SIGGRAPHASIA2009

革新の波動

the pulse of innovation

Approximating Subdivision Surfaces with Gregory Patches for Hardware Tessellation

Charles Loop
Microsoft Research

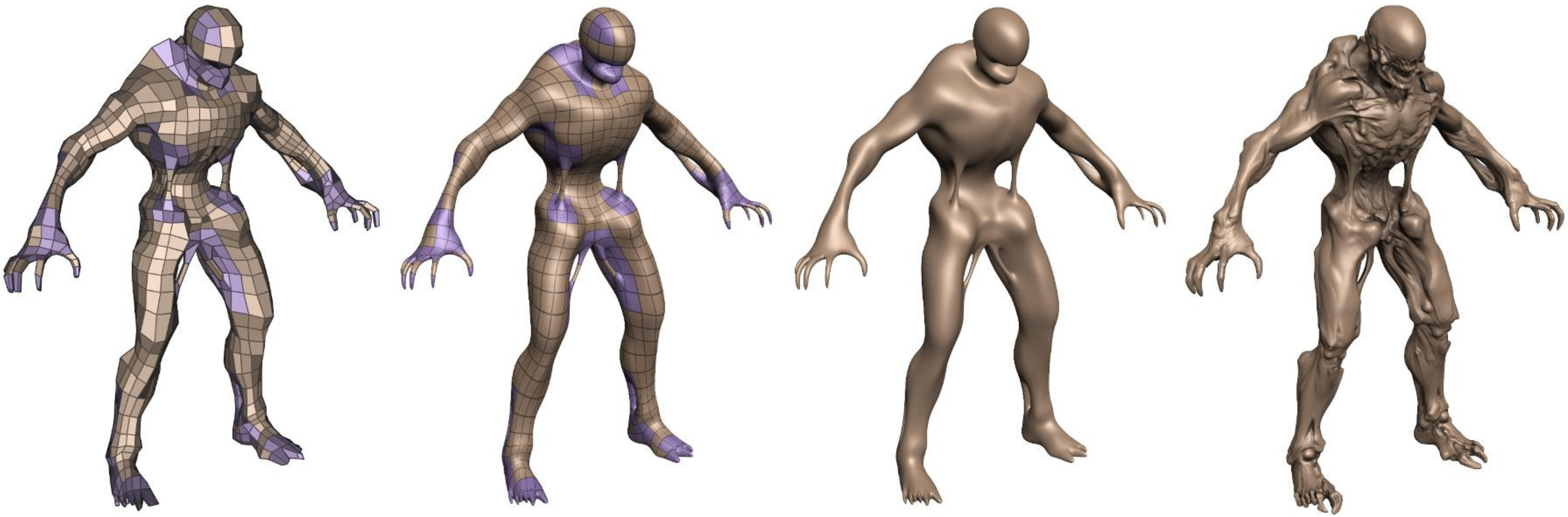
Scott Schaefer
Texas A&M University

Tianyun Ni
NVIDIA

Ignacio Castaño
NVIDIA

SIGGRAPHASIA2009

Goal



Real-Time Displaced ‘Subdivision Surfaces’

SIGGRAPHASIA2009

Problem: Real-Time Animation

- Each vertex 'touched' at runtime
  - new position influenced by many *bones weights* or *morph targets*
- Costly for dense meshes
- Coarse meshes are used
  - faceting artifacts
- Dense static objects
  - high disk/bus consumption

Solution: Hardware Tessellation

- Store/send coarse mesh to GPU
- Animate coarse mesh vertices
  - inexpensive
- Expand geometry on GPU
  - reduce bus traffic
  - exploit GPU parallelism
- Better shape fidelity
  - reduced faceting
  - displacement mapping

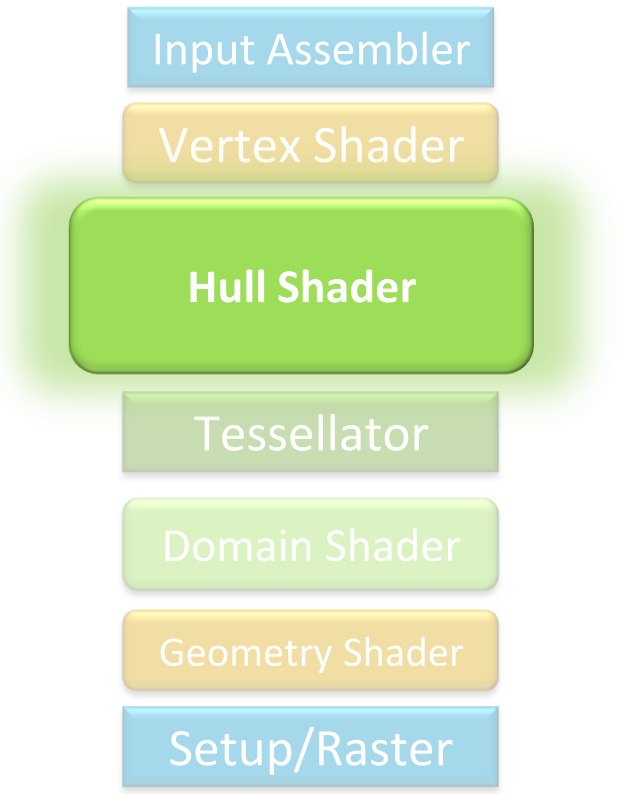
Tessellation Pipeline

- Direct3D11 has support for programmable tessellation

- Two new programable shader stages:
  - Hull Shader (HS)
  - Domain Shader (DS)
- One fixed function stage:
  - Tessellator (TS)

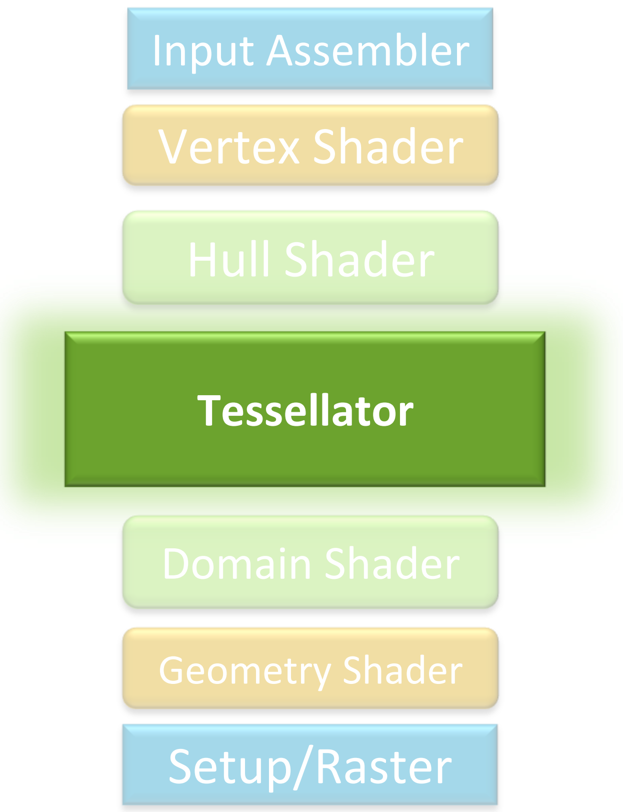
Hull Shader (HS)

- Transforms control points from *irregular* control mesh data to *regular* patch data
- Computes edge tessellation factors

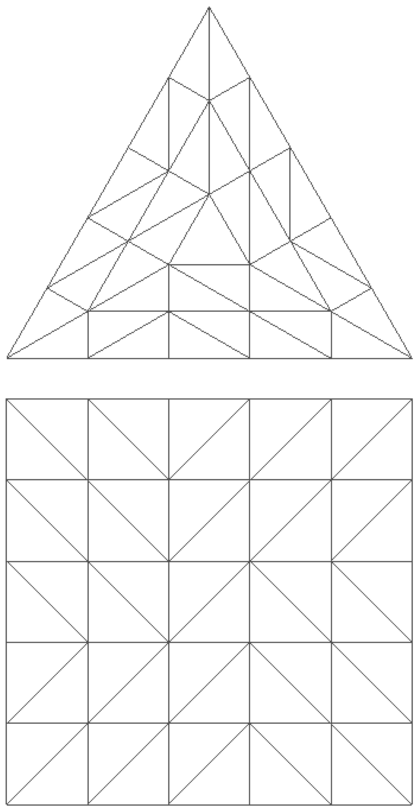


Tessellator (TS)

