Subdivision Techniques

Adam Mally
CIS 277 Spring 2015
University of Pennsylvania
Catmull-Clark Subdivision

1. Initial mesh
Catmull-Clark Subdivision

2. For each Face, compute its centroid

Centroid = average of all vertices

Consider storing these centroids in a way that you can access a centroid given its corresponding face (e.g. a map)
3. Compute the smoothed **midpoint** of each edge in the mesh

\[ e = (v_1 + v_2 + f_1 + f_2) / 4 \]

If only one face is incident to the edge (as is the case for the borders of this image), then

\[ e = (v_1 + v_2 + f) / 3 \]

Note that for this particular mesh, the smoothed midpoints happen to just be the midpoints of the edges.
4. Smooth the original vertices

\[ v' = \frac{(n-2)v}{n} + \frac{\text{sum}(e)}{n^2} + \frac{\text{sum}(f)}{n^2} \]

\( v \) is the vertex’s original position

\( \text{sum}(e) \) is the sum of all adjacent midpoints

\( \text{sum}(f) \) is the sum of the centroids of all faces incident to \( v \)

This is the barycenter of the input components

\( n \) is the number of adjacent midpoints
Catmull-Clark Subdivision

5. For each original face, split that face into $N$ quadrangle faces

- $N$ is the number of vertices that the face originally had
- This is not a trivial task
  - Consider writing a Quadrangulate function that takes a Face, a centroid, and the set of half-edges that points to the midpoints on that face.
Common mistakes

- Forgetting to update/instantiate the following pointers:
  - HalfEdge::next
  - Face::start_edge
  - Vertex::incoming_edge
  - HalfEdge::face
  - HalfEdge::sym

- Smoothing the mesh one face at a time rather than performing each step globally
  - This leads to lopsided smoothed meshes

- Splitting each half-edge of the mesh into midpoints rather than splitting each full edge
  - Also leads to lopsided meshes
Finding all faces incident to a vertex

1. Create a list $L$ in which we will store the faces we’ve found
2. Given a Vertex $v$, store $v$.edge.face in $L$
3. Starting with HalfEdge $e = v$.edge.next, traverse the edges of $v$.edge.face
   a. When you reach a half-edge with a sym that points to $v$, $e = e$.sym
   b. Add $e$.face to $L$
4. Repeat until you discover a face that you’ve already inserted into $L$

This is useful for finding the centroids needed to compute the final position of the original vertices of your mesh.

You can also use this set of faces to find the midpoint vertices adjacent to a given vertex.
Finding all faces incident to a vertex
Finding all faces incident to a vertex
Finding all faces incident to a vertex
Finding all faces incident to a vertex
Finding all faces incident to a vertex
Finding all faces incident to a vertex
Finding all faces incident to a vertex
Finding all faces incident to a vertex
Finding all faces incident to a vertex
Subdividing with sharp edges

● Before a mesh is subdivided, some of its components may be tagged as “sharp”
● A sharp vertex never moves when subdivided
● A vertex bounded by N sharp edges follows these rules:
  ○ N < 2: Treated as a regular vertex; follows normal subdivision rules
  ○ N = 2: \( v' = (0.75)v + (0.125)v_1 + (0.125)v_2 \)
    ■ v1 and v2 are the vertices at the ends of the incident sharp edges
  ○ N > 2: Treated as a sharp vertex
● A sharp face treats all of its vertices as sharp
Subdividing with sharp edges

Fully smooth  Sharp edges  Sharp vertex

Image source: http://xrt.wikidot.com/blog:31
Subdividing with sharp edges

- We can also introduce semi-sharpness, which is a floating point number between 0 and 1.
- If a component has a sharpness of 1, it is treated as being fully sharp as in the previous slide.
- A sharpness of 0 means the component follows normal Catmull-Clark subdivision rules.
- A sharpness between 0 and 1 means the component’s final position is a linear interpolation of its fully sharp position and its fully smoothed position, where the interpolation’s $t$ value is the sharpness value.
Subdividing with sharp edges

Semi-sharp  Sharper  Even sharper

Image source: http://xrt.wikidot.com/blog:31