Johns Hopkins
Engineering for Professionals

605.767 Applied Computer Graphics

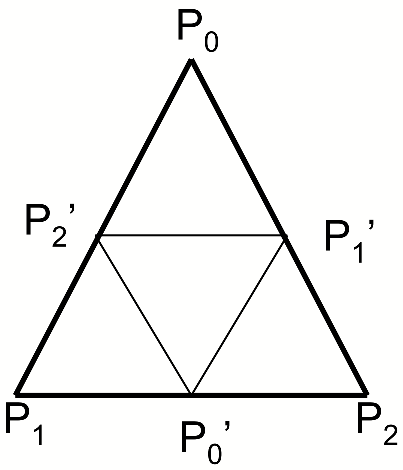
Brian Russin

Johns Hopkins
Whiting School
of Engineering

Module 7C
Subdivision

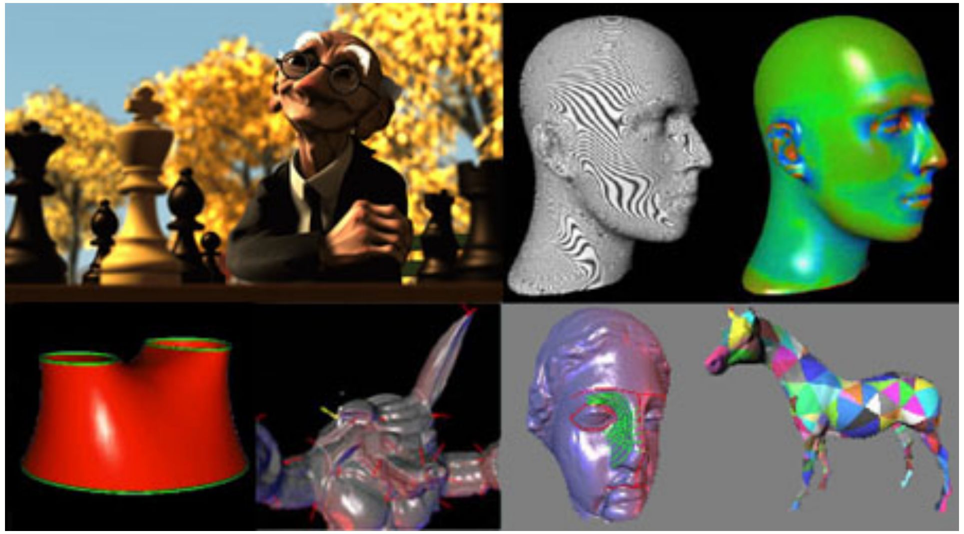
Surface Refinement and Recursive Subdivision

- Can create a sphere by recursive subdivision
  - Begin with a particular shape
  - Repeatedly bisect the faces
  - “Normalize” vertices to specified radius
    - Places the vertices on the surface of the sphere
- Sample lab code contains recursive subdivision of an icosahedron
  - Specify number of levels of subdivision



Bisect the edges of original triangle
Forms 4 new triangles (replacing original)

Subdivision Surfaces



Geri’s Game (1997)
Pixar Animation Studios
Catmull-Clark Subdivision

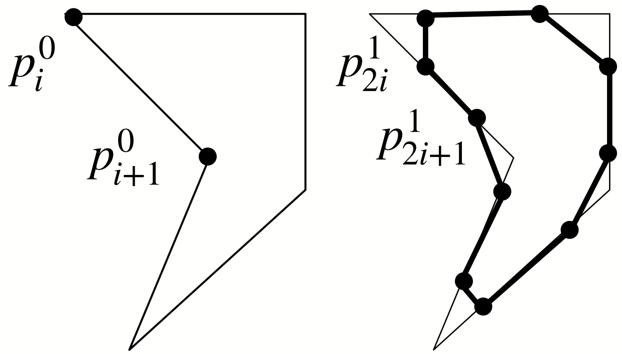
http://mrl.nyu.edu/publications/subdiv-course2000