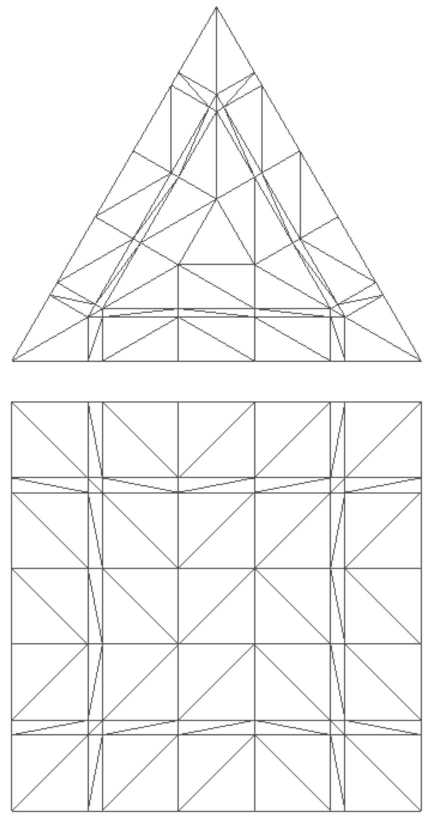
- Fixed function stage, but configurable
- Domains:
  – Triangle, Quad, Line
- Spacing:
  – Discrete, Continuous, Pow2



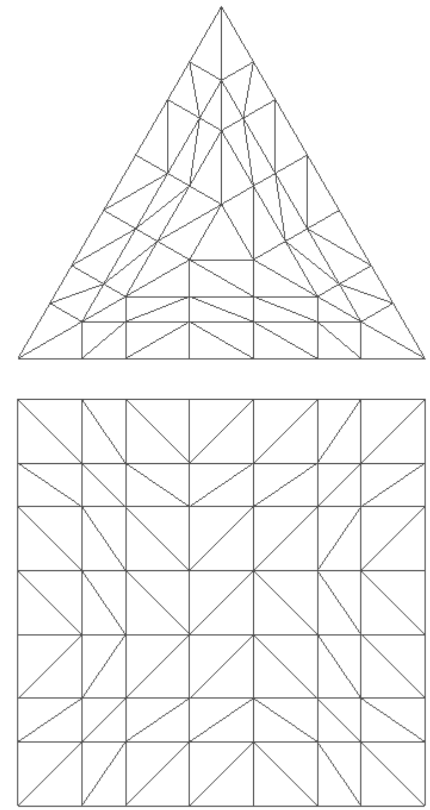
Tessellator (TS)



Level 5



Level 5.4



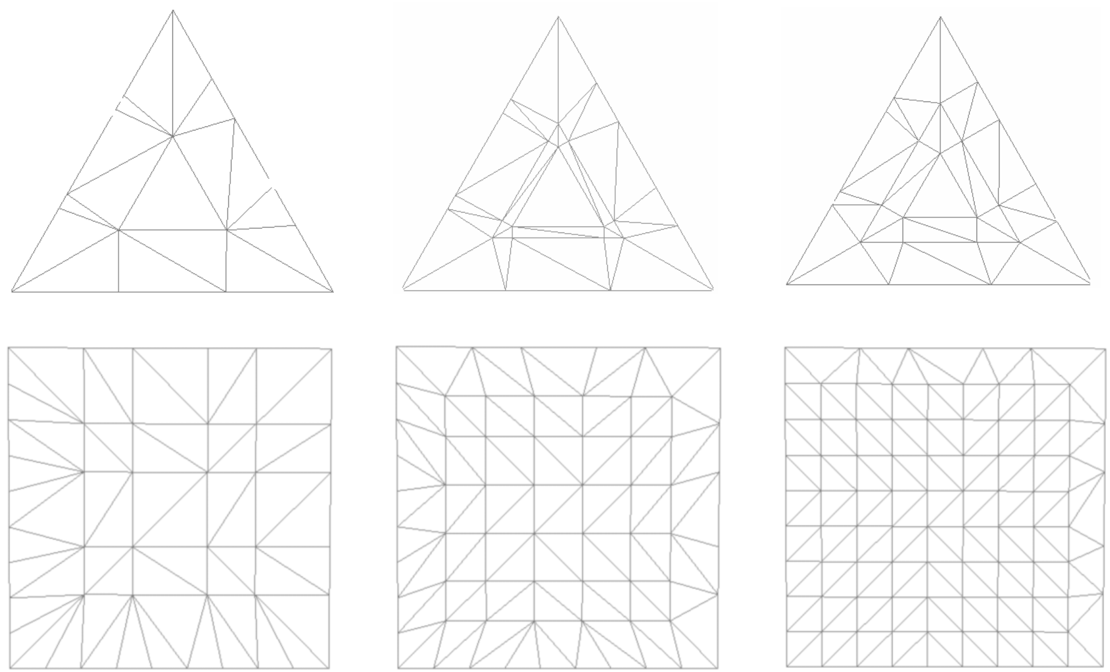
Level 6.6

Tessellator (TS)

Left = 3.5
Right = 4.4
Bottom = 3.0

Top,Right = 4.5

Bottom,Left = 9.0



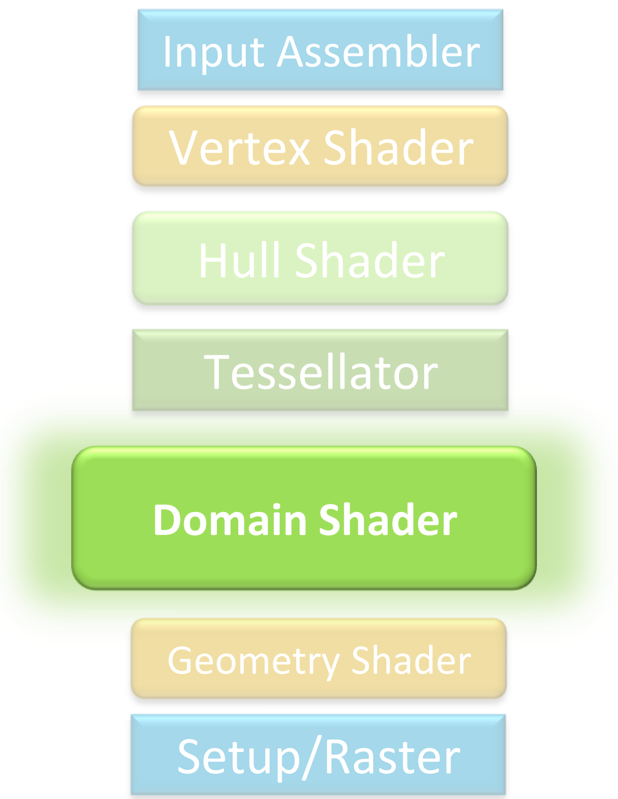
Inside Tess: minimum

Inside Tess: average

Inside Tess: maximum

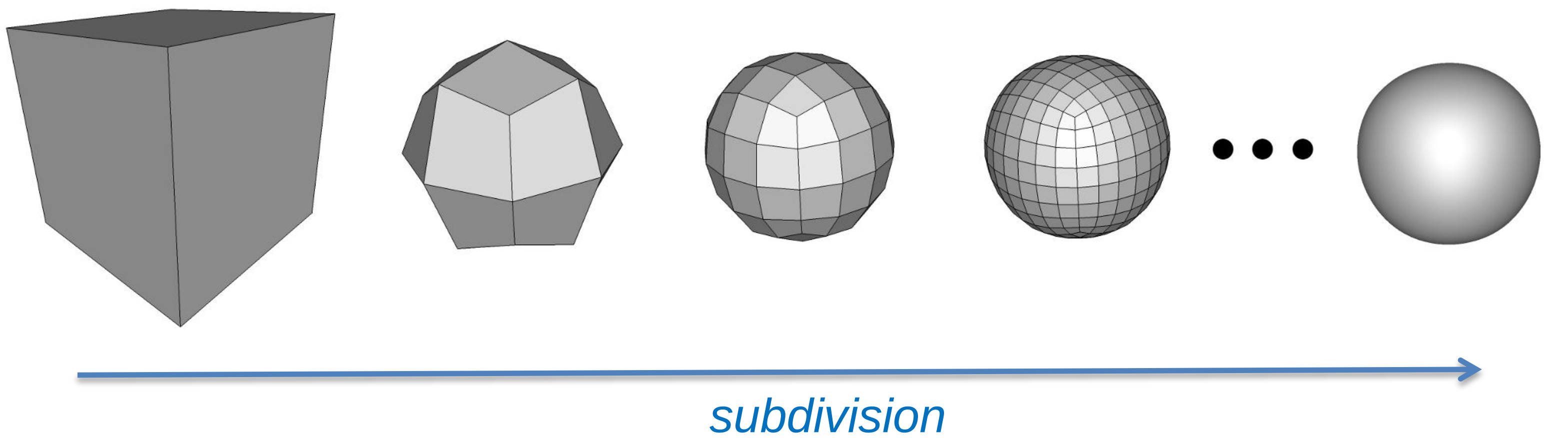
Domain Shader (DS)

