Advanced Computer Graphics
CS 563: *Curves and Curved Surfaces II*

Xin Wang

Computer Science Dept.

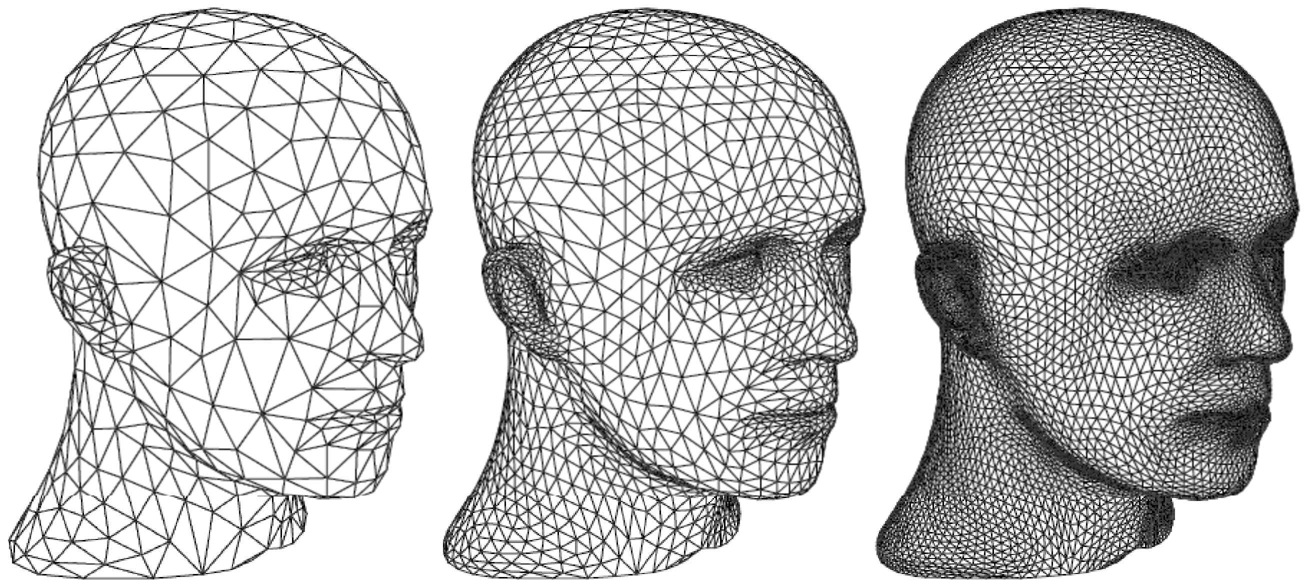
Worcester Polytechnic Institute (WPI)

Continue Curves

- Parametric Curves
- Parametric Curved Surfaces
- Implicit Surfaces
- Subdivision Curves
- Subdivision Surfaces
- Efficient Tessellation

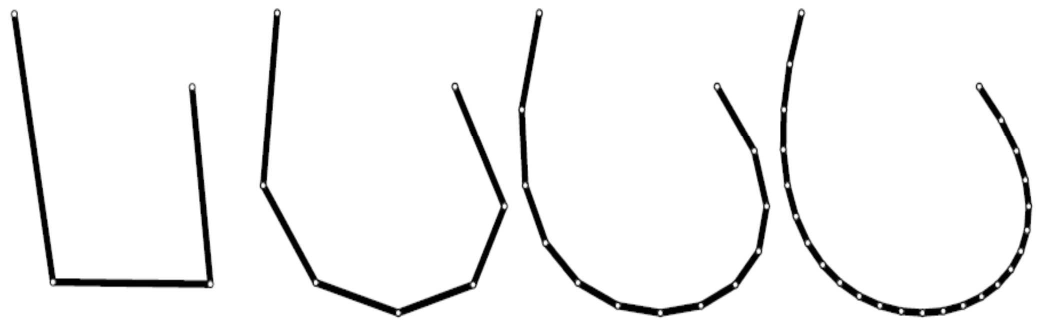
Subdivision Curves

- What is Subdivision?
- Define a smooth surface as the limit of a sequence of successive refinements