Subdivision Curves and Surfaces

- Relatively new
  - Introduced by Catmull and Clark in 1978
  - Much research in late 1990s
- Integrated into many commercial modeling and rendering systems
  - Pixar’s Renderman, Alias’s Maya3D, Newtek’s Lightwave 3D
  - Used to create characters in Disney/Pixar productions: Geri's Game, Bug's Life, Toy Story 2
- Bridges the gap between discrete surfaces (meshes) and continuous surfaces (parametric)
- Algorithmic definition of curves and surfaces
  - Unlike parametric curves and surfaces
  - Constructed through recursive splitting and averaging
- Supports level of detail control
  - By controlling the level of subdivision

Subdivision Curves

- Example uses corner cutting
  - Insert new vertices along edges and remove existing vertex
  - Curve appears smoother with each successive subdivision
    - Continue subdivision until cannot see a visual difference
    - **Limit curve** is the curve that results with infinite subdivision
- Method illustrated below is Chaikin's subdivision
  - Create new vertices ¼ along edge from each vertex
- An **approximating** subdivision method
  - The limit curve does not generally contain the vertices of the original polygon



$$p_{2i}^{k+1} = \frac{3}{4} p_i^k + \frac{1}{4} p_{i+1}^k$$

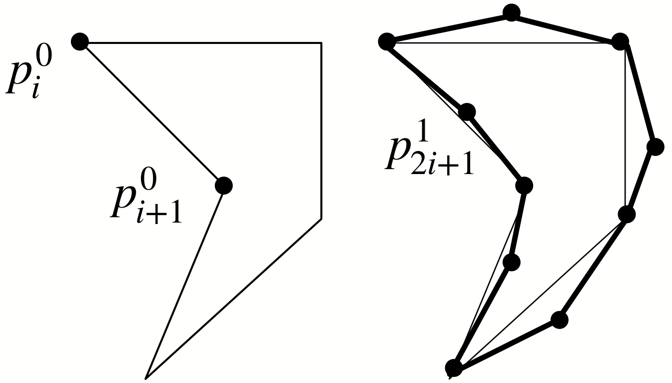
$$p_{2i+1}^{k+1} = \frac{1}{4} p_i^k + \frac{3}{4} p_{i+1}^k$$

i is the vertex index
k is the subdivision level

One subdivision – See Figure 17.38 (13.29 in 3rd Edition) for illustration of successive iterations

Interpolating Subdivision Method for Curves

- An interpolating subdivision method keeps all the points from the previous subdivision
  - The curve interpolates the original polygon
- 4 point subdivision method below keeps the original vertex and adds a new vertex between p_i^k and p_{i+1}^k
  - Uses a weight w – called a tension parameter
    - w = 0 produces a linear interpolation
    - w = $^1/_{16}$ produces a curve like below
  - Weights $0 < w < {}^1/_8$ produce curves with C^1 continuity



$$p_{2i}^{k+1} = p_i^k$$

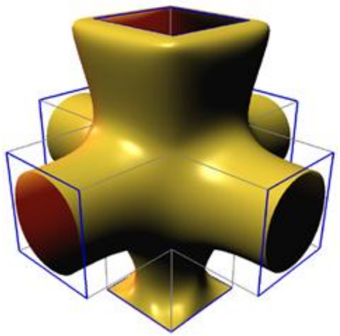
i is the vertex index
k is the subdivision level

$$p_{2i+1}^{k+1} = \left(\frac{1}{4} + w\right)\left(p_i^k + p_{i+1}^k\right) - w\left(p_{i-1}^k + p_{i+2}^k\right)$$

One subdivision – See Figure 17.39 (13.30 in 3rd Edition) for illustration of successive iterations

Subdivision Surfaces

- Powerful method produces smooth, continuous, crack-less surfaces
- Produces a surface with desired level of detail
  - Can generate as many levels of subdivision as needed from an original, compact representation
- Subdivision rules are generally simple
  - Easily implemented
- Good online descriptions
  - http://www.multires.caltech.edu/teaching/courses/subdivision
    - 1998 Siggraph course
  - https://www.gamedeveloper.com/programming/building-your-own-subdivision-surfaces
    - Aaron Lee



