Computer Graphics

Shi-Min Hu

Tsinghua University

Today's Topics: mesh

- **Mesh**
  - Mesh tessellation
  - Mesh simplification
  - Subdivision
  - Mesh parameterization: Seam computation

- Mesh Description
  - A list of faces $F = (f_1, f_2, \ldots, f_n)$
    - Each face is a triangle
  - A list of vertices $V = (v_1, v_2, \ldots, v_n)$
  - Each face in F is a list of indices in V

    e.g.

    f_1 --(v_1, v_2, v_3), f_2 --(v_4, v_5, v_6),

    f_3 --(v_7, v_8, v_9), …

Why mesh representation?

- Computer generated 3D models and captured data have different representations,
  - We need a unified representation in graphics
  - visual accuracy and speed should be acceptable
  - Due to fast development of graphics hardware, we could rasterize(光栅化) and render triangles very quickly nowadays.

Source of 3D data

- **Models could be generated by:**
  - Directly typing in the geometric file
  - Writing code to create such data: *procedure modeling*
  - Using modeling software such as 3ds/max, maya
  - Capturing a real model using a 3D scanner(3D扫描仪)
  - Reconstruction(重构) from one or more photographs (照片), called *photogrammetry*(照相测量法)
  - Combination of techniques above
  - Others…

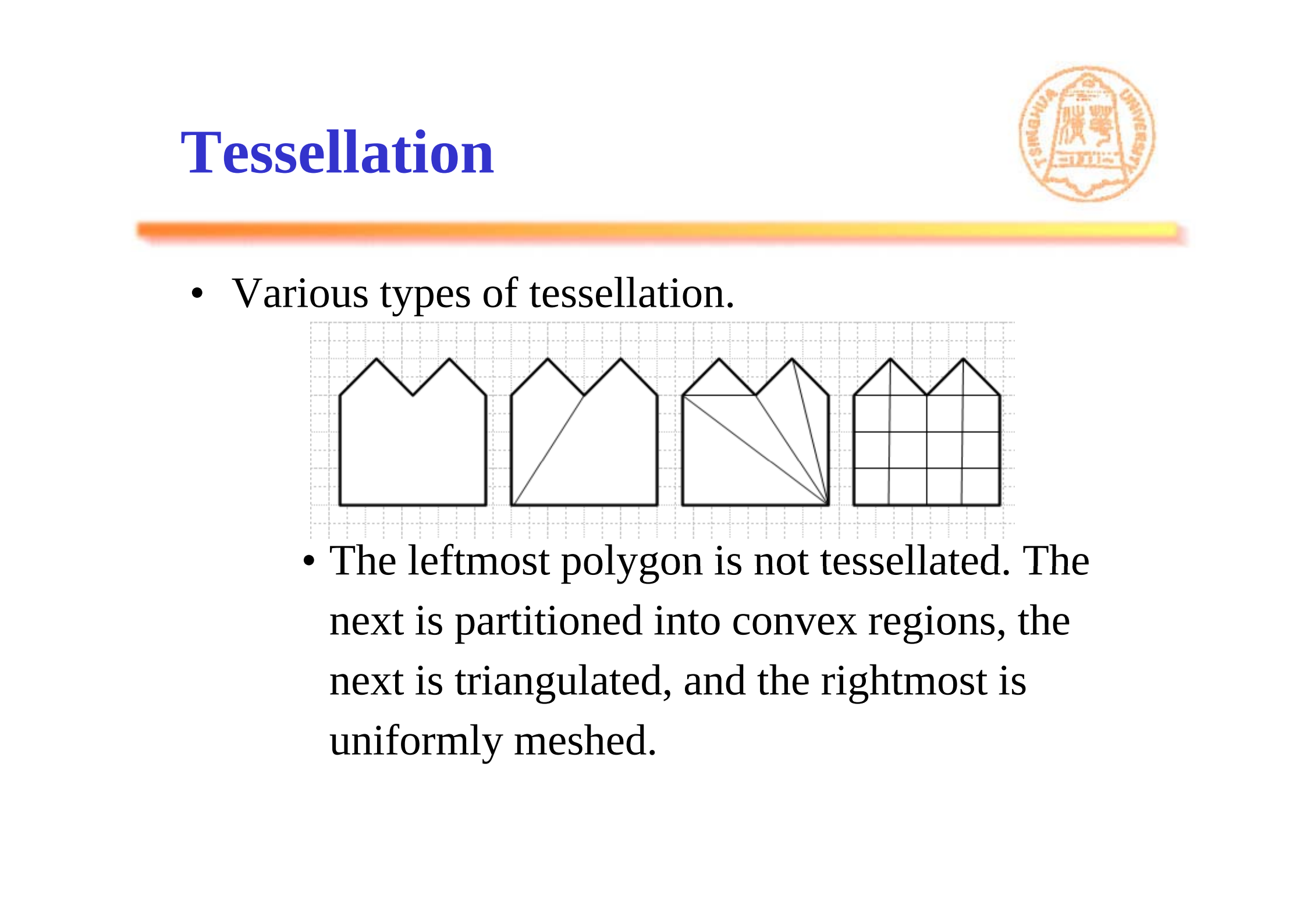
Two main type of modelers

- Solid-based
  - usually seen in the area of *Computer-Aided-Design* (CAD), often emphasize modeling tools correspond to actual machining processes, such as cutting, drilling(钻孔), etc.
- Surface-based
  - do not have a built-in concept of solidity;
  - instead, all objects are thought of in terms of surfaces. Direct manipulation of surfaces, such as adding/deleting polygons or vertices.
  - Easy for displaying

