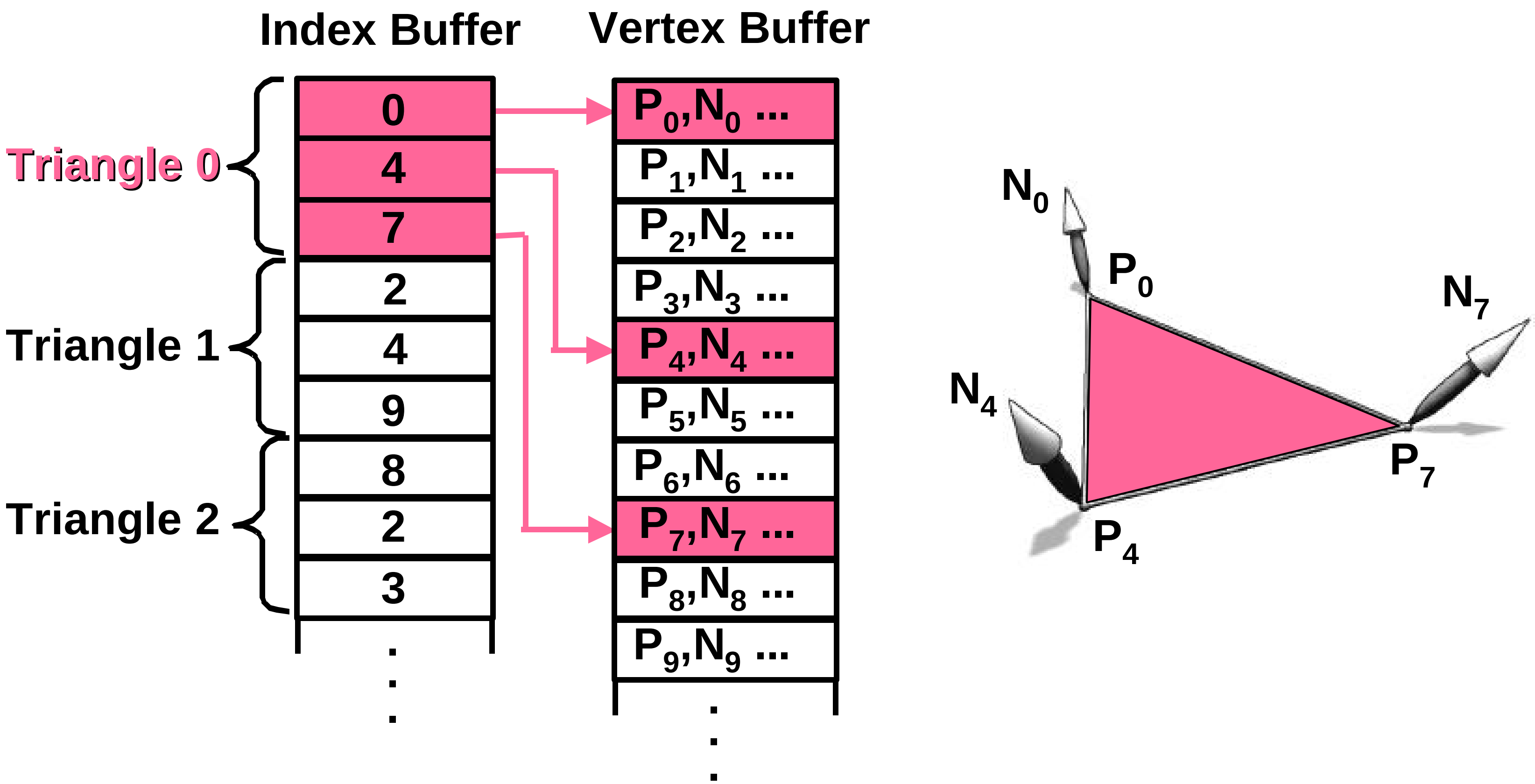
Goal: Improve Visual Quality

- No paradigm change for developers & artists
  - Do not require developers to store geometry differently (triangles)
  - Wide array of art content is usable
  - Backwards compatible
- Minimize change to API's
  - Use existing data structures
  - No explicit connectivity information
- Fit existing hardware designs