- Evaluate surface given parametric u, v coordinates
- Interpolate attributes
- Apply displacements



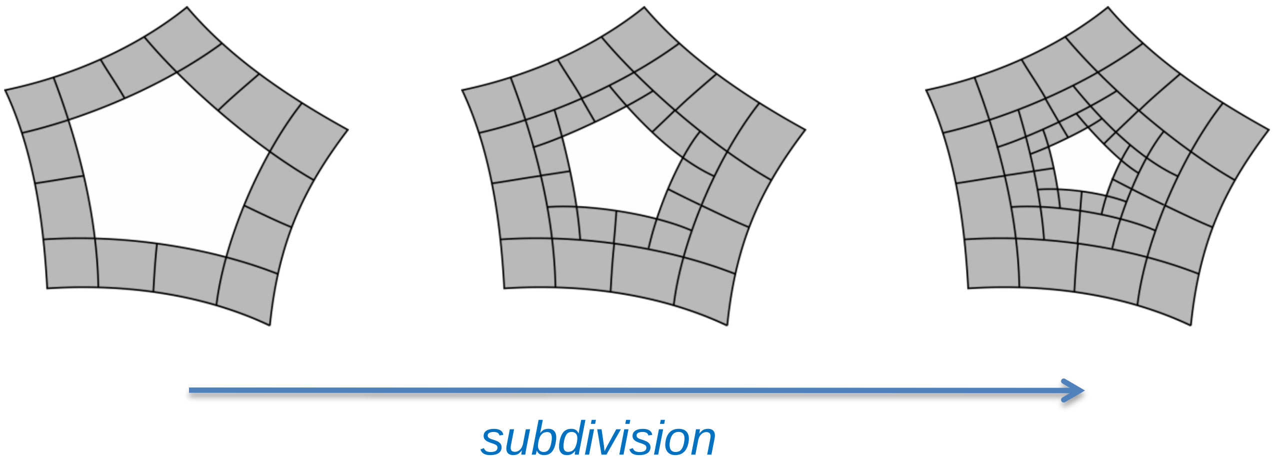
Subdivision Surfaces

Catmull, E. AND Clark, J. 1978,
Recursively generated B-spline surfaces on arbitrary topological meshes



- Already in the content creation pipeline
- Used extensively in film and game industries
- Coarse mesh input leads to smooth higher order surface

Problem: Infinite number of patches



- Does not easily fit hardware tessellation paradigm
- Stam, J. 1998,
  Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values
- Using exact evaluation possible, but expensive
  - Need two levels of subdivision to get started
  - Eigen basis function storage/evaluation costly

Approximation Schemes

- Loop, C. AND Schaefer, S. 2008,
  Approximating Catmull-Clark subdivision surfaces with bicubic patches

  Quads only, continuous geometry, smooth normal field
  25 control points per patch

- Ni, T., Yeo. Y.I., Miles, A, AND Peters, J. 2008,
  GPU smoothing of quad meshes

  Quads only smooth geometry and normal field
  24 control points per patch

- Myles, A, Ni, T., AND Peters, J. 2008,
  Fast Parallel construction of smooth surfaces from meshes with tri/quad/pent facets

  3, 4, or 5 sided faces, smooth geometry and normal field
  19, 25, and 31 control points per patch

- This paper

  3, 4 sided Gregory patches
  15, 20 control points per patch

Gregory Patches

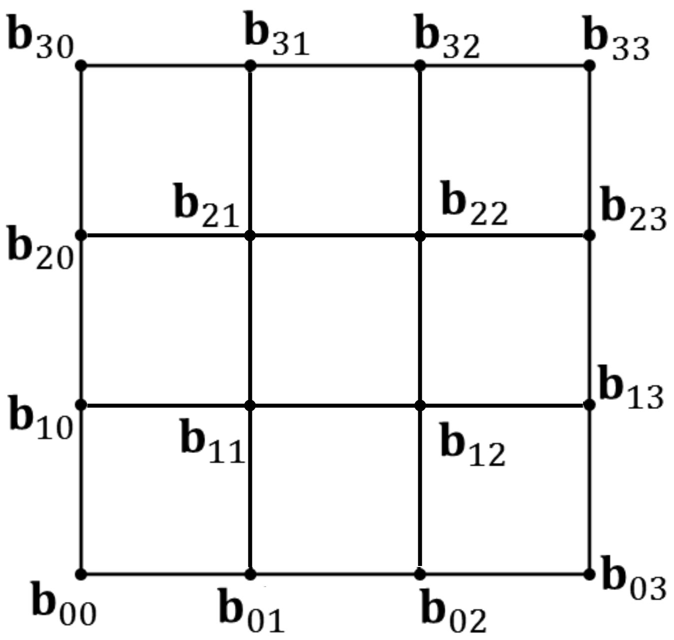
- Gregory, J. 1974,
  Smooth interpolation without twist constraints

  Introduced to solve subtle problem with incompatible mixed partial derivatives, or “twists” at patch corners in the regular setting

- Chiyokura, H. AND Kimura, F., 1983
  Design of solids with free-form surfaces

  Extended to irregular setting, introduced Bézier formulation

Bicubic Bézier Patch

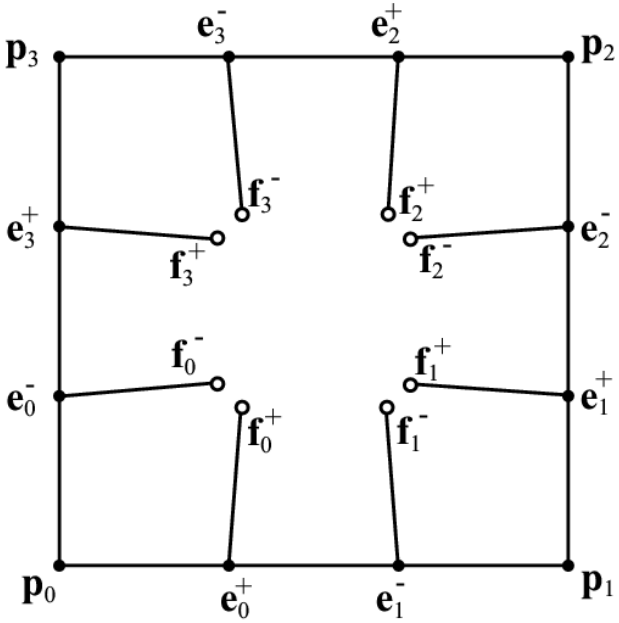


$$B(u,v) = \mathrm{b}^3(u) \cdot \mathbf{B} \cdot \mathrm{b}^3(v)$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02} & \mathbf{b}_{03} \\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12} & \mathbf{b}_{13} \\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22} & \mathbf{b}_{23} \\ \mathbf{b}_{30} & \mathbf{b}_{31} & \mathbf{b}_{32} & \mathbf{b}_{33} \end{bmatrix}$$