Tessellation

- Tessellation in 2D
  - We introduce 2D Tessellation because it's useful in 3D mesh processing
  - Polygons can arrive in many different forms, and may have to be split into more tractable primitives, such as convex polygons, triangles or quads, this process is called tessellation. If the polygon is split into triangles, this process is called triangulation(三角化)

Tessellation



- Various types of tessellation.

  - The leftmost polygon is not tessellated. The next is partitioned into convex regions, the next is triangulated, and the rightmost is uniformly meshed.

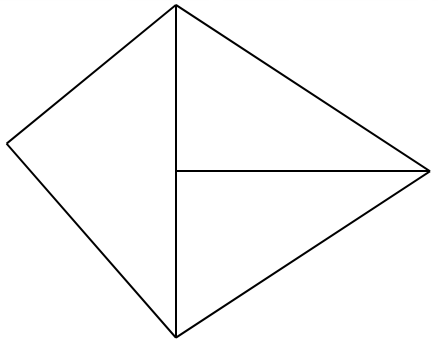
Tessellation

- A basic tessellation method
  - Examine each line segment between any two given vertices on a polygon and see if it intersects or overlaps any edge of polygon.
  - If it does, the line segment cannot be used to split the polygon, so examine the next possible pair of points, else split the polygon into two parts by this segment and triangulate the new polygons by the same method.
  - inefficient

Tessellation

- Another tessellation method: *ear clipping*
  - First, find the ears over the polygon, that is , to look at all triangles with vertex indices $i,i+1,i+2$ *(mod n)* and check line segment $i,i+2$ does not intersect any polygon edges.
  - If so, then triangle $i+1$ forms an ear. Each ear available is removed from the polygon, and the triangles at vertices i and $i+2$ are re-examined to see they are ears or not.

Tessellation



- T-vertices
  - T-vertex
    - Appear when joining flat surfaces where two models' edge meet, but do not share all vertices along them.
    - Even though the edges should theoretically meet perfectly, the renderer does not have enough precision in representing vertex locations on the screen.

Simplification(简化)

- What is Mesh Simplification?
  - approximating a given input mesh with a less complex but geometrically faithful representation

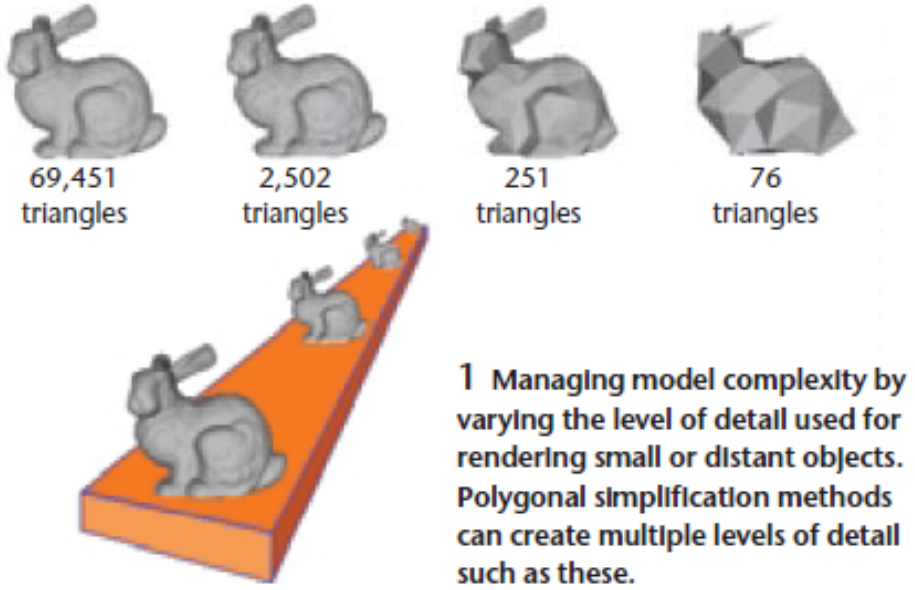
Simplification(简化)

- Why do we need Mesh Simplification?
  - remove redundant(多余的) geometry
    - E.g. a flat region with many small, coplanar(共面的) triangles. Merging these triangles to large polygons could decrease model's complexity.
  - reduce model size
    - reduce the size to store or transmit it.
  - improve run-time performance
    - simplification can be essential(重要的) to efficient rendering (to be continued)

Simplification(简化)

- generating levels of detail (LOD,层次细节) of objects in a scene. Presenting distant(远的) objects with a lower LOD and near object with a higher LOD.