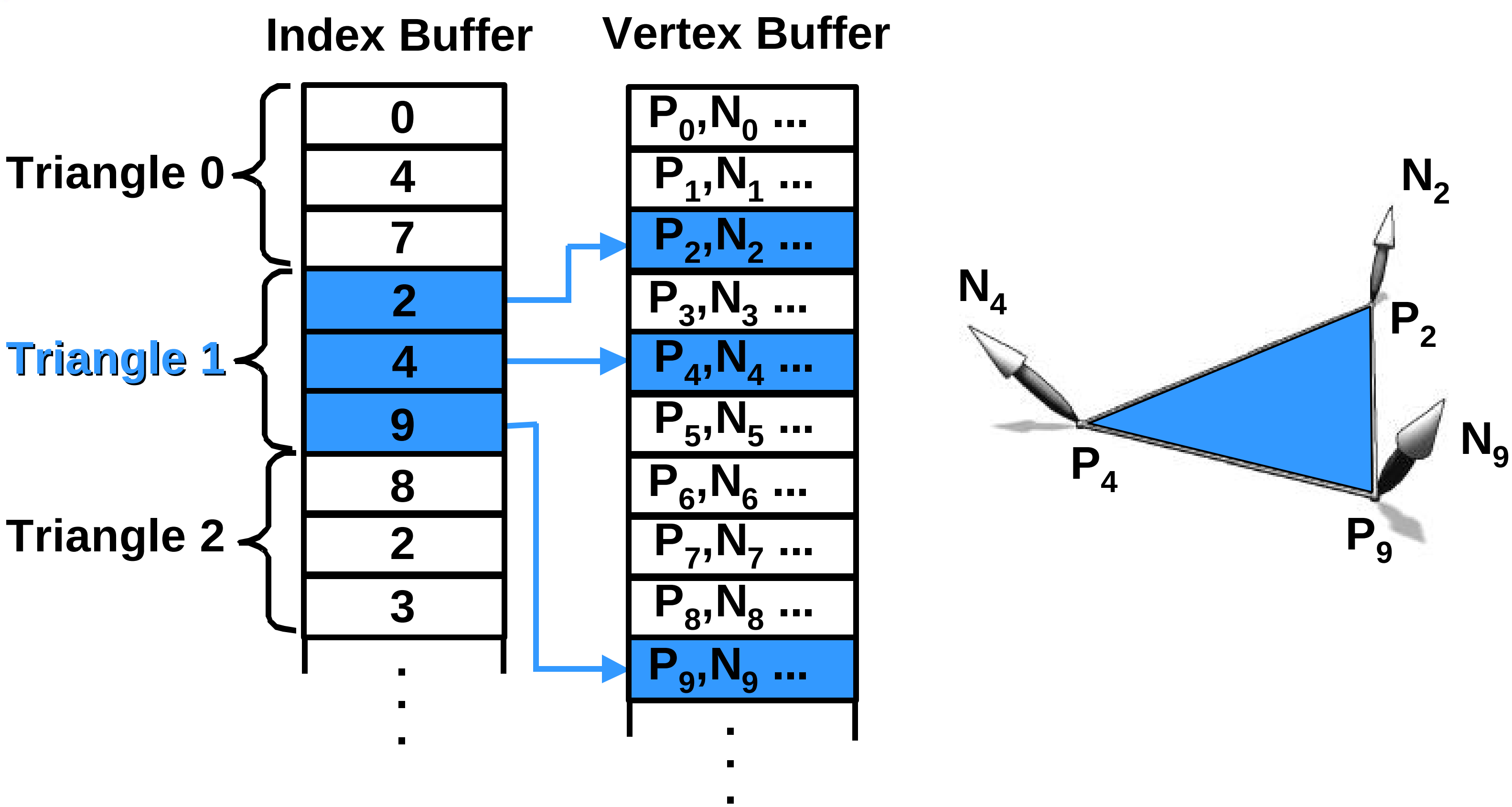
Rendering Curved Surfaces

- Visual cues
  - Smooth silhouettes
  - Lighting cues on the interior of polygons
- Separability of normal and position
  - Familiar paradigm: bump mapping
  - Visual Smoothness
- Curved PN Triangles provide
  - Improved silhouettes
  - Additional sample points for per-vertex operations
    - Lighting operations (approximate Phong shading)
    - Vertex shaders