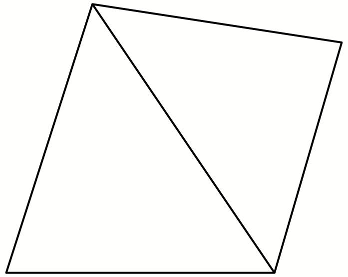
http://mrl.nyu.edu/projects/modeling_simulation/subdivision/
Control mesh shown with wireframe

Subdivision Surfaces

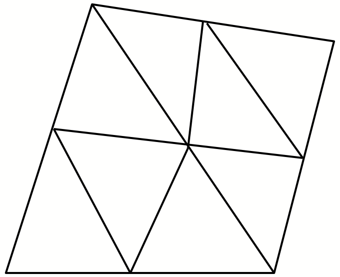
- Original polygon mesh is called the **control mesh**
- Subdivision surfaces are generally a two-phase process
  - **Refinement phase**
    - Creates new vertices and reconnects to form new, smaller triangles
    - A triangle or polygon can be split in different ways depending on the subdivision scheme
  - **Smoothing phase**
    - Computes new positions for some or all of the vertices
    - Choice of subdivision rules in this phase controls the level of continuity and whether the subdivision surface is approximating or interpolating
  - See Figures 17.41-42 (13.32-33 in 3rd Edition)
- Classifying subdivision methods
  - **Triangle-based** – works on triangles and only produces triangles
    - or **polygon-based** – operates on arbitrary polygons
  - **Stationary** – uses the same subdivision rules at each level
    - or **non-stationary** – different rules depending on the subdivision level
  - **Uniform** – uses same rules for every vertex or edge
    - or **non-uniform** – not all vertices and edges treated the same

Loop Subdivision

- Charles Loop’s subdivision method
  - Was first subdivision method for triangles - approximating
  - Updates each existing vertex and creates new vertex along each edge
- Each triangle is subdivided into 4 triangles
- Notation
  - Vertex p^k is a vertex after k subdivisions
    $$p^0 \rightarrow p^1 \rightarrow p^2 \rightarrow \ldots \rightarrow p^n$$
  - **Valence** of p^n is the number of neighboring vertices
    - Vertex with valence = 6 is called **regular** or ordinary
    - Otherwise the vertex is called irregular or extraordinary

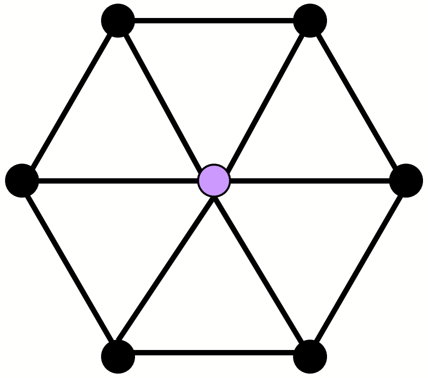


Control mesh



After 1st subdivision

See Figure 17.43 (13.34 in 3rd Edition) for next subdivision



Valence=6

Loop Subdivision

$$p^{k+1} = (1 - n\beta)p^k + \beta(p_0^k + \ldots + p_{n-1}^k), i = 0, \ldots, n-1$$

Rule for updating an existing vertex: $p^k \to p^{k+1}$

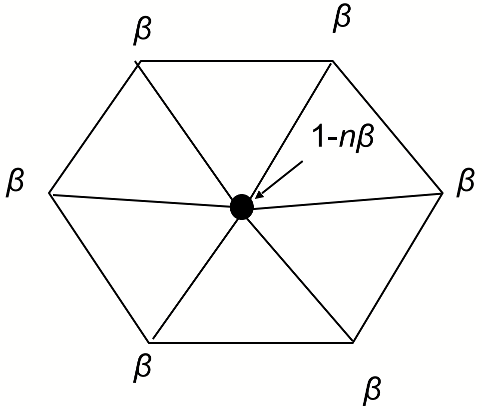
$$p^{k+1} = \frac{3p^k + 3p_i^k + 3p_{i-1}^k + 3p_{i+1}^k}{8}, i = 0, \ldots, n-1$$