Subdivision Curves

- Why uses Subdivision?
  - Arbitrary topology
  - Scalability, LOD
  - Uniformity
  - Numerical quality
  - Code simplicity
- Subdivision Curves are best explained by examples that uses ***corner cutting***



Subdivision Curves

- How are the new points determined?
  - Efficiency -> small number of floating point operations
  - Compact support -> region should be small and finite
  - Local definition -> new point should “not far away”
  - Affine invariance -> shape follow the same transformation as original set of points
  - Simplicity -> process should be simple
  - Continuity -> curves and surface smooth

Subdivision Surfaces

- Loop subdivision
- Modified Butterfly subdivision
- $\sqrt{3}$-Subdivision
- Catmull-Clark subdivision
- Piecewise Smooth subdivision
- Displaced subdivision
- Normal, Texture and Color interpolation

Loop Subdivision

- First subdivision scheme for triangles

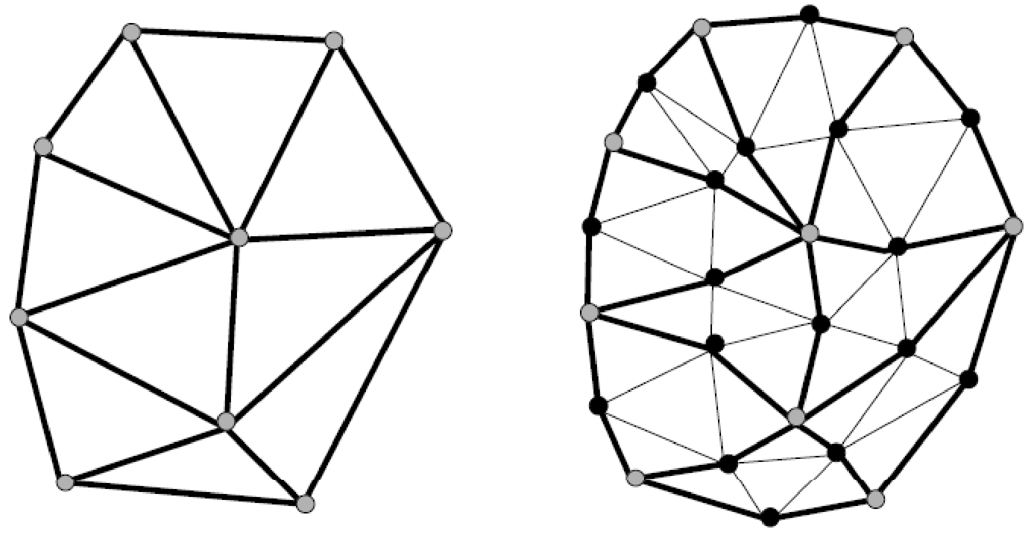


Figure 3.1: Refinement of a triangular mesh. New vertices are shown as black dots. Each edge of the the control mesh is split into two, and new vertices are reconnected to form 4 new triangles, replacing each triangle of the mesh.