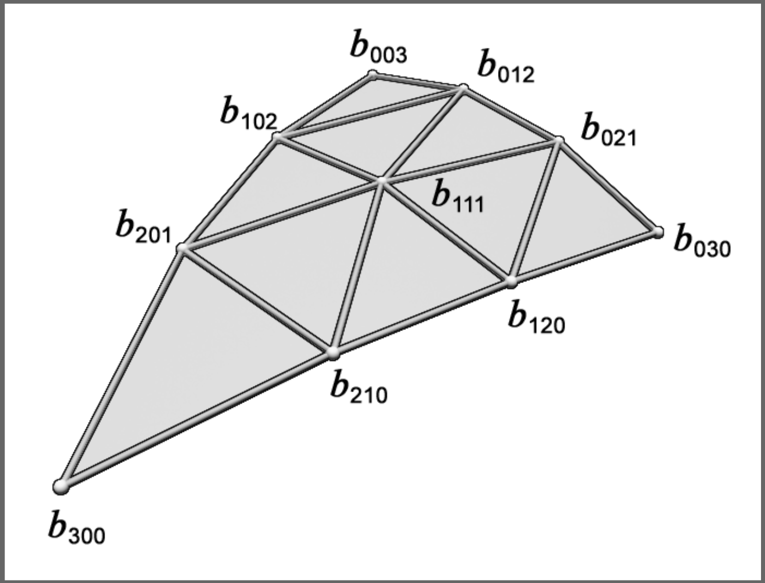
Vertex and Index Buffers



Vertex and Index Buffers



Geometry: cubic Bézier (de Casteljau) form



$\mathbf{b}: \quad R^2 \mapsto R^3$

b_{ijk} = control points = coefficients

control net

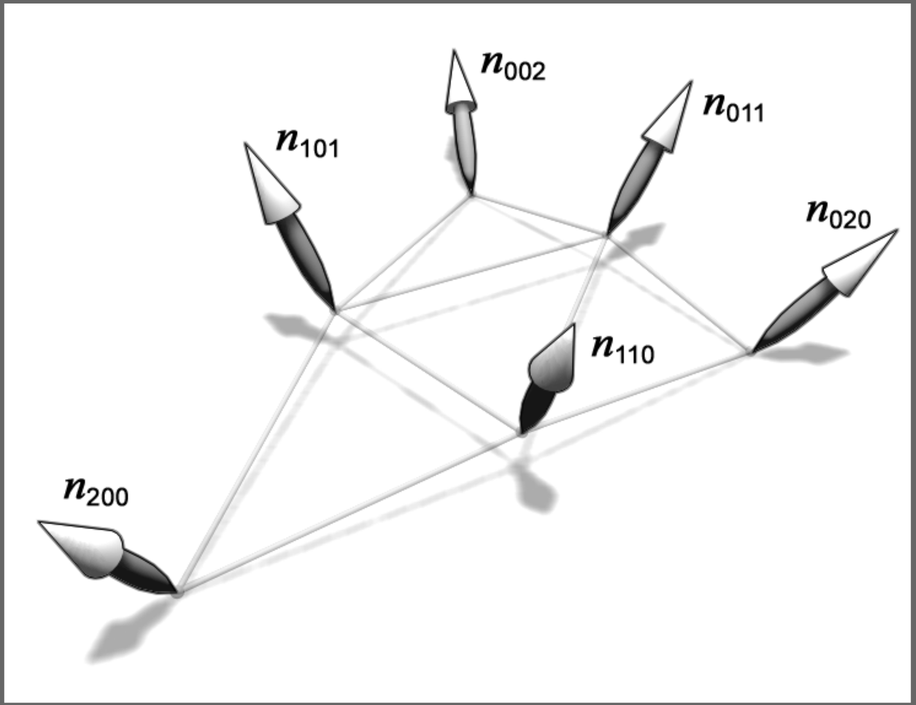
$$\mathbf{b}(u,v) = \begin{array}{llll} b_{003}v^3 & & & \\ +b_{102}3wv^2 & +b_{012}3uv^2 & & \\ +b_{201}3w^2v & +b_{111}6wuv & +b_{021}3u^2v & \\ +b_{300}w^3 & +b_{210}3w^2u & +b_{120}3wu^2 & +b_{030}u^3 \end{array}$$