$$\mathrm{b}^3(u) = [(1-u)^3 \quad 3(1-u)^2 u \quad 3(1-u)u^2 \quad u^3\,]$$

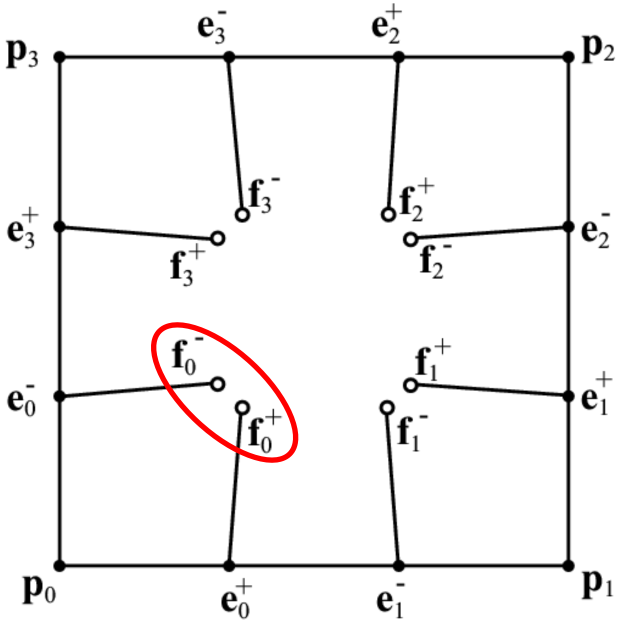
Gregory Quad Patch



$$Q(u,v) = \mathrm{b}^3(u)\ \cdot \mathbf{G}(u,v) \cdot \mathrm{b}^3(v)$$

$$\mathbf{G}(u,v) = \begin{bmatrix} \mathbf{p}_3 & \mathbf{e}_0^- & \mathbf{e}_3^+ & \mathbf{p}_2 \\ \mathbf{e}_0^+ & \mathbf{F}_0(u,v) & \mathbf{F}_3(u,v) & \mathbf{e}_3^- \\ \mathbf{e}_1^- & \mathbf{F}_1(u,v) & \mathbf{F}_2(u,v) & \mathbf{e}_2^+ \\ \mathbf{p}_0 & \mathbf{e}_1^+ & \mathbf{e}_2^- & \mathbf{p}_1 \end{bmatrix}$$

Gregory Quad Patch



$$Q(u,v) = \mathrm{b}^3(u) \cdot \mathbf{G}(u,v) \cdot \mathrm{b}^3(v)$$

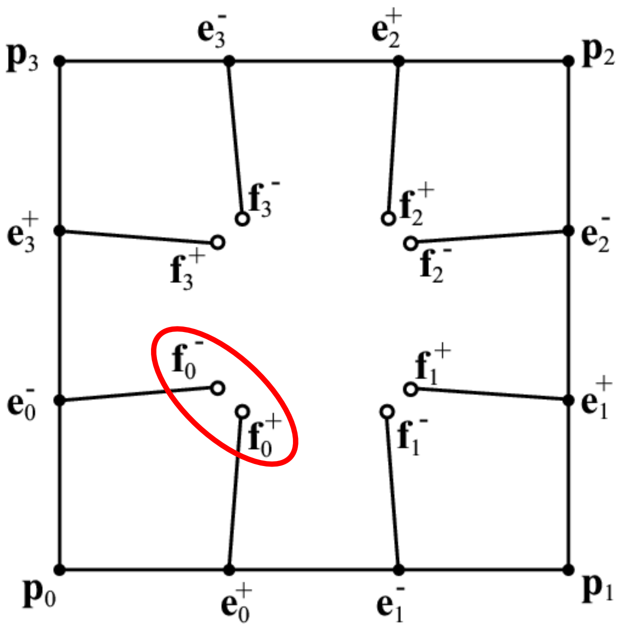
$$\mathbf{G}(u,v) = \begin{bmatrix} \mathbf{p}_3 & \mathbf{e}_0^- & \mathbf{e}_3^+ & \mathbf{p}_2 \\ \mathbf{e}_0^+ & \mathbf{F}_0(u,v) & \mathbf{F}_3(u,v) & \mathbf{e}_3^- \\ \mathbf{e}_1^- & \mathbf{F}_1(u,v) & \mathbf{F}_2(u,v) & \mathbf{e}_2^+ \\ \mathbf{p}_0 & \mathbf{e}_1^+ & \mathbf{e}_2^- & \mathbf{p}_1 \end{bmatrix}$$

$$\mathbf{F}_0(u,v) = \frac{u\,\mathbf{f}_0^+ + v\,\mathbf{f}_0^-}{u+v}$$

$\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3$ are similar

SIGGRAPHASIA2009

Gregory Quad Patch



$$Q(u,v) = \mathrm{b}^3(u) \cdot \mathbf{G}(u,v) \cdot \mathrm{b}^3(v)$$

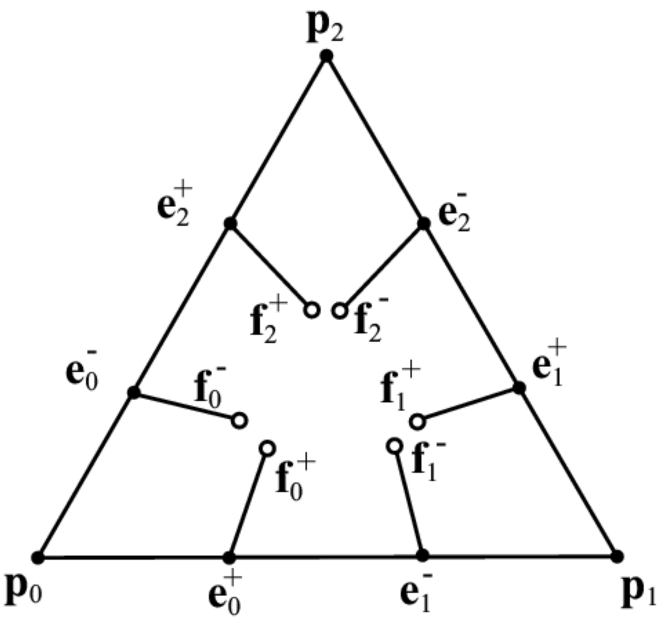
$$\mathbf{G}(u,v) = \begin{bmatrix} \mathbf{p}_3 & \mathbf{e}_0^- & \mathbf{e}_3^+ & \mathbf{p}_2 \\ \mathbf{e}_0^+ & \mathbf{F}_0(u,v) & \mathbf{F}_3(u,v) & \mathbf{e}_3^- \\ \mathbf{e}_1^- & \mathbf{F}_1(u,v) & \mathbf{F}_2(u,v) & \mathbf{e}_2^+ \\ \mathbf{p}_0 & \mathbf{e}_1^+ & \mathbf{e}_2^- & \mathbf{p}_1 \end{bmatrix}$$

$$\mathbf{F}_0(u,v) = \frac{u\,\mathbf{f}_0^+ + v\,\mathbf{f}_0^-}{u+v}$$

$\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3$ are similar

Note that $\mathbf{F}_0(0,0) = \mathbf{F}_1(1,0) = \mathbf{F}_2(1,1) = \mathbf{F}_3(0,1) = \frac{0}{0}$