Loop Subdivision

- Subdivision rules
  - A generalized three-directional quartic box spline

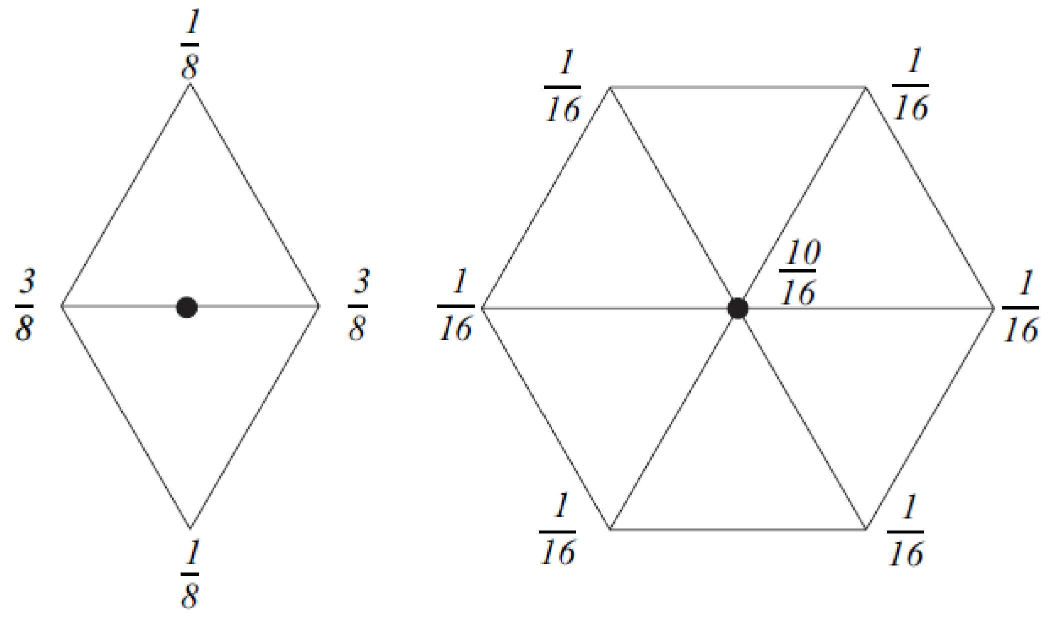
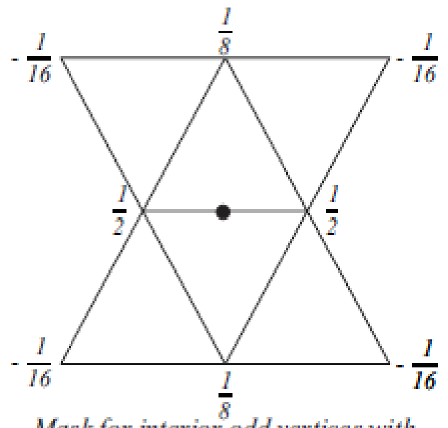


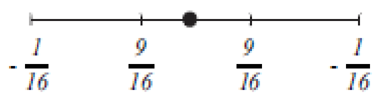
Figure 3.2: Subdivision coefficients for a three directional box spline.

Modified Butterfly Subdivision

- Subdivision rules
  - Regular setting
  - Semi-regular setting
  - Irregular setting
  - Boundaries

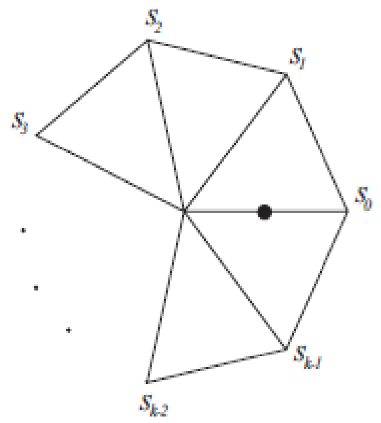


Mask for interior odd vertices with regular neighbors



Mask for crease and boundary vertices

a. Masks for odd vertices



b. Mask for odd vertices adjacent to an extraordinary vertex

Figure 3.16: Modified Butterfly subdivision. The coefficients s_i are $\frac{1}{k}\left(\frac{1}{4}+\cos\frac{2i\pi}{k}+\frac{1}{2}\cos\frac{4i\pi}{k}\right)$ for $k>5$. For $k=3$, $s_0=\frac{5}{12}$, $s_{1,2}=-\frac{1}{12}$; for $k=4$, $s_0=\frac{3}{8}$, $s_2=-\frac{1}{8}$, $s_{1,3}=0$.