$$w = 1 - u - v$$
$$u, v, w \geq 0.$$

Normal: quadratic Bézier (de Casteljau) form

$\mathbf{n} : \quad R^2 \mapsto R^3$

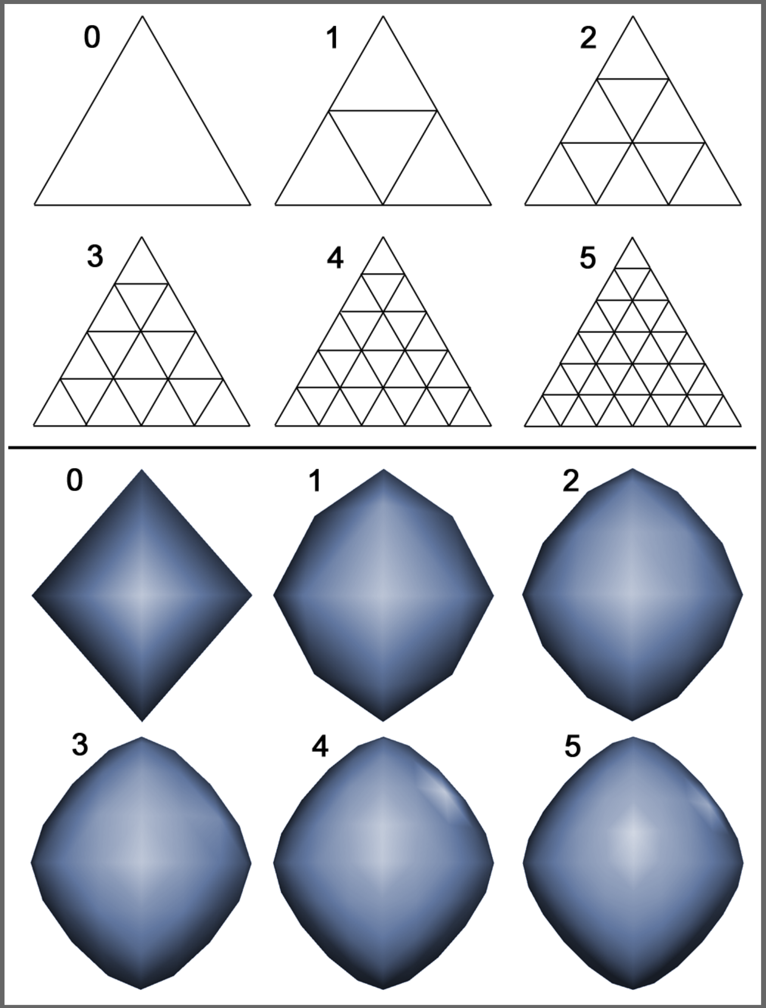
$$\mathbf{n}(u,v) = \begin{array}{ccc} & n_{002}v^2 & \\ & +n_{101}2wv \quad +n_{011}2uv & \\ & +n_{200}w^2 \quad +n_{110}2uw & \\ & \quad\quad +n_{020}u^2 & \end{array}$$



(or $\mathbf{n}$ linear == Phong shading)

Evaluation API

`lod` $\in \{0, 1, 2, \ldots\}$ = number of evaluation points on one edge minus two



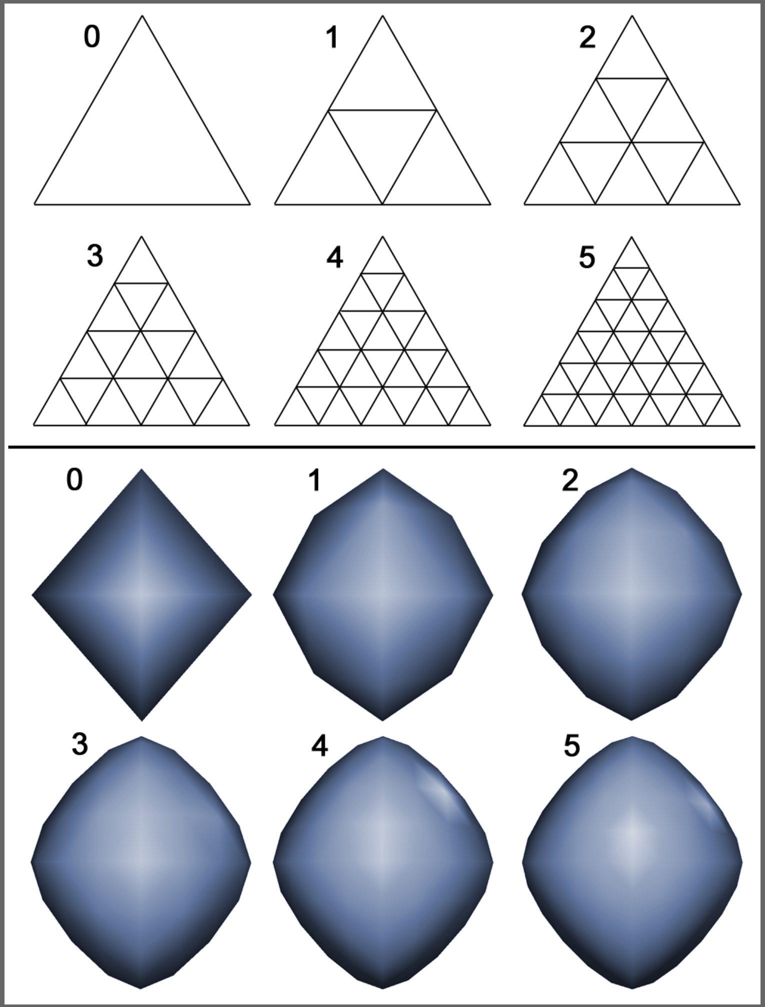
Insertion into Graphics Pipeline

On chip

- Vertex and primitive assembly
- Vertex shading
- Triangle setup
- Rasterization

Insertion into Graphics Pipeline

`lod`



On chip

Vertex and primitive assembly

. in 1 triangle (P & N)

Curved PN triangle tesselation

. out $(\texttt{lod} + 1)^2$ microtriangles

Vertex shading

Triangle setup

Rasterization

Why this choice?