- An example



1 Managing model complexity by varying the level of detail used for rendering small or distant objects. Polygonal simplification methods can create multiple levels of detail such as these.

Simplification(简化)

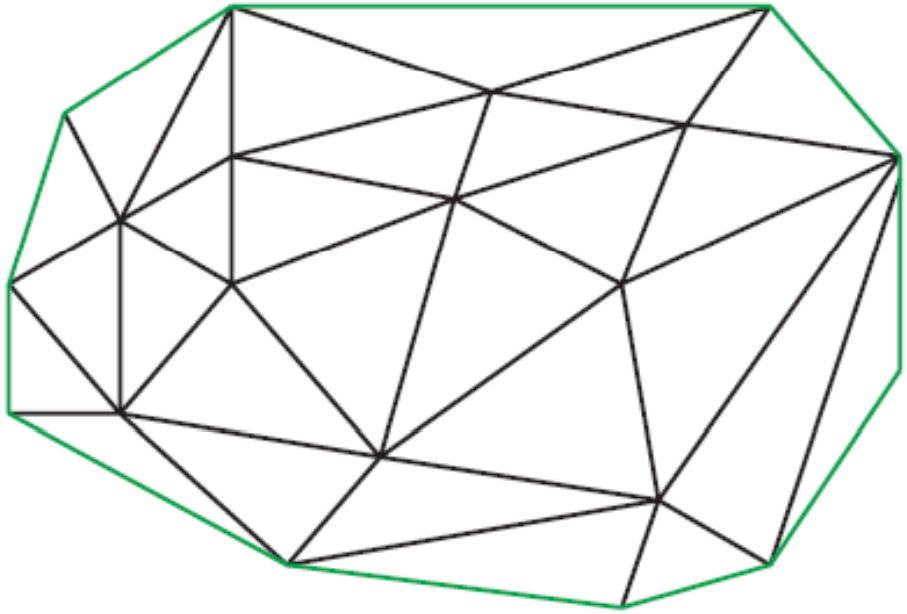
- Topology (拓扑结构)
  - refers to the connected polygonal mesh's structure
  - genus (亏格)
    - the number of holes in the mesh surface.
    - For example, a sphere and a cube have a genus of zero, while a torus(环) and a coffee cup have a genus of one.
  - *local topology* of a face, edge, or vertex
    - refers to the connectivity of that feature's neighborhood.

Simplification(简化)

- Topology (拓扑结构)
  - *2D manifold*
    - if the local topology is everywhere equivalent to a disc.
    - In a triangle mesh manifold topology, exactly two triangles share every edge, and every triangle shares an edge with exactly three neighboring triangles.
  - *2D manifold with boundary* (带边界的)
    - permits boundary edges, which belong to only one triangle.

Simplification(简化)

- Topology (拓扑结构)
  - *2D manifold with boundary*



2 A 2D manifold with a boundary (boundary edges in green). One or two triangles share each edge and a connected ring of triangles shares each vertex.

Simplification(简化)

- Mechanism: how to simplify?
  - Nearly every simplification technique in the literature uses some variation or combination of four basic polygon removal mechanisms below:
    - sampling,
    - adaptive subdivision,
    - decimation,
    - and vertex merging.

Mechanism

- Sampling(采样)
  - Sampling algorithms sample the initial model's geometry, with points on the model's surface. These methods are among the more elaborate and difficult to code approaches.
  - They may have trouble to sample the high-frequency features accurately. These algorithms usually work best on smooth surfaces with no sharp corners.

Feature sensitive Re-meshing

Mechanism

- Adaptive subdivision(自适应细分)
  - Adaptive subdivision algorithms find a simple *base mesh* that can be recursively(递归的) subdivided(细分) to closely approximate the initial model. This approach works best when the base model is easily found. For example, the base model for a terrain(地形) is typically a rectangle.
  - Adaptive subdivision methods preserve the surface topology, which may limit their capacity(能力) for drastic(大的) simplification.

Mechanism

- Decimation(去除)
  - Decimation techniques iteratively(迭代的) remove vertices or faces from the mesh, re-triangulating the resulting hole after each step.
  - These algorithms are relatively simple to code and can be very fast. Most use strictly local changes that tend to preserve the genus. These algorithms are good at removing redundant(冗余的) geometry such as coplanar(共面的) polygons.

Mechanism

- Vertex-merging(顶点合并)
  - Vertex-merging schemes operate by collapsing(断裂) two or more vertices of a triangulated model together into a single vertex, which in turn can be merged with other vertices.
  - Vertex merging is a simple and easy-to-code mechanism, but algorithms use various(多种的) techniques to determine which vertices to merge in what order.
  - Edge-collapse algorithms(边坍塌算法), which always merge two vertices sharing an edge, tend to preserve local topology, but algorithms permitting general vertex-merge operations can modify topology and aggregate objects.