$\sqrt{3}$-Subdivision

- Loop's and MB schemes split each triangle into four new ones
- Kobbelt's $\sqrt{3}$-schema creates only three new triangles per subdivision step

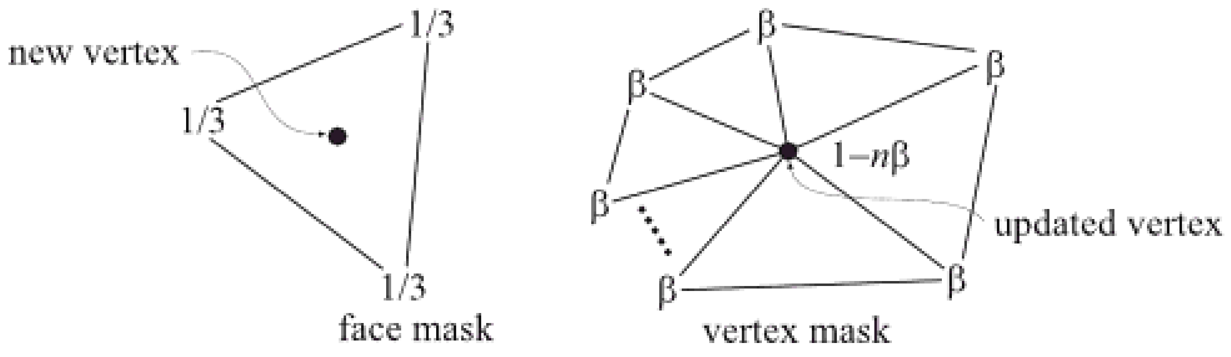


Figure 13.45. The masks for the $\sqrt{3}$-subdivision scheme. As can be seen, the face mask gives minimal support, since it uses only the three vertices of the triangle. The vertex mask uses all the vertices in the ring, called the 1-ring, around the vertex.

Catmull-Clark subdivision

- Handle polygonal meshes
  - Catmull-Clark and Doo-Sabin

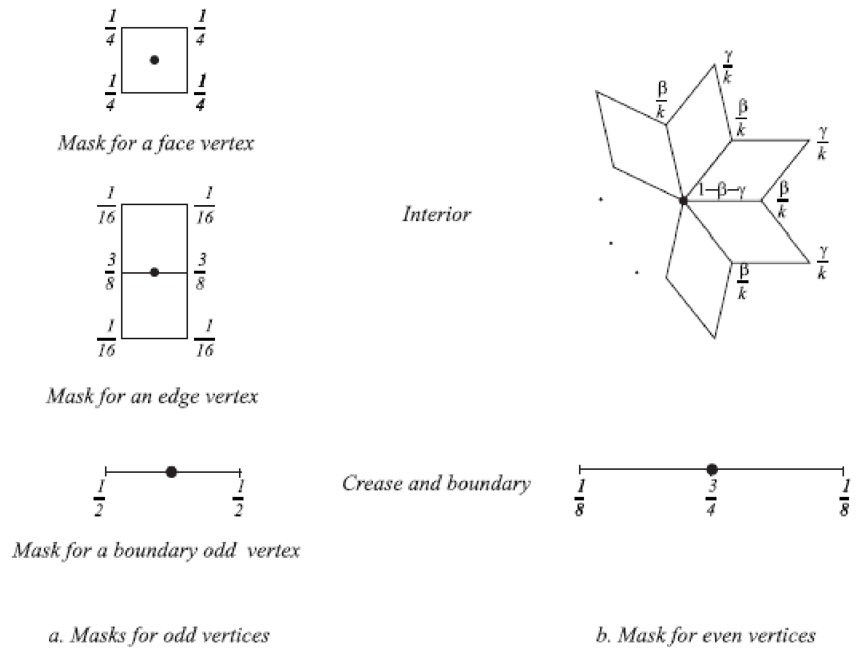


Figure 3.17: Catmull-Clark subdivision. Catmull and Clark [2] suggest the following coefficients for rules at extraordinary vertices: $\beta = \frac{3}{2k}$ and $\gamma = \frac{1}{4k}$