Constraints:

- Isolation (cannot access mesh neighbors)
- Fast Evaluation (including normal)
- Modeling range (smoother contours and better shading)
- Symmetry (aesthic)

Constraints:

- Isolation
- Fast Evaluation
- Modeling range
- Symmetry

Decision:

- Rules out subdivision or surface spline
  Continuous normal without neighbor?
  (Prescribe normal along boundary – but shape can be poor)

.

Constraints:

- Isolation
- Fast Evaluation
- Modeling range
- Symmetry

Decision:

- No subdivision or surface spline C^1? (can't see neighbor!)
- No rational surface (normal of) rational surface is expensive [Gregory, Chiyokura, Grimm & Hughes]
  not multiple pieces [Clough-Tocher, Shirman-Sequin, Powell-Sabin]
  $\rightarrow$ polynomial

.

Constraints:

- Isolation
- Fast Evaluation
- Modeling range
- Symmetry

Decision:

- No subdivision or surface spline C^1? (can't see neighbor!)
- Not rational parametrization
  not multiple pieces
  $\rightarrow$ polynomial
- 'shape preservation'
  Inflections: geometry cubic, normal quadratic
  predictability: want curved PN triangle 'close to' the flat triangle

Constraints:

- Isolation
- Fast Evaluation
- Modeling range
  (shape preservation)
- Symmetry

Decision:

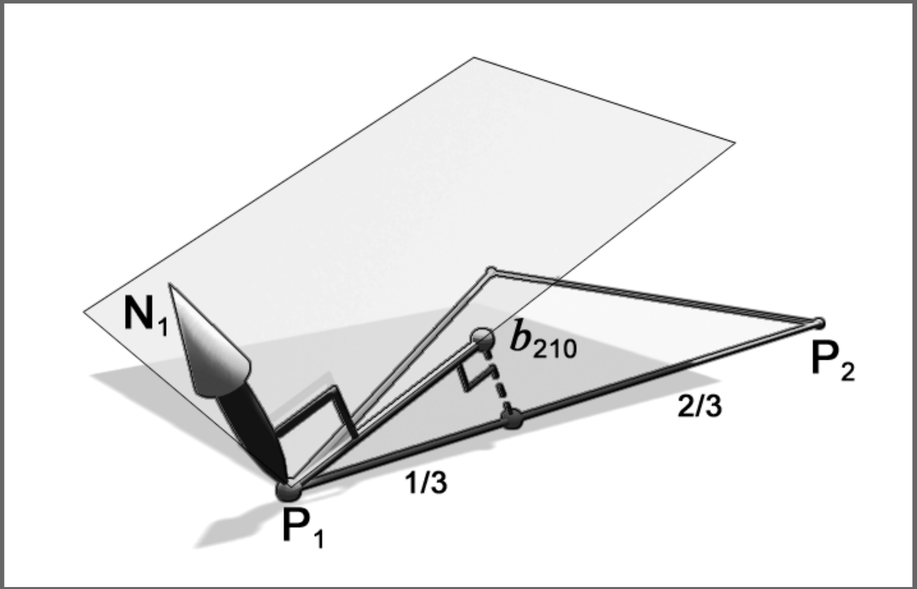
- No subdivision or surface spline
  C^1? (can't see neighbor!)
- Not rational surface
  not multiple pieces
  $\rightarrow$ polynomial
- Inflections: geometry cubic, normal quadratic
  predictable: prove curved patch is 'close to' the flat triangle
- 3-sided Bézier patch.

Geometry coefficients

$b_{300} = P_1$ etc.