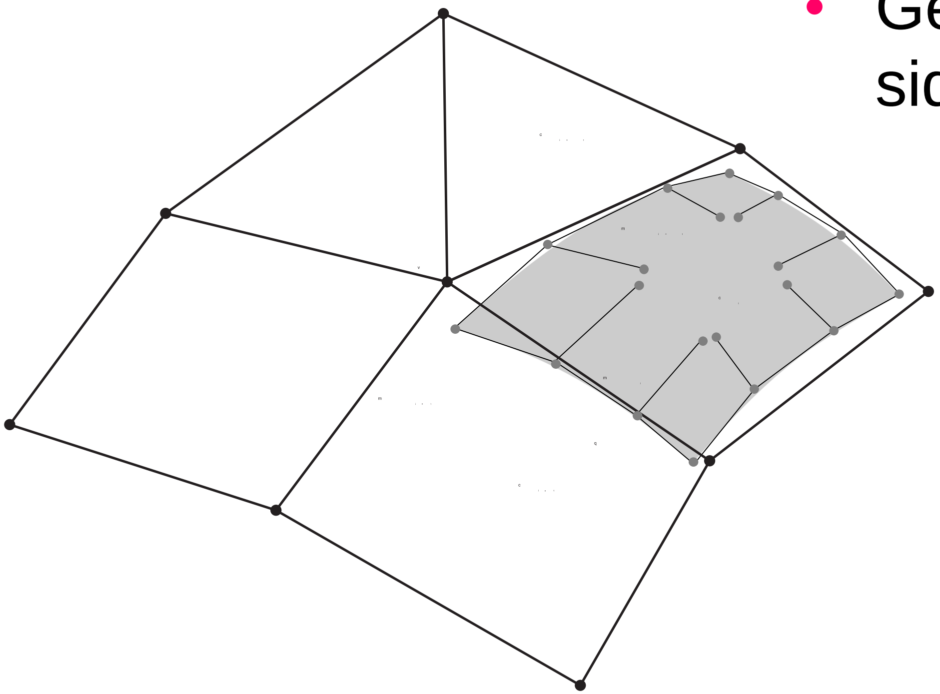
Gregory Triangle Patch



$$
\begin{aligned}
T(u,v,w) = \quad & u^3\mathbf{p}_0 + v^3\mathbf{p}_1 + w^3\mathbf{p}_2 \\
+ \quad & 3uv(u+v)(u\mathbf{e}_0^+ + v\mathbf{e}_1^-) \\
+ \quad & 3vw(v+w)(v\mathbf{e}_1^+ + w\mathbf{e}_2^-) \\
+ \quad & 3wu(w+u)(w\mathbf{e}_2^+ + u\mathbf{e}_0^-) \\
+ \quad & 12uvw(u\mathbf{F}_0 + v\mathbf{F}_1 + w\mathbf{F}_2)
\end{aligned}
$$

$$
\mathbf{F}_0(v,w) = \frac{w\,\mathbf{f}_0^- + v\,\mathbf{f}_0^+}{v+w}, \quad \mathbf{F}_1(u,w) = \frac{u\,\mathbf{f}_1^- + w\,\mathbf{f}_1^+}{w+u}, \quad \mathbf{F}_2(u,v) = \frac{v\,\mathbf{f}_2^- + u\,\mathbf{f}_2^+}{u+v}
$$

Patch Construction

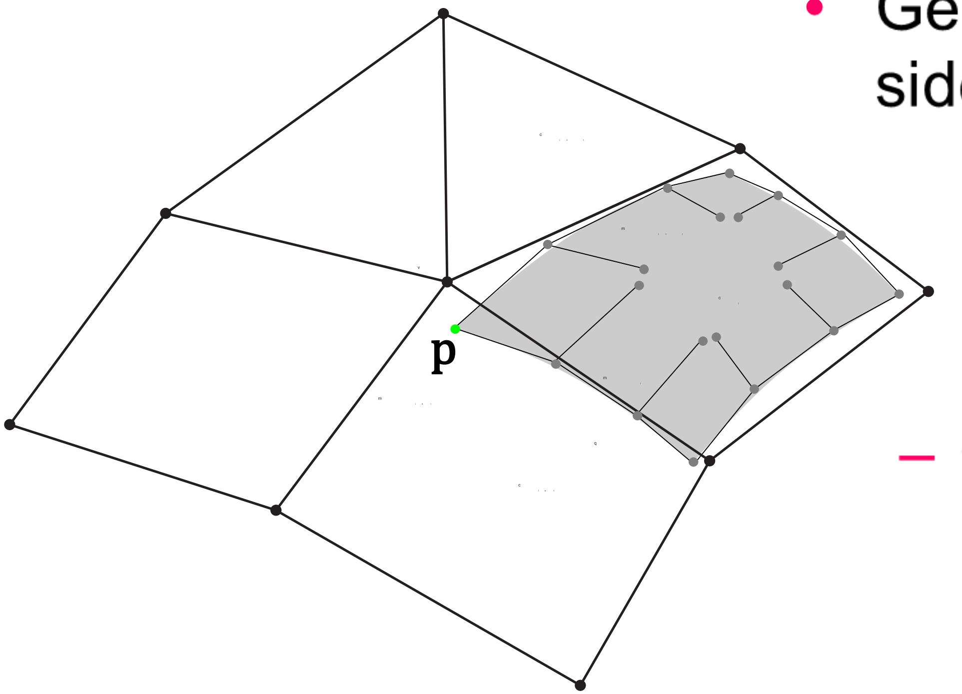


- General construction for 3 or 4 sided faces

Gregory patches in 1-1 correspondence with control mesh faces

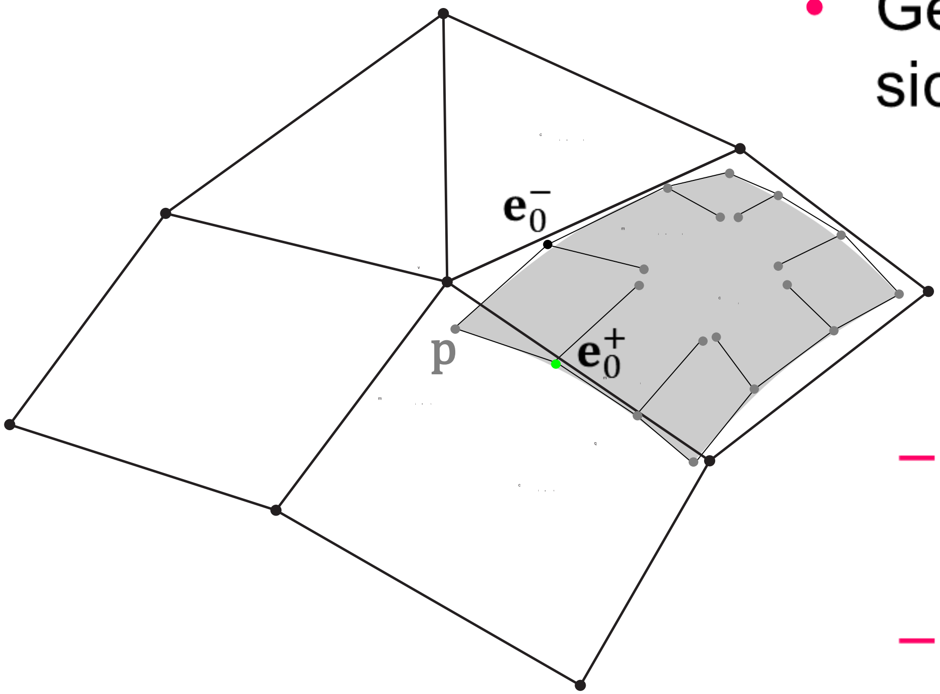
SIGGRAPHASIA2009

Patch Construction



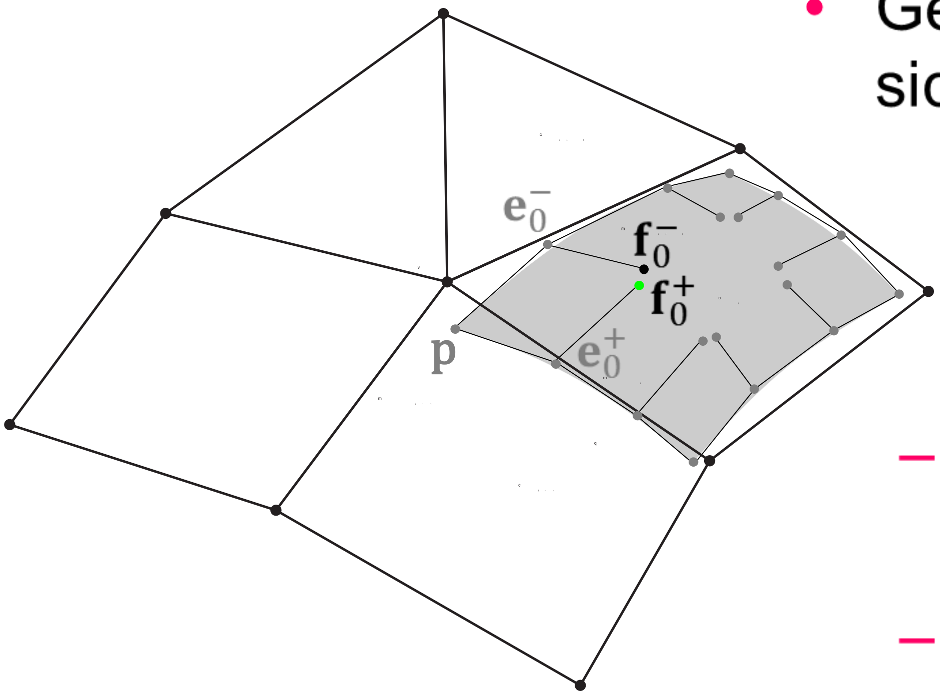
- General construction for 3 or 4 sided faces
  - Corner Points $\mathbf{p}$