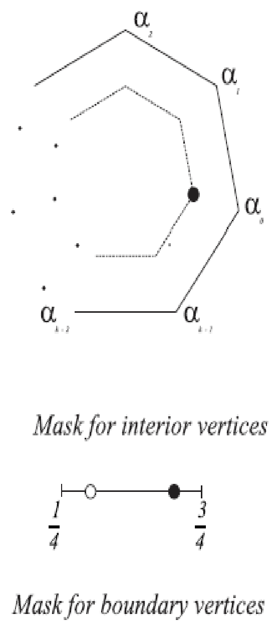
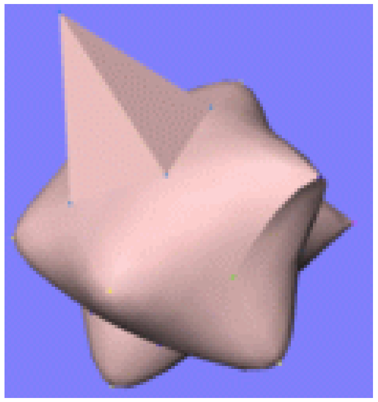

α_2 α_1 α_0 α_{k-1} α_{k-2}
Mask for interior vertices
1/4 3/4
Mask for boundary vertices

Figure 3.20: The Doo-Sabin subdivision. The coefficients are defined by the formulas $\alpha_0 = 1/4 + 5/4k$ and $\alpha_i = (3 + 2\cos(2i\pi/k))/4k$, for $i = 1 \ldots k-1$

Piecewise Smooth Subdivision

- To improved the detail of darts, corners and creases



- Vertices are classified into:
  - Smooth s = 0
  - Dart s = 1
  - Crease s = 2
  - Corner s > 2

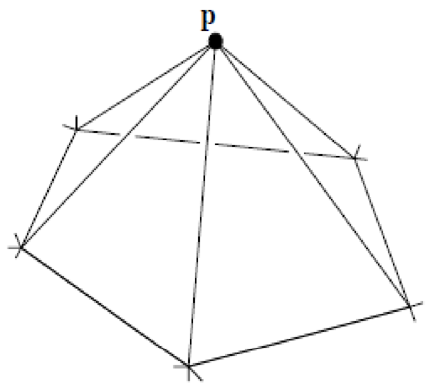


Figure 3: A piecewise linear hat function for a vertex **p** of valence 5.

Displaced Subdivision

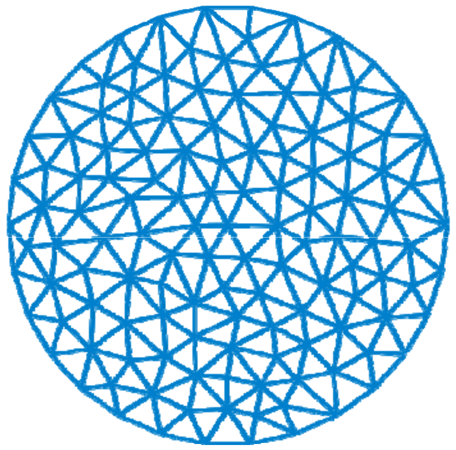
- To add additional geometric detail to a tessellated surface, we can use *displacement mapping*
- With this technique, a displacement map is stored, which is much like a texture map that stores a height value per texel instead of a color
- As with texture mapping, we can assign a texture coordinate to each corner of the patch that allows us to position the displacement map onto the surface

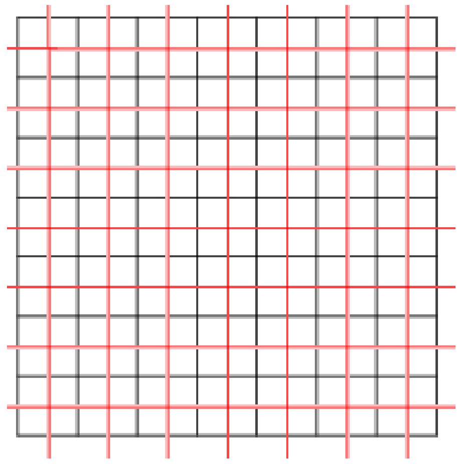
Normal, Texture and Color Interpolation

- Compute normal
  - Limit normals for Loop’s and MB
  - Approximating technique for nonrmals of Catmull-Clark
- Compute texture and color