$b_{210} = (2P_1 + P_2 - w_{12}N_1)/3$

. $\quad w_{ij} = (P_j - P_i) \cdot N_i \in \mathbf{R}$



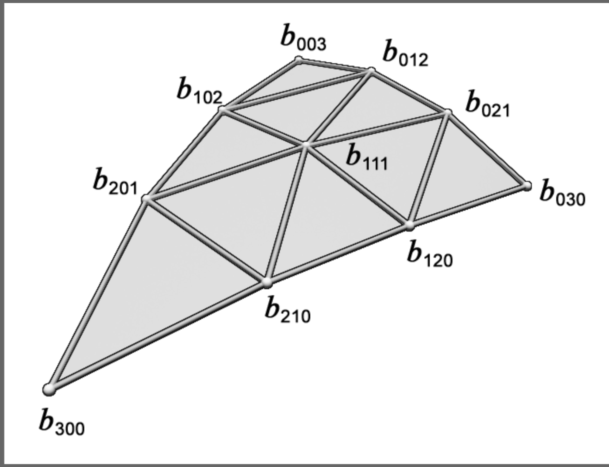
$b_{111} = E + (E - V)/2$

. $\quad E = (b_{210} + b_{120} + b_{021}$

. $\quad\quad +b_{012} + b_{102} + b_{201})/6$

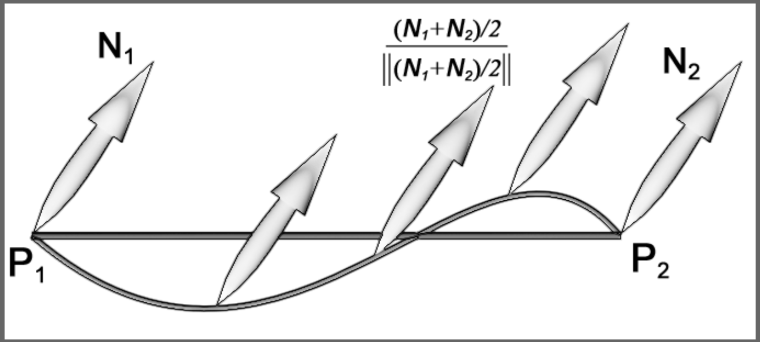
. $\quad V = (P_1 + P_2 + P_3)/3.$

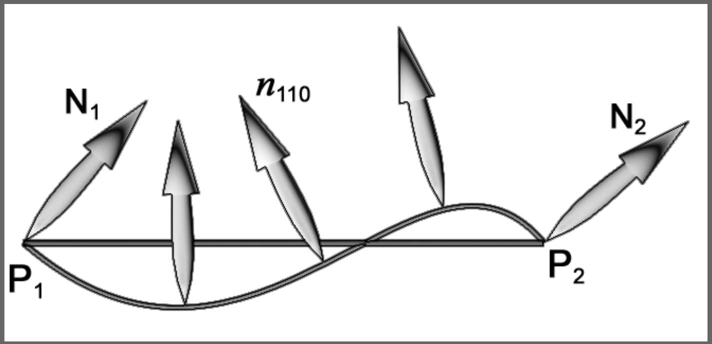
(Farin 83)