Patch Construction



- General construction for 3 or 4 sided faces
  - Corner Points $\mathbf{p}$
  - Edge Points $\mathbf{e}_0^+, \mathbf{e}_0^-$

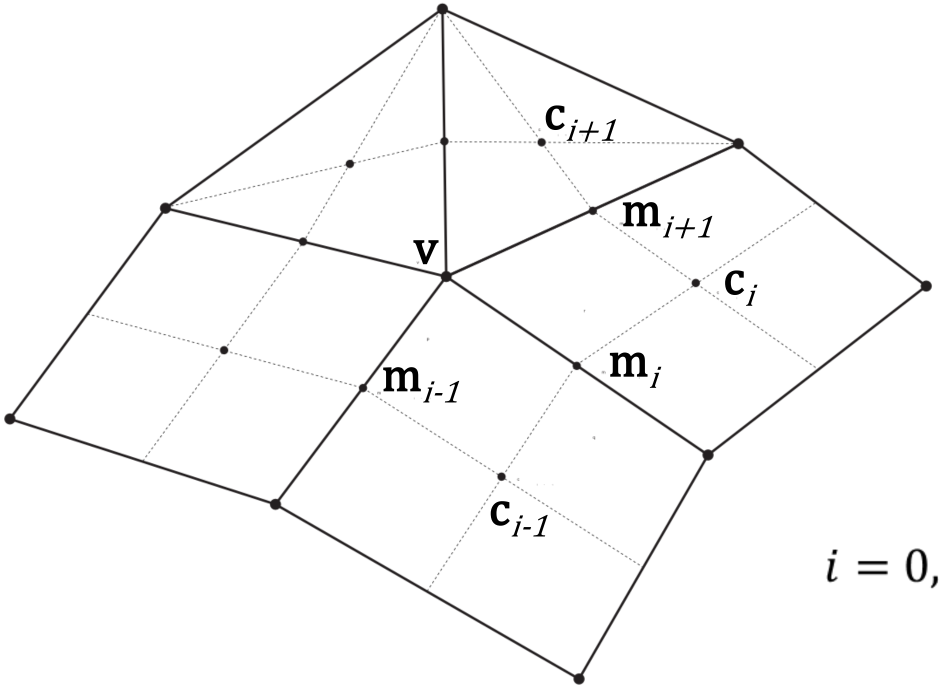
Patch Construction



- General construction for 3 or 4 sided faces
  - Corner Points $\mathbf{p}$
  - Edge Points $\mathbf{e}_0^+, \mathbf{e}_0^-$
  - Face Points $\mathbf{f}_0^+, \mathbf{f}_0^-$

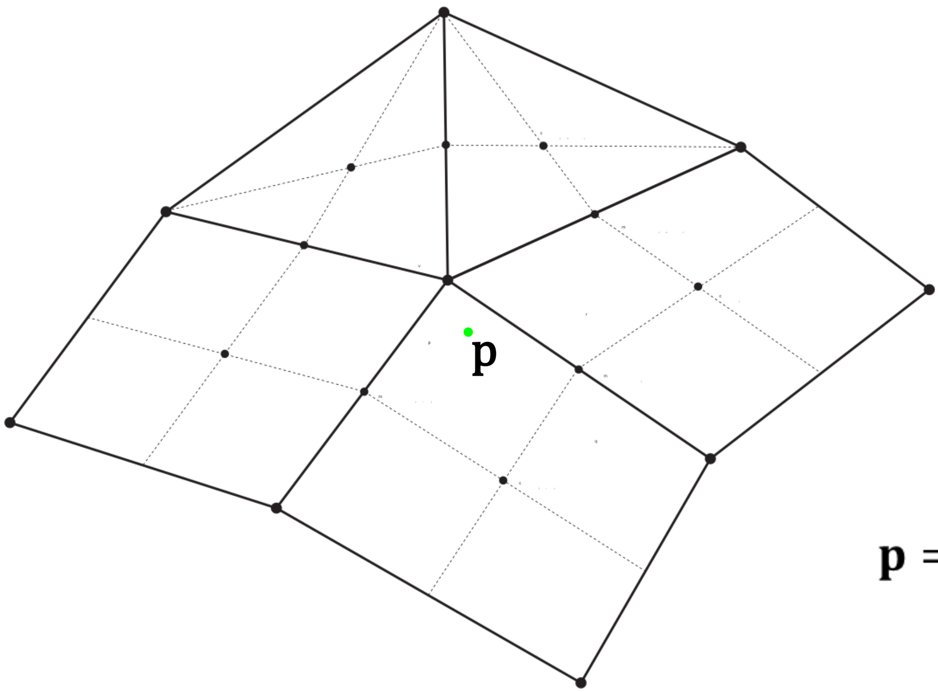
SIGGRAPHASIA2009

Edge Midpoints/Face Centroids



$i = 0, \dots n-1$ where n is the valence of $\mathbf{v}$

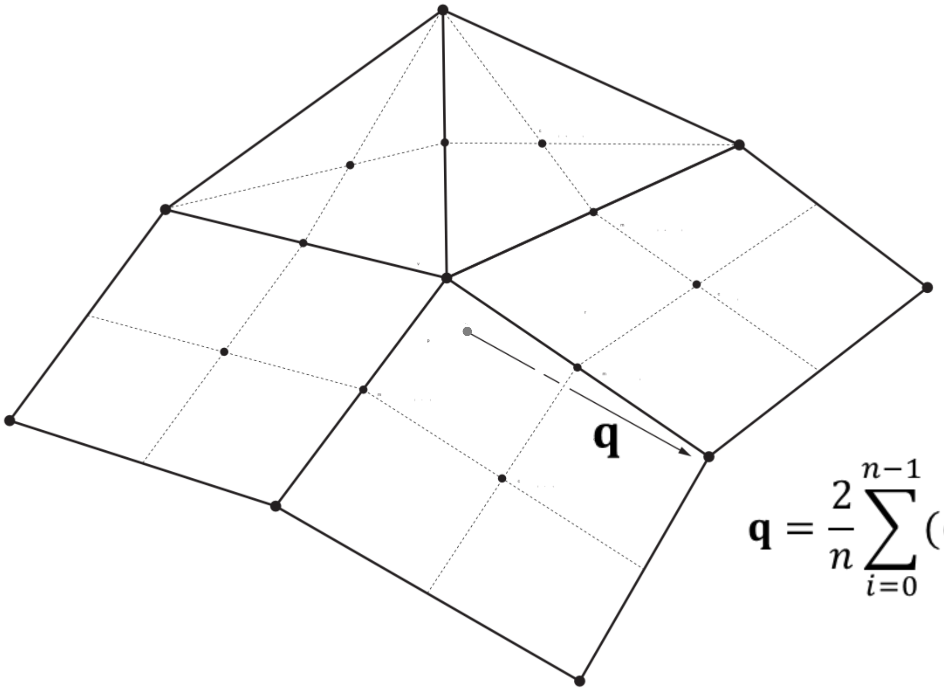
Corner Point



$$\mathbf{p} = \frac{n-3}{n+5}\mathbf{v} + \frac{4}{n(n+5)}\sum_{i=0}^{n-1}(\mathbf{m}_i + \mathbf{c}_i)$$

