Geometric Modeling

Bing-Yu Chen
National Taiwan University
The University of Tokyo

Surface Simplification

- Motivation
- Basic Idea of LOD
- Discrete LOD
- Continuous LOD
- Simplification Problem Characteristics
- Methodology Overview
- Simplification Algorithms
- The Vertex Tree

Motivation

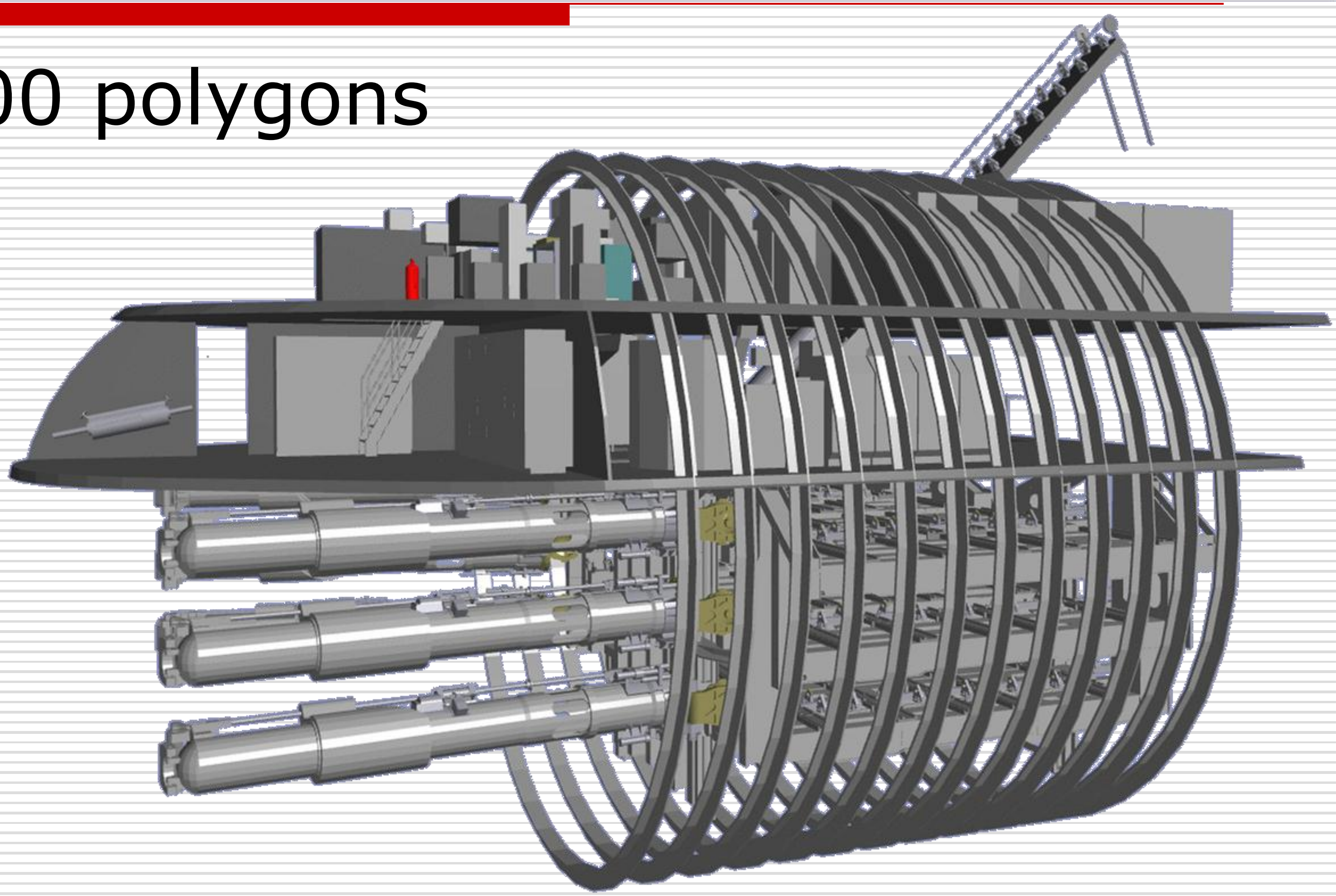
- Interactive rendering of large-scale geometric datasets is important
  - Scientific and medical visualization
  - Architectural and industrial CAD
  - Training (military and otherwise)
  - Entertainment