Tessellation

- What is Tessellation?
  - Create a triangle mesh representation of the surface





- Why is Tessellation?
  - Efficient

Tessellation

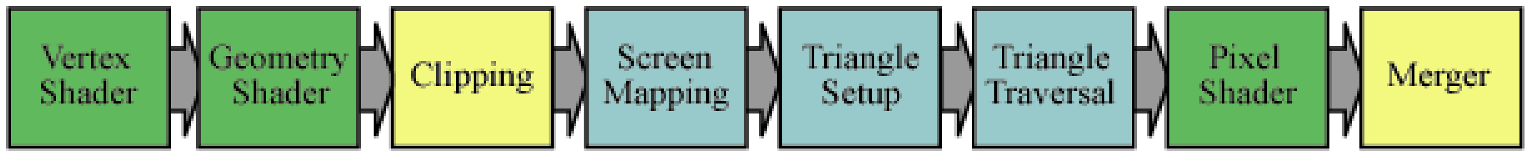
- *Tessellation* is the process of taking a complex surface (like a bicubic patch) and approximating it with a set of simpler surfaces (like triangles)
- In computer graphics, there are a lot of different types of complex surfaces one might want to tessellate, such as:
  - Parametric surfaces (such as Bezier surfaces)
  - Displacement mapped surfaces
  - Subdivision surfaces
  - Fractals
  - Procedural models
  - Implicit surfaces

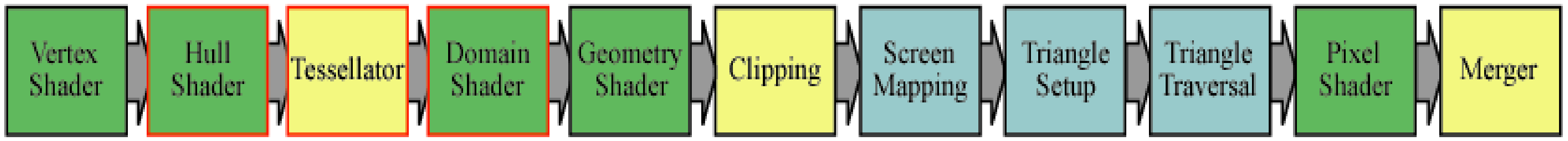
Efficient Tessellation

- Hardware Tessellation pipeline
- Fractional Tessellation
- Vertex + Evaluation Shader
- Adaptive Tessellation
- Catmull-Clark Surfaces with Hardware Tessellation

Hardware Tessellation Pipeline

- Change rendering pipeline





Fractional Tessellation

- To obtain smoother level of detail for parametric surfaces

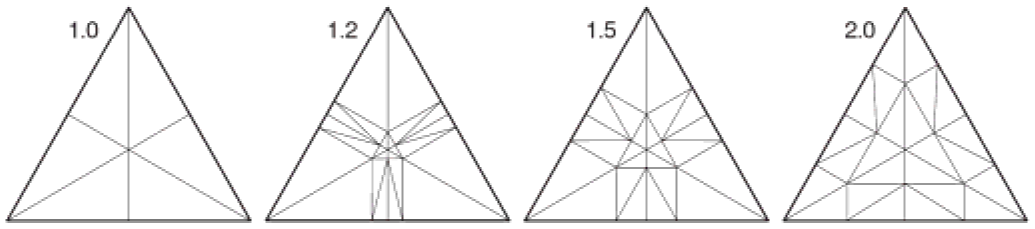


Figure 13.56. Fractional tessellation of a triangle, with tessellation factors shown. Note that the tessellation factors may not correspond exactly to those produced by actual tessellation hardware. (*After Tatarchuk [1252].*)