Simplification(简化)

- **Category(分类)**
  - Static simplification(静态简化)
  - Dynamic simplification(动态简化)
  - View-dependent simplification(随视点相关的简化)

Category

- Static simplification(静态简化)
  - Creates several discrete simplified versions of each model in a preprocess, each at a different level of detail (层次细节). At runtime, rendering algorithms choose the appropriate LOD to represent the object.
  - Static simplification has many advantages. Decoupling simplification and rendering makes this the simplest model to program. The simplification algorithm generates LODs without regard to real-time rendering constraints, and the rendering algorithm simply chooses which LODs to render.

Category

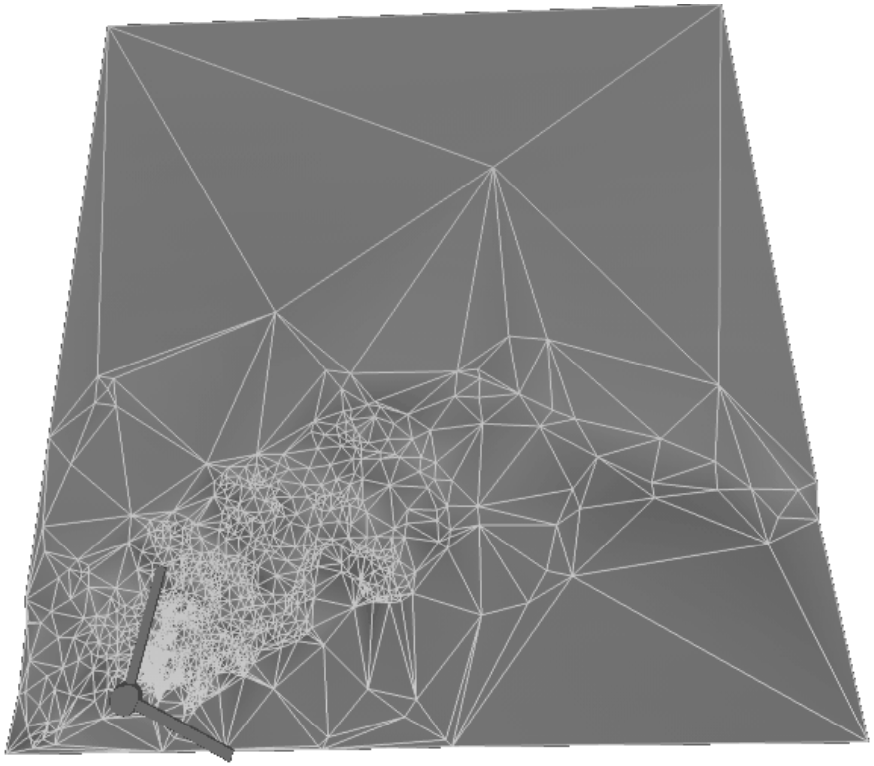
- Dynamic simplification(动态简化)
  - *Dynamic polygonal simplification* is different from the traditional static approach. Whereas a static simplification algorithm creates individual LODs during the preprocessing stage, a dynamic simplification system creates a data structure encoding a continuous(连续的) level of detail.
  - A major advantage of this approach is better continuity and granularity(粒度), rather than choosing from a few pre-created simplified models, which would produce gap(缺口) when switching from one model to another one.

Category

- View-dependent simplification(随视点相关的简化)
  - extends dynamic simplification by using view-dependent criteria(标准) to select the most appropriate LOD for the current view.
  - In a view-dependent system, a single object can span multiple levels of simplification. For instance, nearby portions(部分) of the object may appear at a higher resolution than distant portions, or silhouette(轮廓) regions of the object may appear at a higher resolution than interior(内部) regions.

Category

- View-dependent simplification(与视点相关的简化)



Simplification(简化)

- Now, we will introduce two algorithms for simplification in detail
  - vertex decimation （顶点去除）
  - edge contraction(收缩)

Vertex Decimation

- Vertex Decimation
  - First proposed by [Schroeder et. al. 1992], operates on a single vertex by deleting that vertex and re-tessellating the resulting hole.
  - Typically consists of:
    - select the vertex to delete by some error metric
    - based on the adjacent information of the selected vertex, determine how to re-tessellate the hole
    - Recursively delete vertices using step 1&2.

Vertex Decimation

- Error metric to select vertex
  - for each vertex v
    - consider the triangles in v's neighbourhood, and compute an approximating plane, based on the area-weighted average of the triangle normals n_k, centers x_k, and areas A_k:

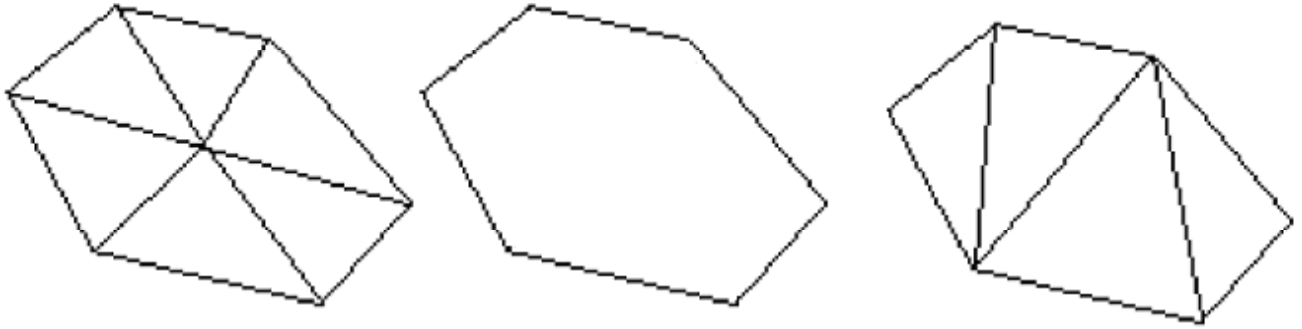
$$\mathbf{n} = \frac{\sum_k A_k \mathbf{n}_k}{\sum_k A_k} \qquad \hat{\mathbf{n}} = \frac{\mathbf{n}}{|\mathbf{n}|} \qquad \mathbf{x} = \frac{\sum_k A_k \mathbf{x}_k}{\sum_k A_k}$$

Vertex Decimation

- Error metric to select vertex （Continued）
  - Then calculate the distance from vertex v to this plane as the “error metric”
  - select the vertex with the minimal distance and remove this vertex
  - re-tessellate the hole caused by deleting of the vertex
- This metric ensures that vertices in smooth regions will be decimated before vertices that define sharp features

Vertex Decimation

- Illustration



  - left: before deleting vertex
  - middle: after deleting vertex
  - right: re-tessellate
- Refer to paper: [Schroeder92] “Decimation of triangle meshes ”

Edge Contraction(收缩)

- Edge Contraction
  - originally proposed in [Hoppe et al. 1993], is the most common simplification operation used in computer graphics.
  - An edge contraction operates on a single edge {i, j} and contracts that edge to a single vertex {h}, updating all edges previously connected on {i} and {j} to reference {h}. (see the figure on next page)

Edge Contraction(收缩)

- Edge Contraction

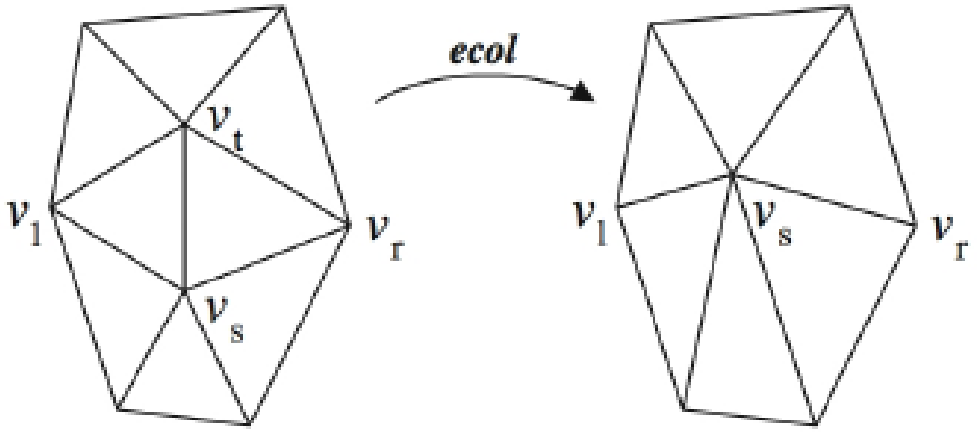


Fig. 1. The edge contraction operation [Hoppe 1993].

Edge Contraction(收缩)

- Edge Contraction
  - Select an edge: based on the length of the edge
  - New vertex placement
    - however, for a given contraction edge {i, j} → vertex {h}, it is not immediately clear what value should be assigned to vertex {h}, i.e. where the resulting vertex should be placed.
    - Obvious choices such as

      $\mathbf{v}_h = \mathbf{v}_i,\ \mathbf{v}_h = \mathbf{v}_j,$ or $\mathbf{v}_h = (\mathbf{v}_i + \mathbf{v}_j)/2$

      are convenient, but not necessarily optimal (最优)

Edge Contraction(收缩)

- Edge Contraction
  - Error metric for new vertex placement [Garland and Heckbert 1997]
    - consider the newly generated triangles around the new vertex $\mathbf{v}_h$
    - Calculate the sum *S* of square of distances from vertex $\mathbf{v}_i$ and $\mathbf{v}_j$ to these triangles
    - find the $\mathbf{v}_h$ that makes the sum S minimal
  - Refer to the paper [Garland and Heckbert 1997] “Surface simplification using quadric error metrics”

Edge Contraction(收缩)