Rule for creating a new vertex along an edge

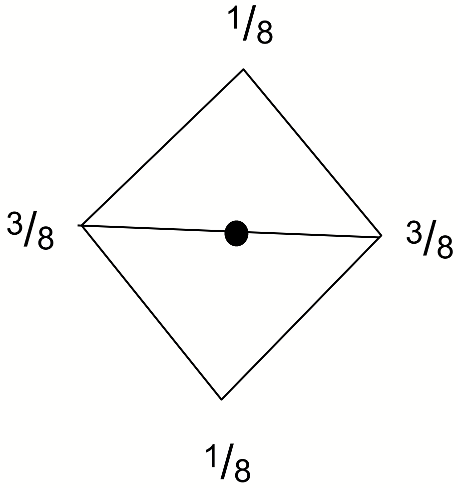
- Assume indices are modulo n
  - For i=n-1, i+1 will use index =0
- Subdivision rules can be visualized as masks
  - Simple illustration to describe the subdivision scheme
  - Weights sum to 1
- β is a function of n

$$\beta(n) = \frac{1}{n}\left(\frac{5}{8} + \frac{3 + 2\cos\left(\frac{2\pi}{n}\right)^2}{64}\right)$$

  - C^2 continuity at every regular vertex
  - C^1 elsewhere



Vertex mask



Edge mask

Loop Subdivision (cont.)

- Two "limit" tangents for a vertex p^k can be computed by weighting neighboring vertices:

$$t_u = \sum_{i=0}^{n-1} \cos\left(\frac{2\pi i}{n}\right) p_i^k \qquad t_v = \sum_{i=0}^{n-1} \sin\left(\frac{2\pi i}{n}\right) p_i^k$$

- Normal is: $n = t_u \times t_v$
  - Often more efficient than computing an average vertex normal based on normals of triangles sharing the vertex
  - Gives the exact normal at the point
- Advantages to Loop subdivision
  - Produces a **fair** surface
    - Fairness relates to how smoothly a surface bends
  - Converge faster than interpolating subdivision schemes
- Disadvantage
  - Shape often shrinks compared to the control mesh

Other Triangular Subdivision Methods

- Modified Butterfly Subdivision
  - Interpolating scheme
    - Once a vertex exists in the mesh its location never changes
  - Quality is highly model dependent
  - Uses 4 different rules for creating new vertices between existing vertices
    - **Regular setting**: create a new vertex between 2 regular vertices
    - **Semi-regular setting**: one vertex is regular and the other is irregular
    - **Irregular setting**: both existing vertices are irregular
    - **Boundaries**: when an edge has only one triangle connected to it
  - See Figure 17.47 (13.38 in 3rd Edition) for comparison between Loop and Modified Butterfly Subdivision
  - Good discussion:
    - https://www.gamedeveloper.com/programming/implementing-subdivision-surface-theory
- $\sqrt{3}$ Subdivision
  - Creates only 3 new triangles per subdivision step
    - Creates a new vertex in the middle of the triangle
      - Called a **mid-vertex**
  - To improve uniformity of triangles edges are flipped
    - Old edge is changed to connect 2 neighboring mid-vertices
  - https://people.eecs.berkeley.edu/~sequin/CS284/PAPERS/root3subdiv.pdf