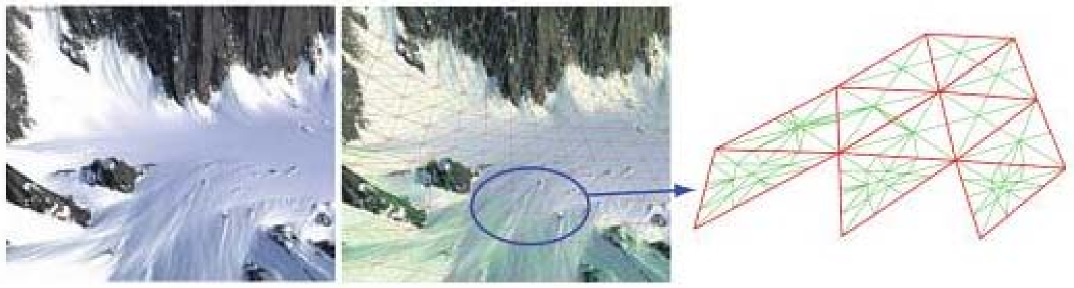


Figure 13.57. Displaced terrain rendering using adaptive fractional tessellation. As can be seen in the zoomed-in mesh to the right, independent fractional tessellation rates are used for the edges of the red triangles, which gives us adaptive tessellation. *(Images courtesy of Game Computing Applications Group, Advanced Micro Devices, Inc.)*

Vertex + Evaluation Shader

- Compute each vertex position which located on a curved surface
- The hardware tessellator simply forwards the parameteric coordinates to the vertex shader