- Result

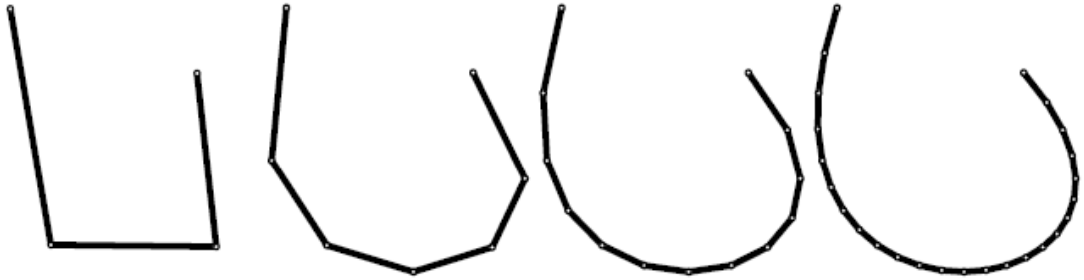
  - A sequence of simplified models generated by an edge contraction algorithm [Garland and Heckbert 1997]

Subdivision(细分)

- Subdivision
  - the publication of the papers by Catmull and Clark, Doo and Sabin in 1978 marked the beginning of subdivision for surface modeling. Now we can regularly see subdivision used in movie production.
  - a loose description of subdivision
    - Given a original mesh, generate a smoother one by a sequence of successive refinement

Subdivision(细分)

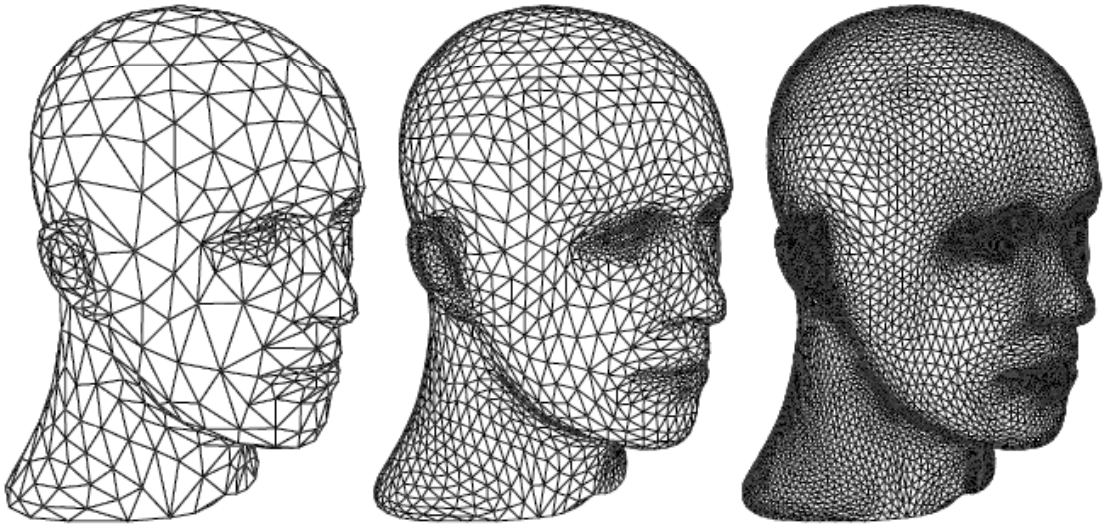
- Example of Subdivision in 1D



  - On the left 4 points connected with straight line segments.
  - To the right of it a refined version: 3 new points have been inserted between the old points.
  - After two more steps of subdivision the curve starts to become rather smooth.

Subdivision(细分)

- Another Example of Subdivision



Example of subdivision, showing 3 successive levels of refinement. On the left an initial triangular mesh. Each triangle is split into 4 according to a particular subdivision rule (middle). On the right the mesh is subdivided in this fashion once again.

Subdivision(细分)

- Subdivision
  - the subdivision of surfaces can be thought of as a two-phase process. (The starting is called *control mesh*).
    - The first phase, called the *refinement phase*, creates new vertices and reconnects to create new, smaller triangles
    - The second phase, called the *smoothing phase*, computes the new positions for some or all vertices.
  - The details of these two phases determine a subdivision scheme. In the first phase, a triangle can be split in different ways; in the second phase, the positions for vertices could be interpolated differently.

Subdivision Scheme

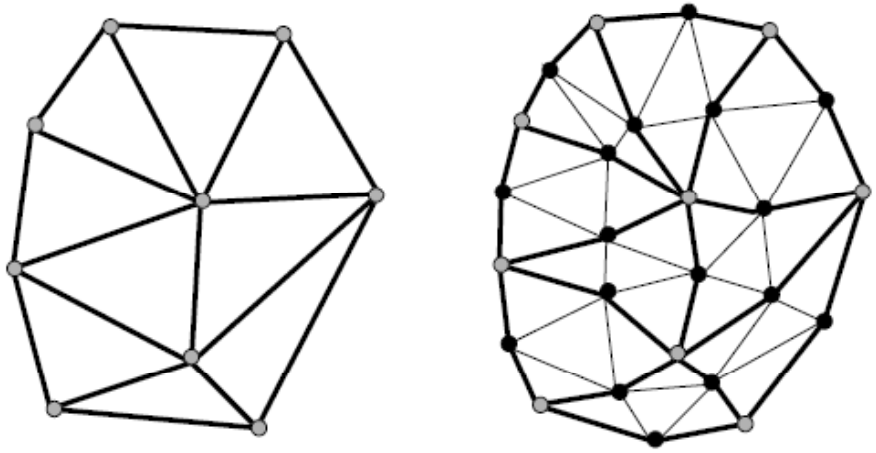
- Subdivision Schemes
  - we will introduce 2 subdivision scheme
    - Loop Subdivision
    - $\sqrt{3}$ Subdivision

Loop Subdivision

- Loop Subdivision
  - Loop subdivision scheme was the first subdivision scheme for triangles.
  - It updates each existing vertex and creates a new vertex for each edge, then each triangle is subdivided into 4 new triangles. So after *n* subdivision steps, a triangle will be subdivided into 4^n triangles.