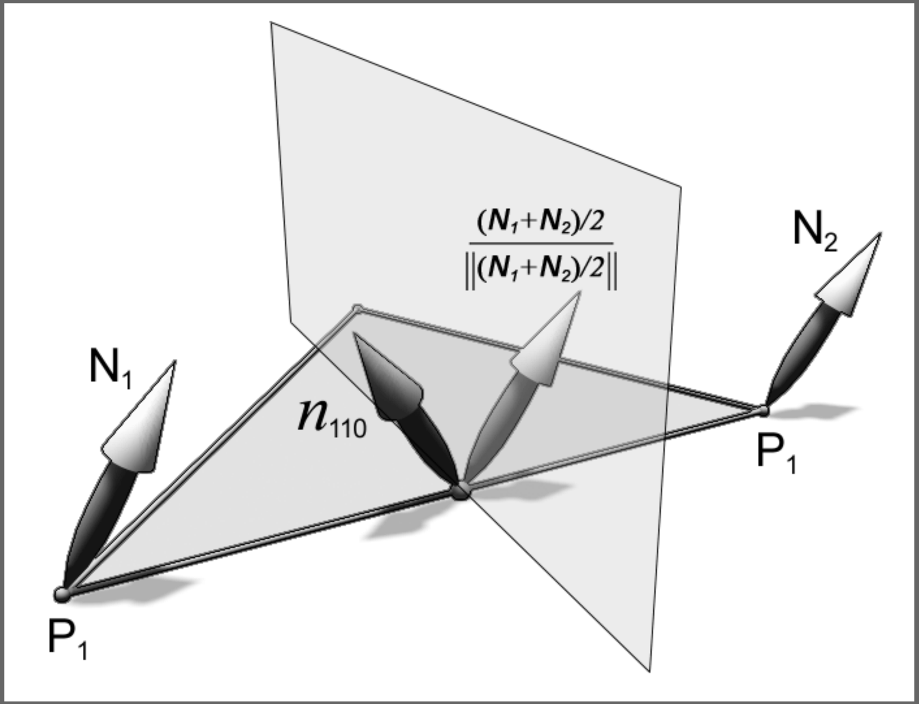


Normal coefficients

Linear vs Quadratic Normal







Normalize

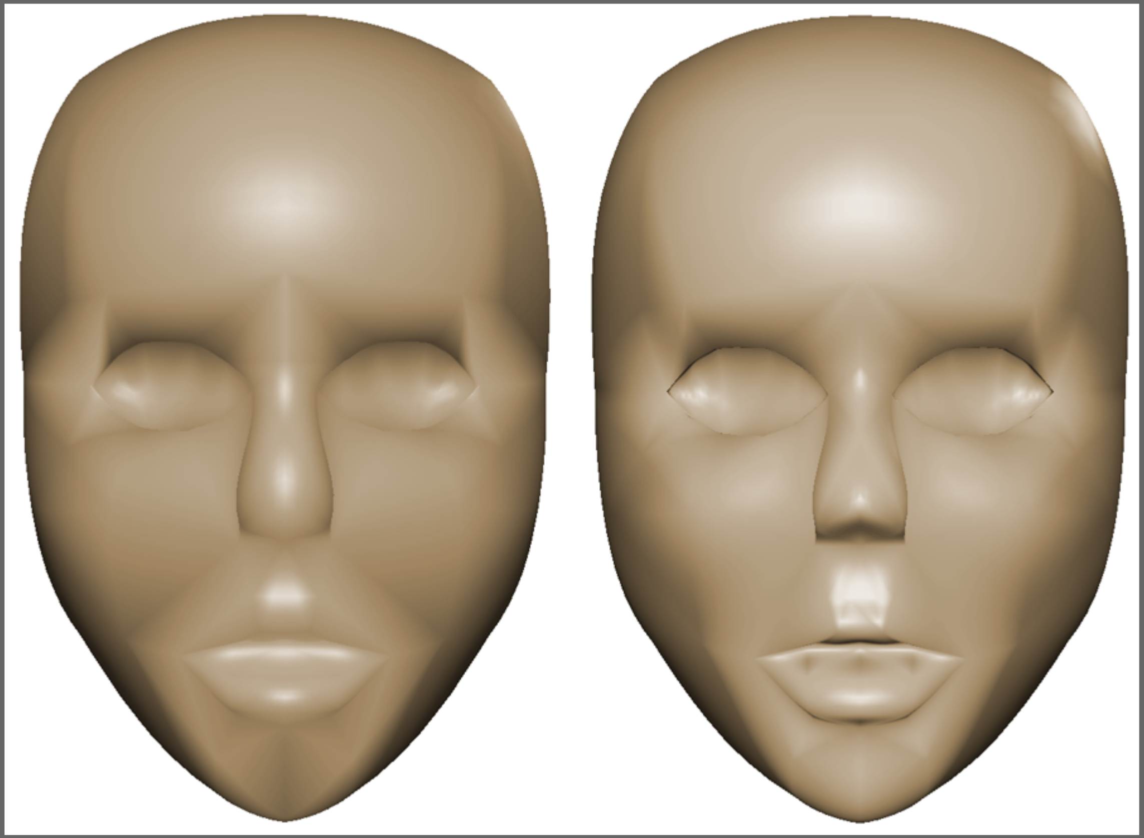
$$h_{110} = \frac{N_1+N_2}{2} - \frac{v_{12}}{2}(P_2 - P_1)$$

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

(VanOverveld, Wyvill)