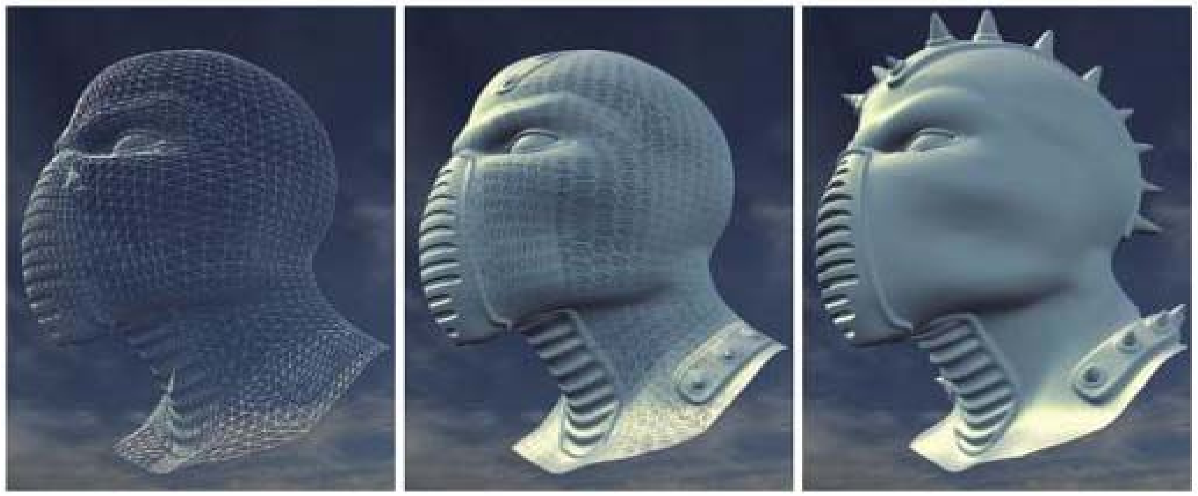


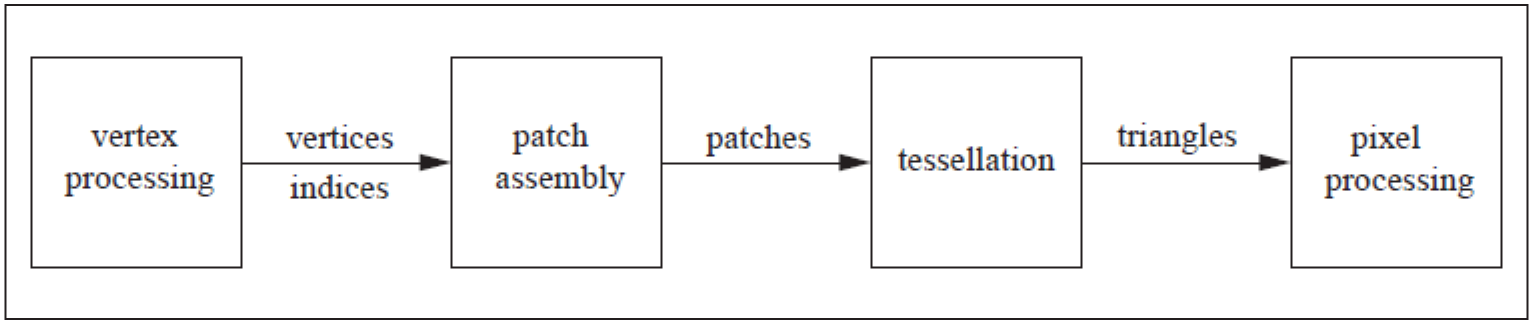
Figure 13.58. Left: wireframe of a ninja model. Middle: The model to the left has now been tessellated with fractional rates on the triangle sides. Right: After tessellation, each vertex has been displaced. *(Images courtesy Game Computing Applications Group, Advanced Micro Devices, Inc.)*

Adaptive Tessellation

- Very often, the goal of a tessellation is to provide the fewest triangles necessary to accurately represent the original surface
- For a curved surface, this means that we want more triangles in areas where the curvature is high, and fewer triangles in areas where the curvature is low
- We may also want more triangles in areas that are closer to the camera, and fewer farther away
- *Adaptive tessellation* schemes are designed to address these requirements

Catmull-Clark Surfaces with Hardware Tessellation

- Combine together



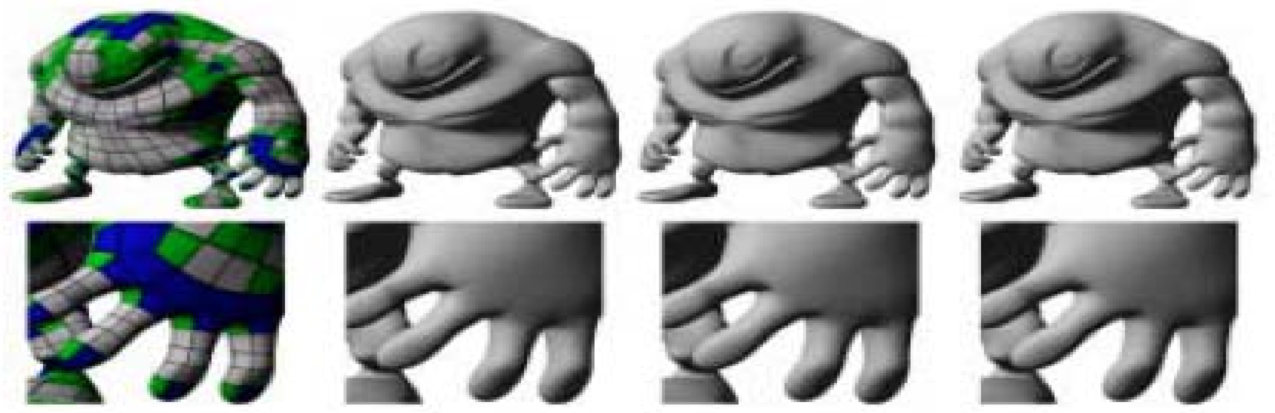


Fig. 10. An example mesh (top) and a zoomed in region of a complex patch structure (bottom). From left to right: Catmull-Clark patch structure, Geometry patch approximation, Geometry/Tangent patch approximation and Catmull-Clark limit mesh.

References

- *Real-Time Rendering, 3rd Edition, Akenine-Möller, Haines, Hoffman*
- *Subdivision for Modeling and Animation, SIGGRAPH 98 course notes, Peter Schröder, Denis Zorin*
- *http://www.multires.caltech.edu/teaching/courses/subdivision/*