Interpolate limit position of Catmull-Clark Surface

Edge Points

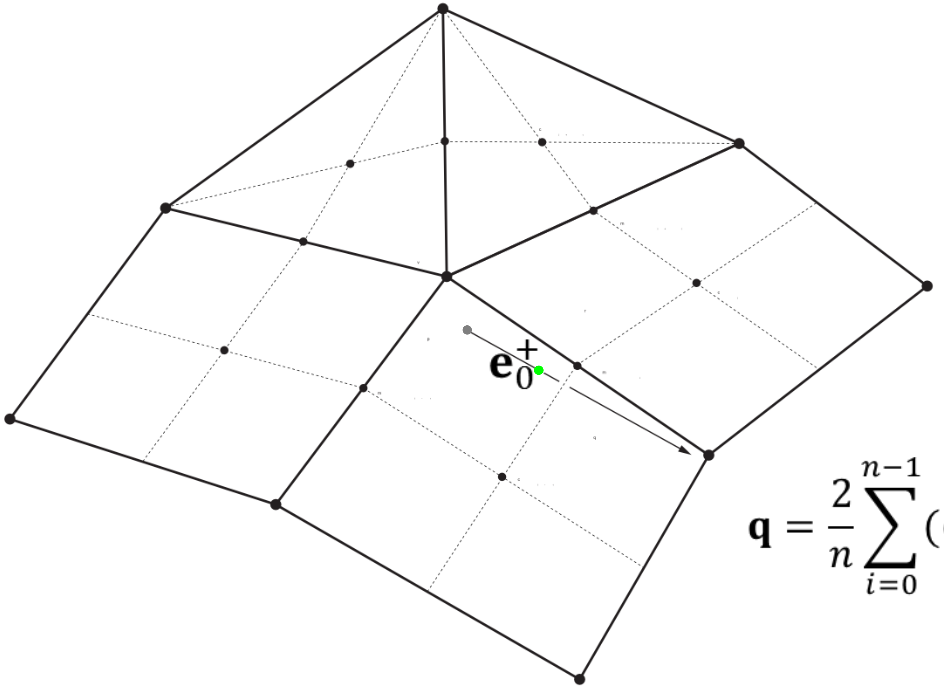


$$\mathbf{q} = \frac{2}{n}\sum_{i=0}^{n-1}\left(\left(1-\sigma\cos\left(\frac{\pi}{n}\right)\right)\cos\left(\frac{2\pi i}{n}\right)\mathbf{m}_i + 2\sigma\cos\left(\frac{2\pi i+\pi}{n}\right)\mathbf{c}_i\right)$$

$$\sigma = \left(4+\cos^2\left(\frac{\pi}{n}\right)\right)^{-1/2}$$

Interpolate limit tangent of Catmull-Clark Surface

Edge Points



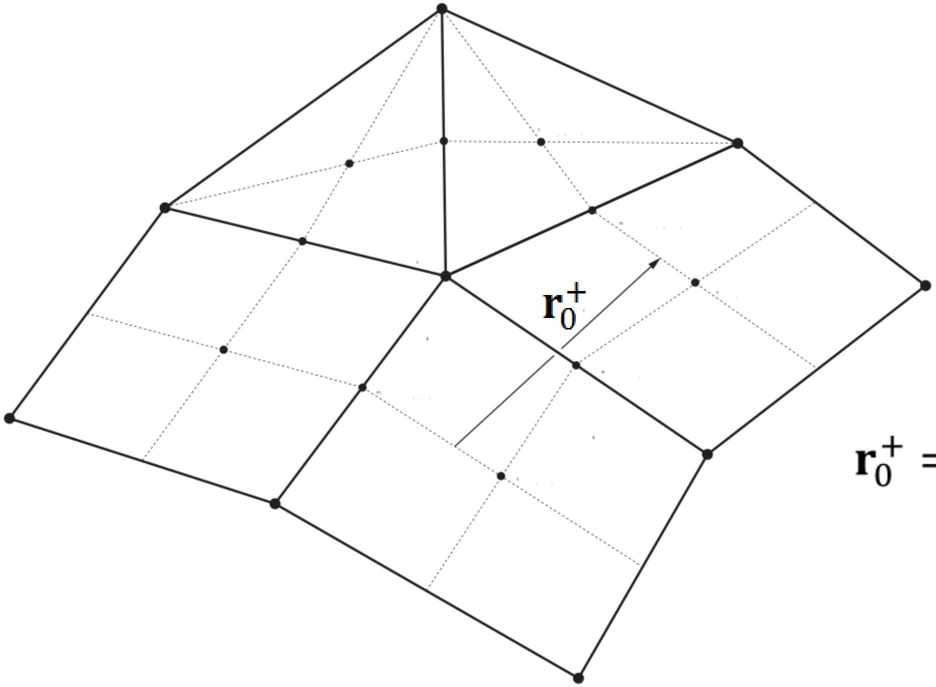
$$\mathbf{q} = \frac{2}{n}\sum_{i=0}^{n-1}\left(\left(1-\sigma\cos\left(\frac{\pi}{n}\right)\right)\cos\left(\frac{2\pi i}{n}\right)\mathbf{m}_i + 2\sigma\cos\left(\frac{2\pi i+\pi}{n}\right)\mathbf{c}_i\right)$$

$$\sigma = \left(4+\cos^2\left(\frac{\pi}{n}\right)\right)^{-1/2}$$

$$\mathbf{e}_0^+ = \mathbf{p} + \frac{2}{3}\lambda\,\mathbf{q}$$

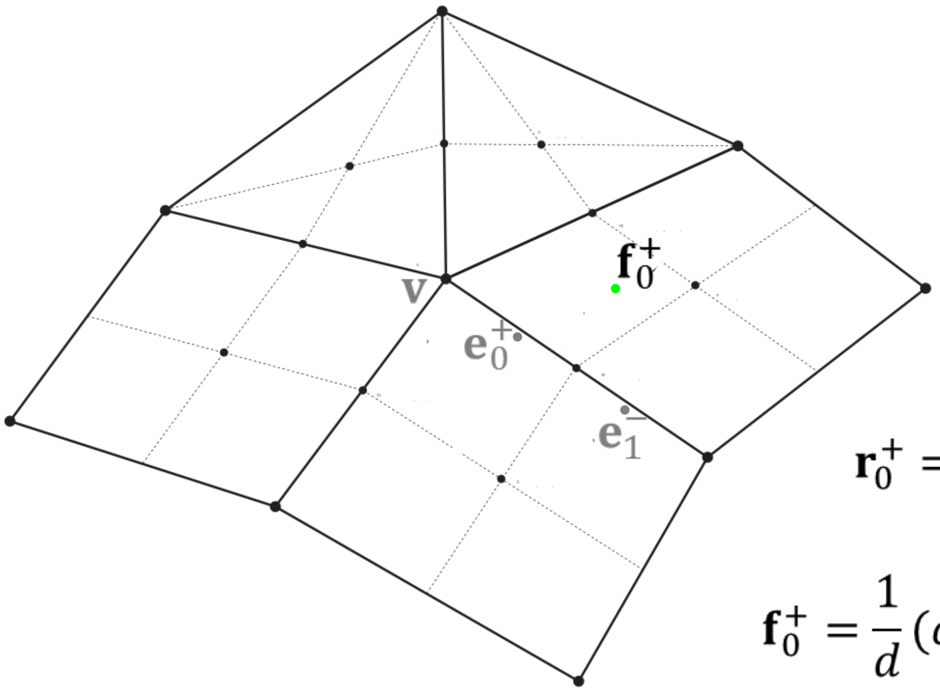
Interpolate limit tangent of Catmull-Clark Surface

Face Points



$$\mathbf{r}_0^+ = \frac{1}{3}(\mathbf{m}_{i+1} - \mathbf{m}_{i-1}) + \frac{2}{3}(\mathbf{c}_i - \mathbf{c}_{i-1})$$

Face Points



$$\mathbf{r}_0^+ = \frac{1}{3}(\mathbf{m}_{i+1} - \mathbf{m}_{i-1}) + \frac{2}{3}(\mathbf{c}_i - \mathbf{c}_{i-1})$$

$$\mathbf{f}_0^+ = \frac{1}{d}(c_1\mathbf{v} + (d - 2c_0 - c_1)\mathbf{e}_0^+ + 2c_0\mathbf{e}_1^- + \mathbf{r}_0^+)$$