Loop Subdivision

- Loop Subdivision



Example of Loop Scheme. New vertices are shown as black dots. a new vertex is added on each edge, and new vertices are reconnected to form 4 new triangles, replacing each triangle of the mesh.

Loop Subdivision

- Denotation of subdivision rule
  - Focus on an existing vertex p^k ,where k is the number of current subdivision steps. This means that p^0 is the vertex of the control mesh. In general, subdivision processes

$$p^0 \to p^1 \to p^2 \to p^3 \ldots$$

  - If the valence(度) of p^k is n, then p^k has n neighbor vertices, denote the neighbors are $p_i^k, i \in \{0,1,\ldots,n-1\}$

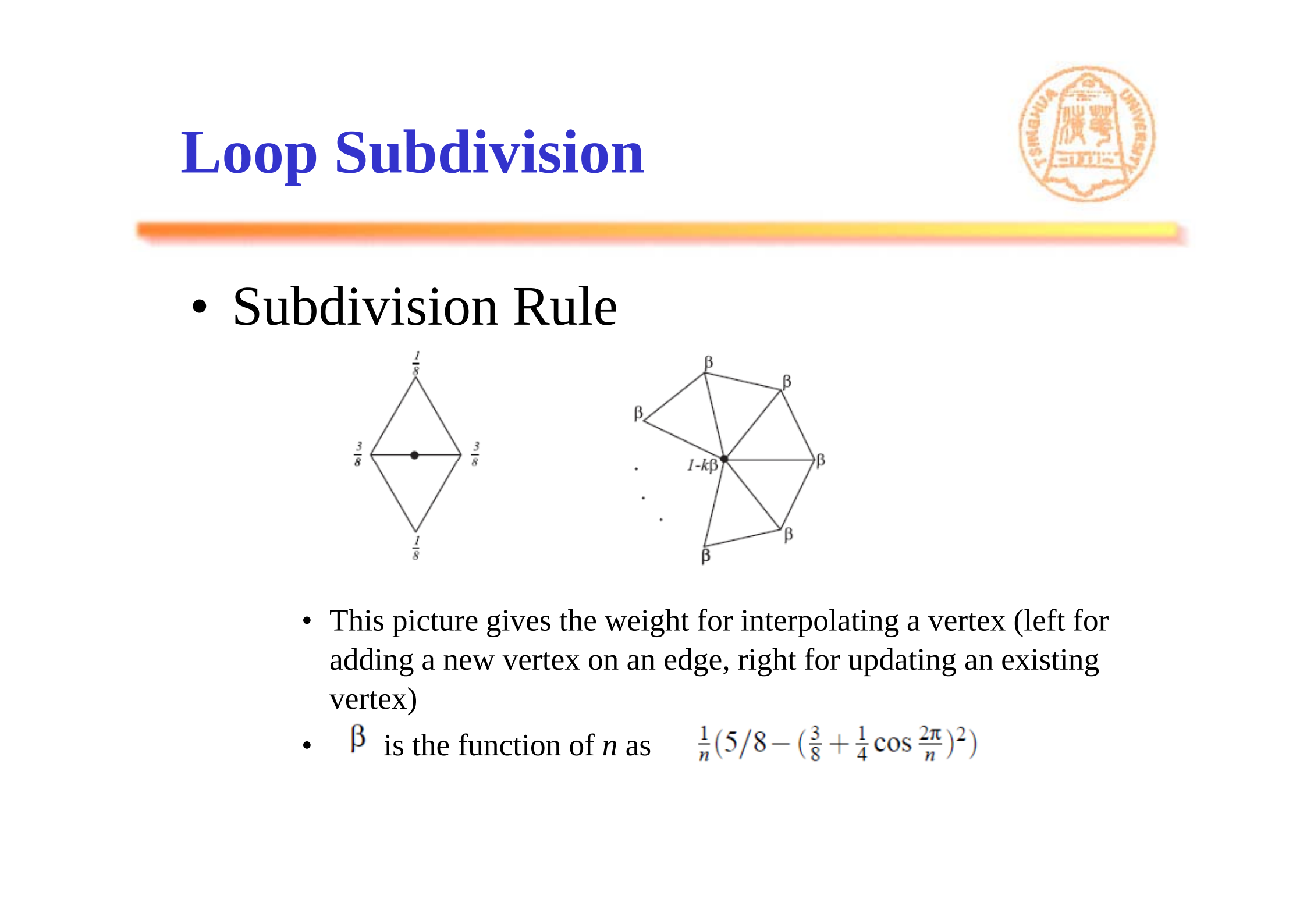
Loop Subdivision

- Subdivision rule
  - Below gives the subdivision rules of Loop's scheme. The first formula is the rule for updating an existing vertex from p^k into p^{k+1}, and the second formula is for creating a new vertex p_i^{k+1}, on the edge between p^k and p_i^k .