Motivation: Big Models

- The problem:
  - Polygonal models are often too complex to render at interactive rates
- Even worse:
  - Incredibly, models are getting bigger as fast as hardware is getting faster…

Big Models: Submarine Torpedo Room

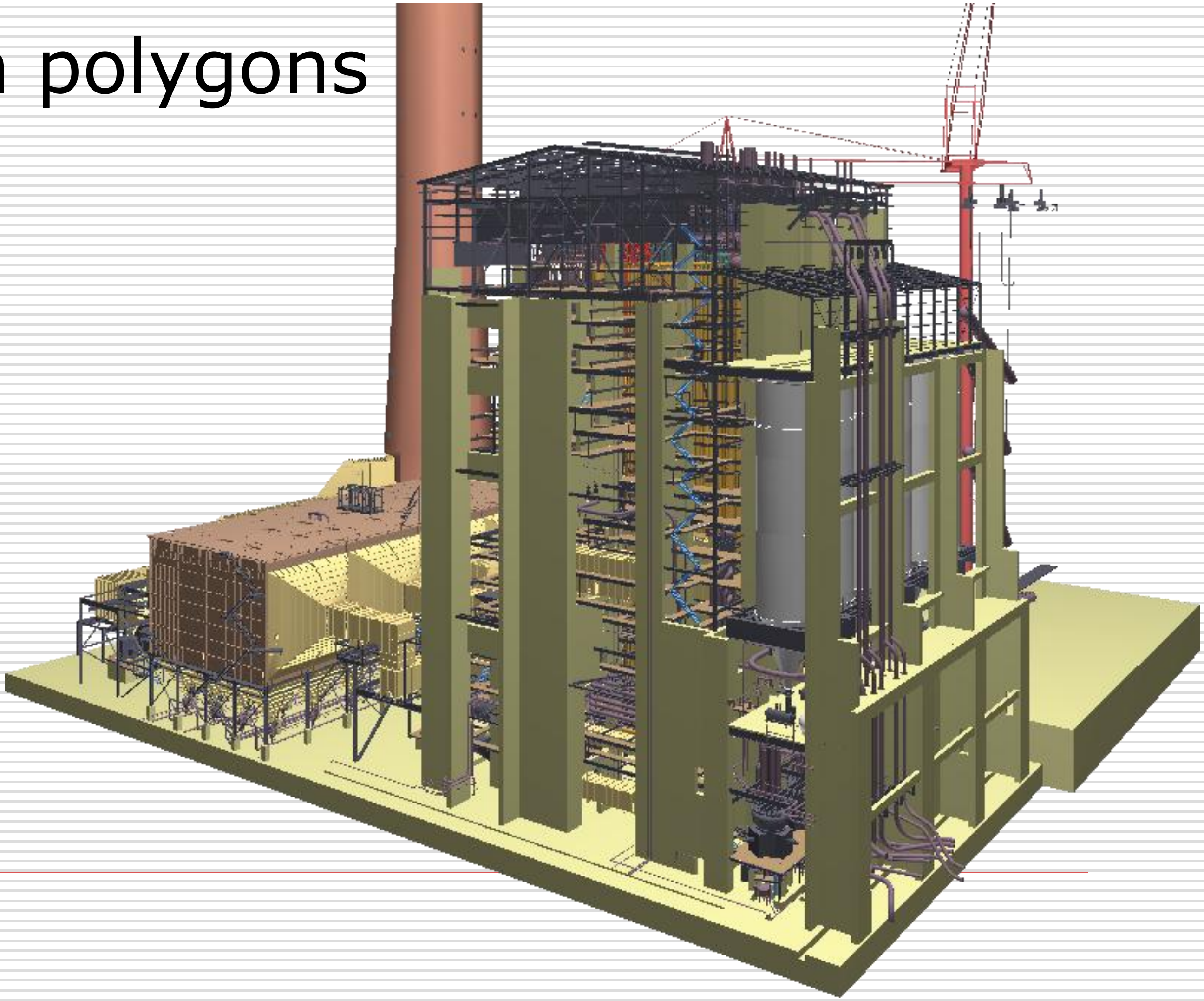
- 700,000 polygons



Courtesy General Dynamics, Electric Boat Div.