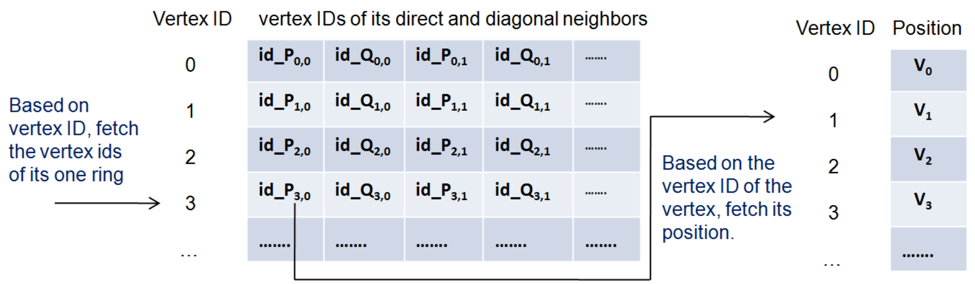
$$d = \begin{cases} 3 \; quad \\ 4 \; triangle \end{cases}$$

Two GPU Implementations

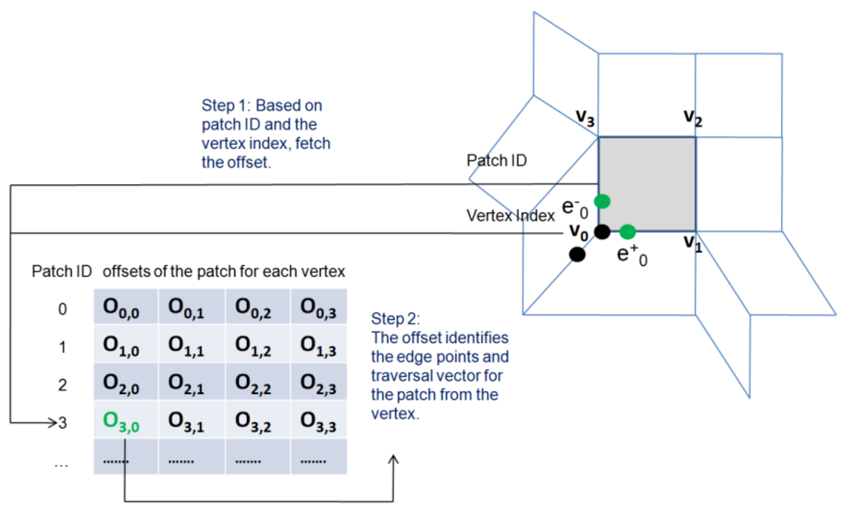
- Vertex/Hull Shaders
  - Exploit vertex-centric nature of computations
  - Avoid redundant computations
- Hull Shader Stencil Approach
  - Map patch construction to hull shader exclusively
  - Frees vertex shader for other tasks
- ‘Best’ implementation will depend on
  - LOD, low V/H better, high HSS better
  - Hardware vendor
  - Application

Vertex/Hull Shader Approach

- Compute $\mathbf{p}, \mathbf{e}_i^+, \mathbf{e}_i^-, \mathbf{r}_i^+, \mathbf{r}_i^-, i = 0, \ldots, n-1,$ in vertex shader



- Pass 4 (or 3) point primitive to hull shader



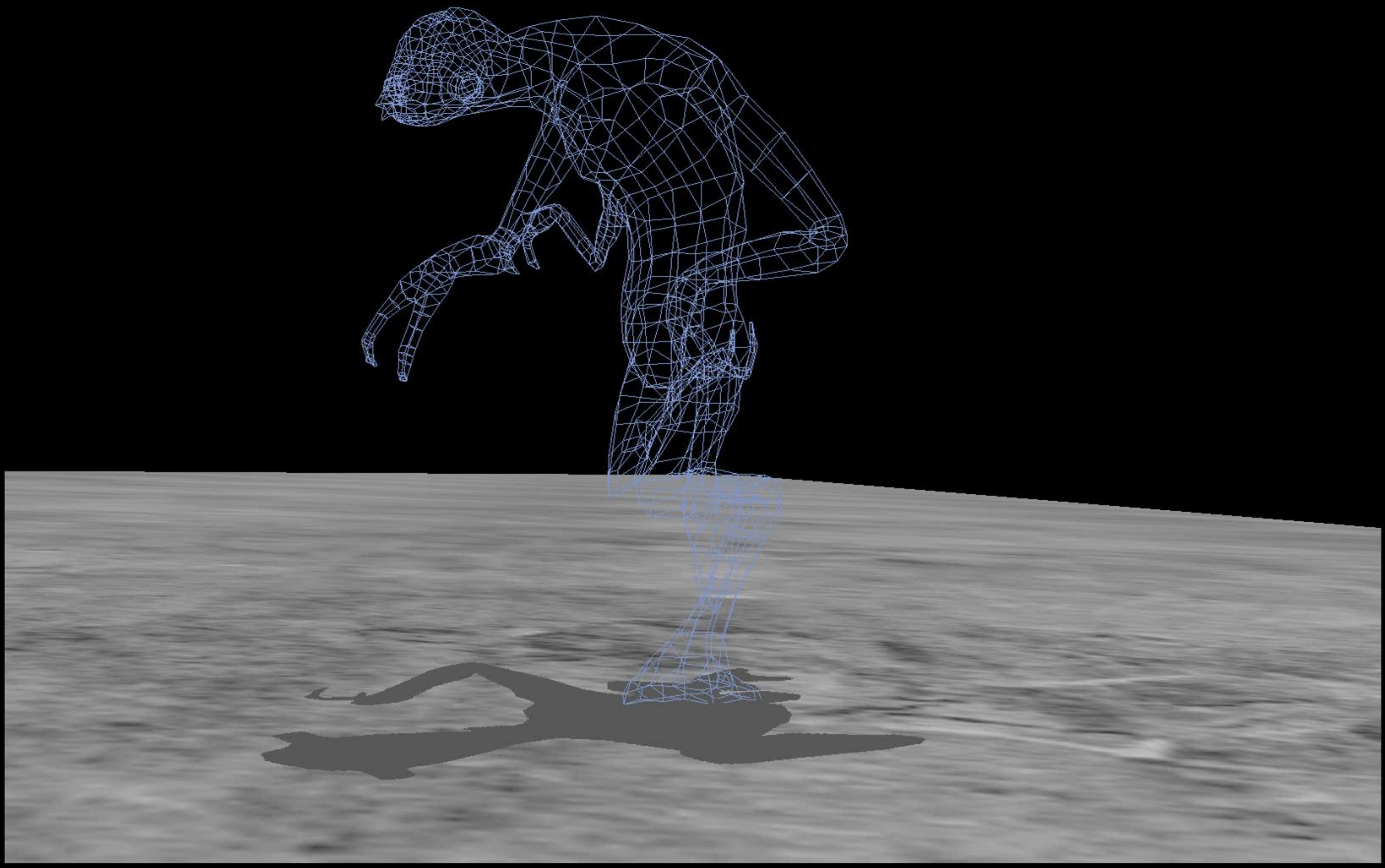
SIGGRAPHASIA2009

Hull Shader Stencil Approach

- Sort mesh into patch *connectivity types*
  - permutation of a face 1-ring neighborhood
- Each connectivity type determines a *weight matrix*
  - store these matrices in a texture
  - hull shader computes patch as matrix/vector product
- Advantages
  - simple code, low register/shared memory pressure
  - fits tessellator pipeline well
- Disadvantages
  - sparse matrix, many unnecessary fetches/products
  - redundant computations – corner/edges points

Domain Shader

- Evaluates patches at u, v values generated by tessellator
- Two cases: 3 and 4 sided patches
- In both cases
  - evaluate the $\mathbf{F}_i$ (rational function of u and v)
  - use DeCasteljau's algorithm for position/normal
    - normal not 'correct', but continuous on edges
- Corner degeneracy $(\frac{0}{0})$
  - use conditional assignment
  - low impact on SIMD efficiency



Results

SIGGRAPHASIA2009

Conclusions

- Simple geometry construction
  - Handling boundaries in paper
- Lowest fetch overhead for domain shader
  - 20 control points for quads, 15 for triangles
  - Critical performance bottleneck
- Error to ‘true’ Catmull-Clark surface small
  - See paper
  - “artist intent” problem solved by migration to tool chain

Thank You