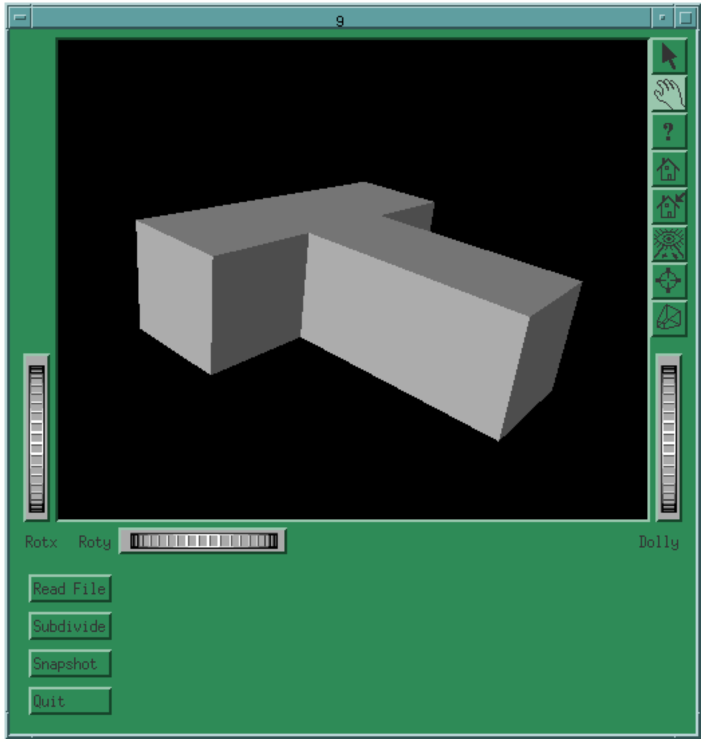
Catmull-Clark Subdivision

- Subdivision of a polyhedron
  - New vertices (**face points**) are placed at the center of each original face
  - New **edge points** are placed at the center of each original edge
  - New edges are added to connect the new edge points to the new adjacent face points.
- The positions of the new vertices are calculated as follows:
  - Face points are positioned as the average of the positions of the face's original vertices
  - Edge point locations are the average of the center point of the original edge and the average of the locations of the two new adjacent face points
- The old vertices are repositioned according to the equation

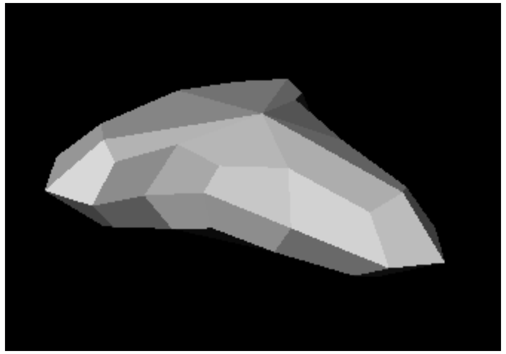
$$\frac{Q}{n} + \frac{2R}{n} + \frac{S(n-3)}{n}$$

  - Q is the average of the new face points surrounding the old vertex
  - R is the average of the midpoints of the edges that share the old vertex
  - S is the old vertex point
  - n is the number of edges that share the old vertex.

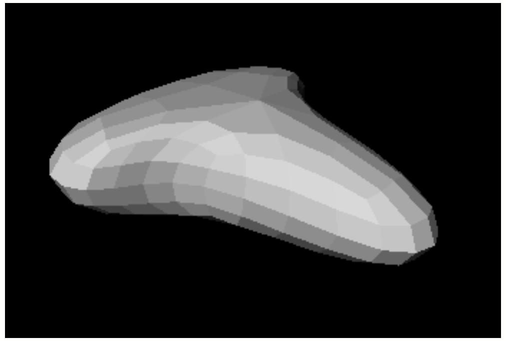
Catmull-Clark Subdivision



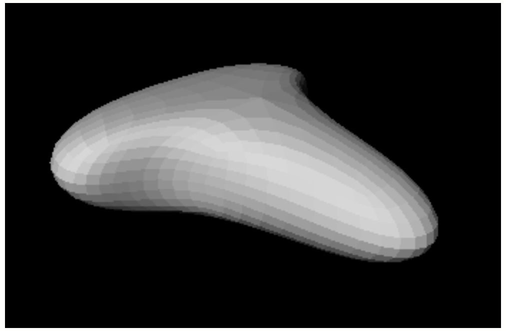
Original Polyhedron



Subdivision 1



Subdivision 2



Subdivision 3

http://symbolcraft.com/pjl/graphics/subdivision/ - (Link no longer found)