Big Models: Coal-fired Power Plant

- 13 million polygons



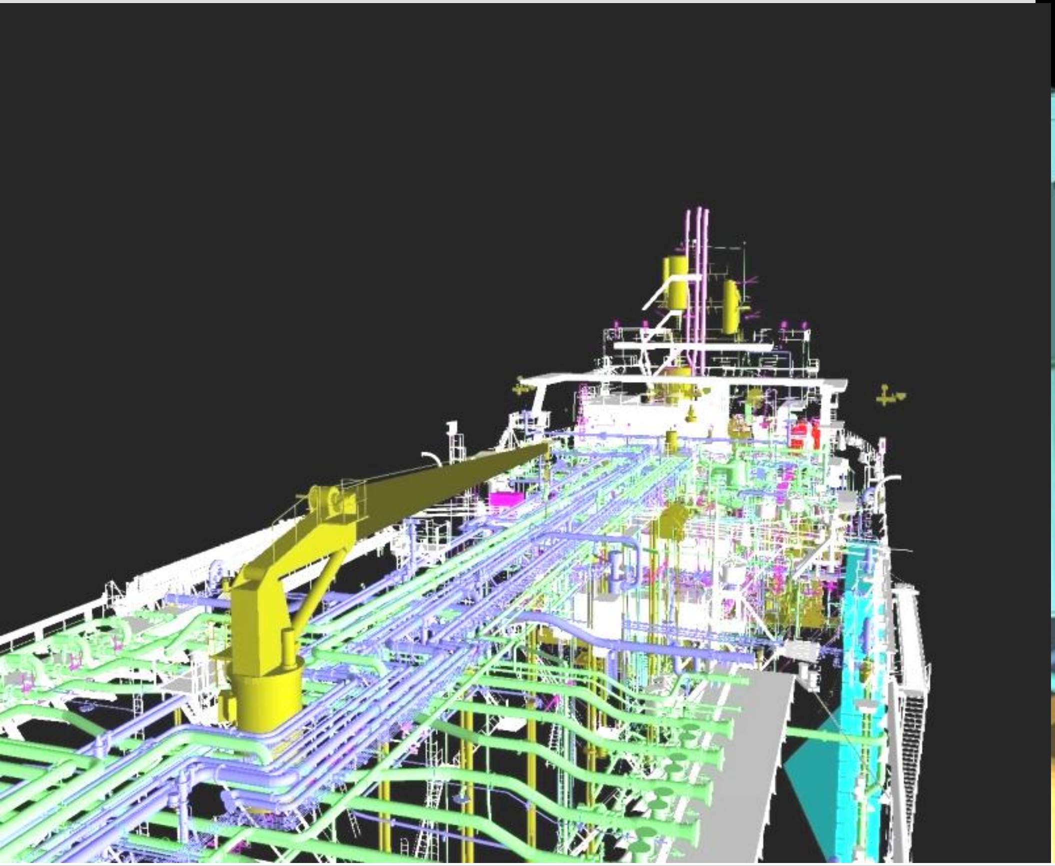
Big Models:
Plant Ecosystem Simulation

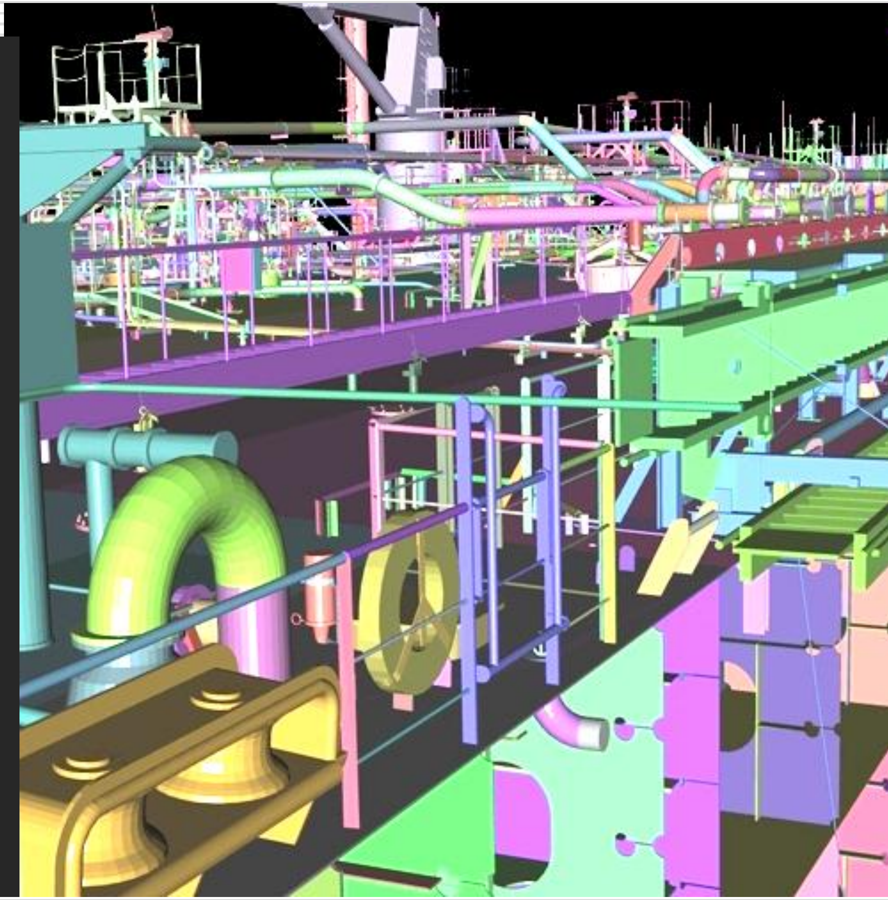
- 16.7 million polygons (sort of)

Deussen et al: *Realistic Modeling of Plant Ecosystems*

Big Models:
Double Eagle Container Ship

- 82 million polygons





Courtesy Newport News Shipbuilding

Big Models: The Digital Michelangelo Project

- David:
  56,230,343 polygons
- St. Matthew:
  372,422,615 polygons

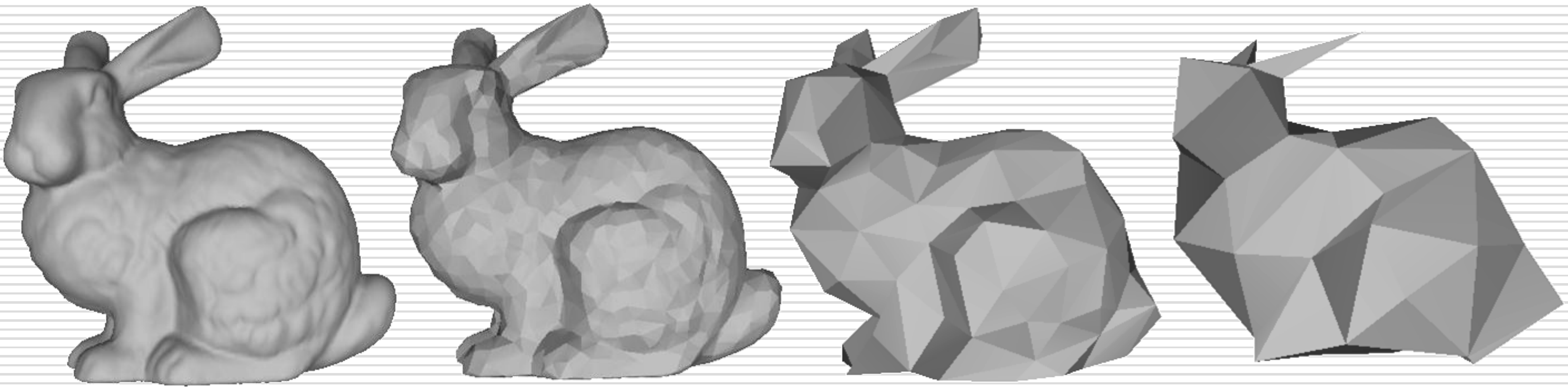
Courtesy Digital Michelangelo Project, Stanford Univ.