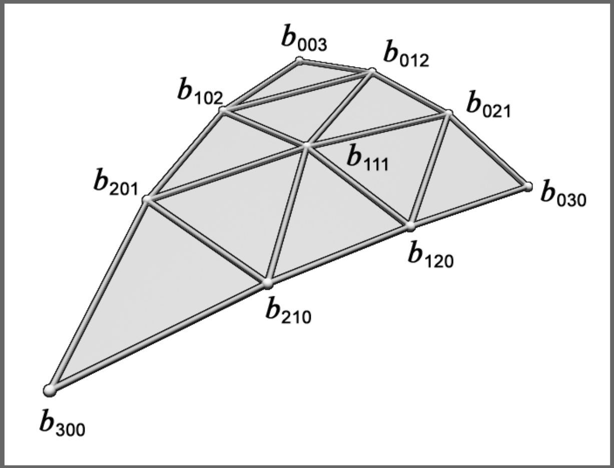
Normal coefficients

Linear vs Quadratic Normal

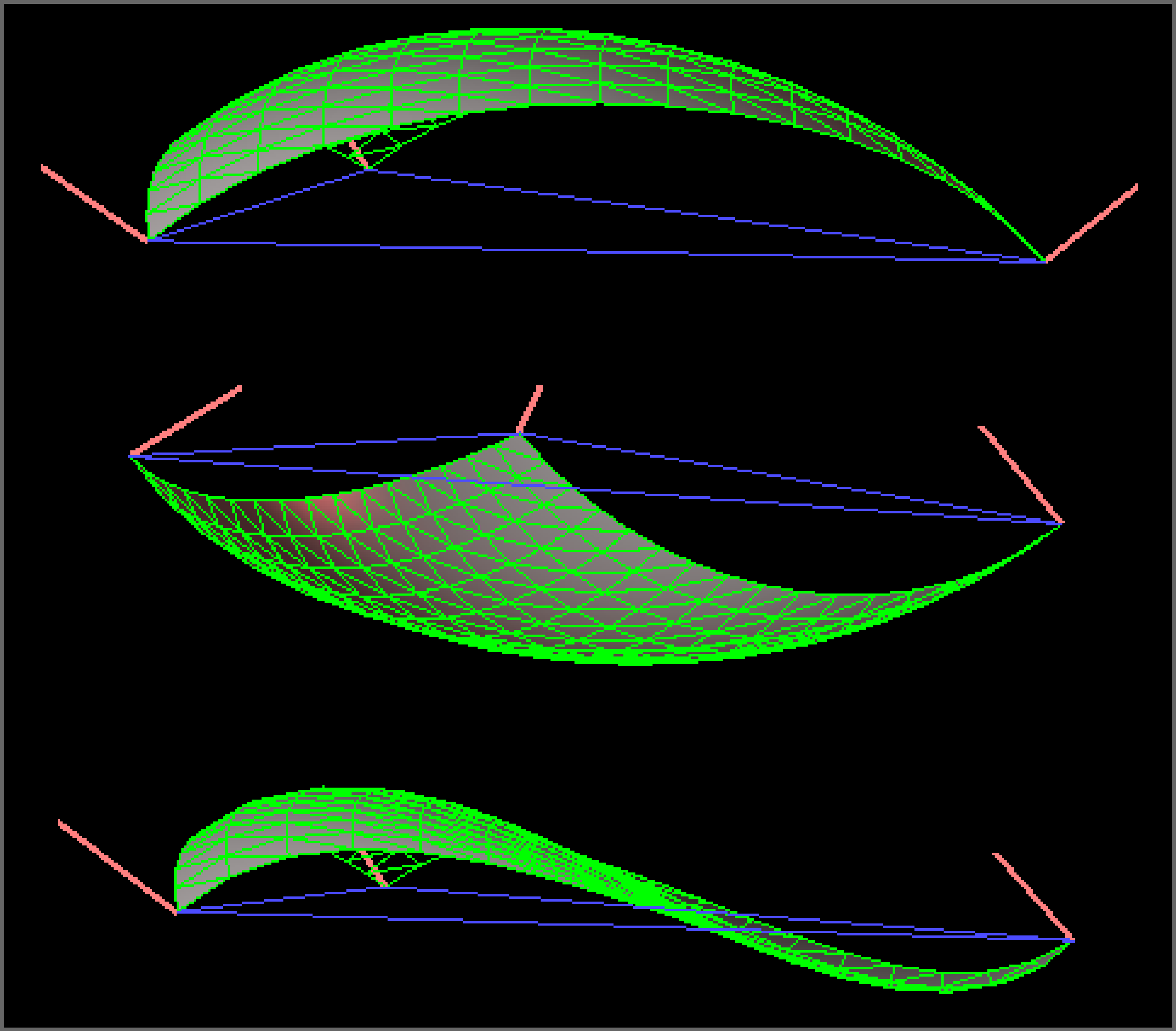


Properties

- Interpolates P_i and N_i.
- b_{210} on circle with center $P_1 + L$ and radius $\|L\|$. $L := (P_1 - P_2)/6$
- Quadratic precision: b_{111}.
- Surface is smooth at corners;
  Surface looks smooth (except possibly at silhouttes)



PN triangle calisthenics



Performance

- Geometry, textures, display lists, shaders etc compete for memory
- Bandwidth savings
- Now able to actually feed high performance transform/shader engines

Basic Advantages

- Coarse triangulations yield geometry compression
- Level of detail (LOD)
- Geometry looks smoother!

Summary

- API-friendly
- Developer-friendly
- Provides a vast visual improvement to existing and future content
  - Improve silhouettes
  - Improve lighting calculations
- Advantage of geometry compression and memory savings
- Overall performance is increased