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

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

Loop Subdivision



- Subdivision Rule

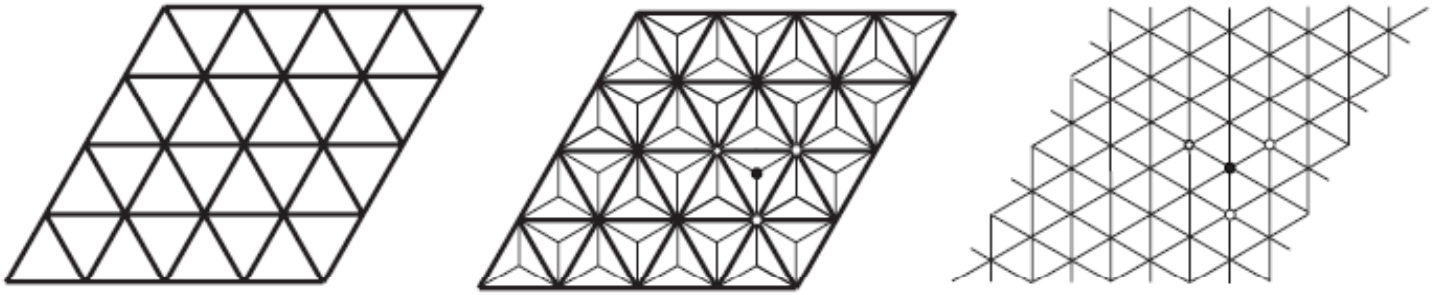
  - This picture gives the weight for interpolating a vertex (left for adding a new vertex on an edge, right for updating an existing vertex)
  - β is the function of n as $\frac{1}{n}(5/8-(\frac{3}{8}+\frac{1}{4}\cos\frac{2\pi}{n})^2)$

$\sqrt{3}$ - Subdivision

- $\sqrt{3}$ - Subdivision
  - Loop's scheme splits each triangle into four new ones, and so creates triangles at a rate of $4^n m$, where m is the number of triangles in control mesh, and n is the number of subdivision step.
  - Different from Loop's scheme, Kobbelt's $\sqrt{3}$ scheme creates only 3 new triangles per-step.
  - It creates a new vertex (called *mid-vertex*) in the middle of each triangle, instead of a new vertex per edge.

$\sqrt{3}$ - Subdivision

- $\sqrt{3}$ - Subdivision
  - To get more uniformly shaped triangles, each old edge is flipped(翻转) so that it connects two neighboring mid-vertices, instead of connecting two neighboring existing vertices.



  - this figure shows the process of $\sqrt{3}$ subdivision

$\sqrt{3}$ - Subdivision

- Subdivision rule
  - As shown in the formula in the first line, where p_m denotes the mid-vertex, computed as the average of the triangle vertices: p_a, p_b, p_c
    Each of the existing vertices, p^k , are updated using the formula in the second line, where p_i^k denotes the neighbors of p^k. And as in Loop's scheme, n is the valence of p^k, and k is the subdivision step.

$$p_m^{k+1} = (p_a^k + p_b^k + p_c^k)/3$$

$$p^{k+1} = (1 - n\beta)\, p^k + \beta \sum_{i=0}^{n-1} p_i^k$$

$\sqrt{3}$ - Subdivision

- Subdivision rule
  - Again, β is a function of valence n, and the following choice of $\beta(n)$ generates a surface, where the continuity is at least C1.

$$\beta(n) = \frac{4 - 2\cos(2\pi / n)}{9n}$$

Reference

- UIUC 02-PolygonalModeling.pdf
- William J. Schroeder Jonathan A. Zarge William E. Lorensen **Decimation of Triangle Meshes** http://www.cse.iitd.ernet.in/~jatin/minip/papers/deci-schroeder.pdf
- Hugues Hoppe **Progressive Meshes** **http://research.microsoft.com/en-us/um/people/hoppe/pm.pdf**
- Michael Garland, Paul S. Heckbert **Surface Simplification Using Quadric Error Metrics** **http://mgarland.org/files/papers/quadrics.pdf**

Reference

- http://www.pixar.com/
- http://www.cs.princeton.edu/courses/archive/spring09/cos426/lectures/07-subdivision.pdf
- J. Vorsatz, C. Rössl, L. P. Kobbelt and H.-P. Seidel **Feature Sensitive Remeshing** **http://www3.interscience.wiley.com/cgi-bin/fulltext/118914590/PDFSTART**
- **Leif Kobbelt** - Subdivision $\sqrt{3}$ http://delivery.acm.org/10.1145/350000/344835/p103-kobbelt.pdf?key1=344835&key2=6753041421&coll=GUIDE&dl=GUIDE&CFID=34130496&CFTOKEN=68501460