Level of Detail: The Basic Idea

- One solution:
  - Simplify the polygonal geometry of small or distant objects
  - Known as *Level of Detail* or *LOD*
    - A.k.a. polygonal simplification, geometric simplification, mesh reduction, multiresolution modeling, …

Level of Detail: Traditional Approach

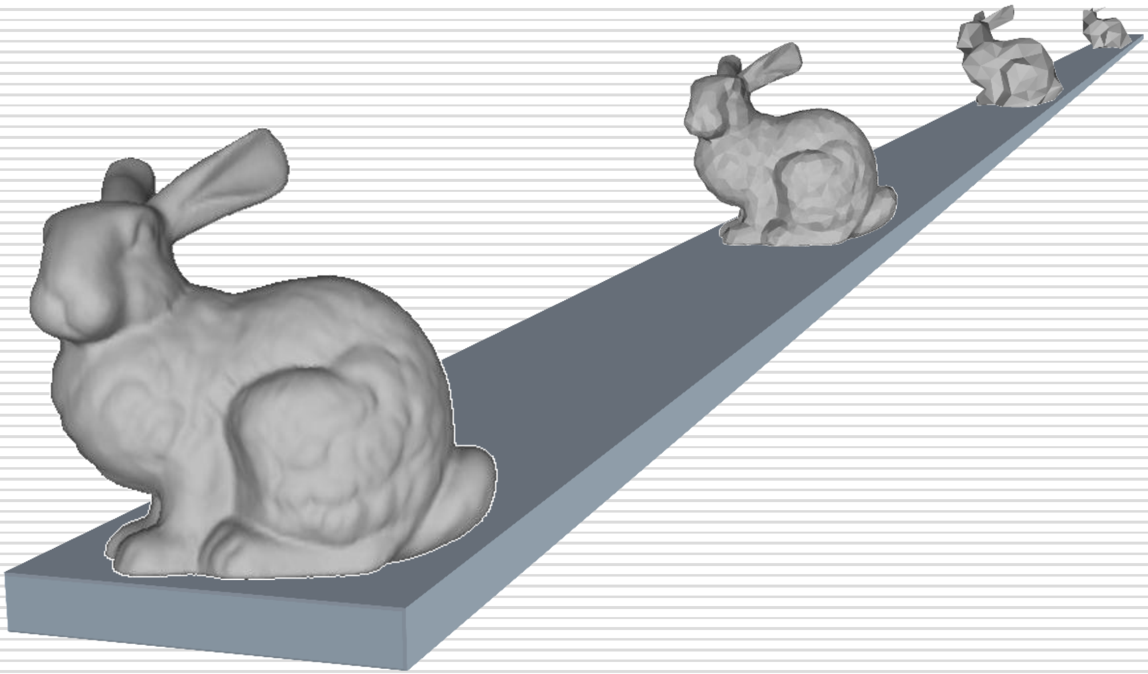
- Create *levels of detail* (*LODs*) of objects:



69,451 polys | 2,502 polys | 251 polys | 76 polys

Level of Detail: Traditional Approach

- Distant objects use coarser LODs:



Traditional Approach: Discrete Level of Detail

- Traditional LOD in a nutshell:
  - Create LODs for each object separately in a preprocess
  - At run-time, pick each object's LOD according to the object's distance (or similar criterion)
- Since LODs are created offline at fixed resolutions, this can be referred as Discrete LOD

Discrete LOD: Advantages

- Simplest programming model; decouples simplification and rendering
  - LOD creation need not address real-time rendering constraints
  - Run-time rendering need only pick LODs

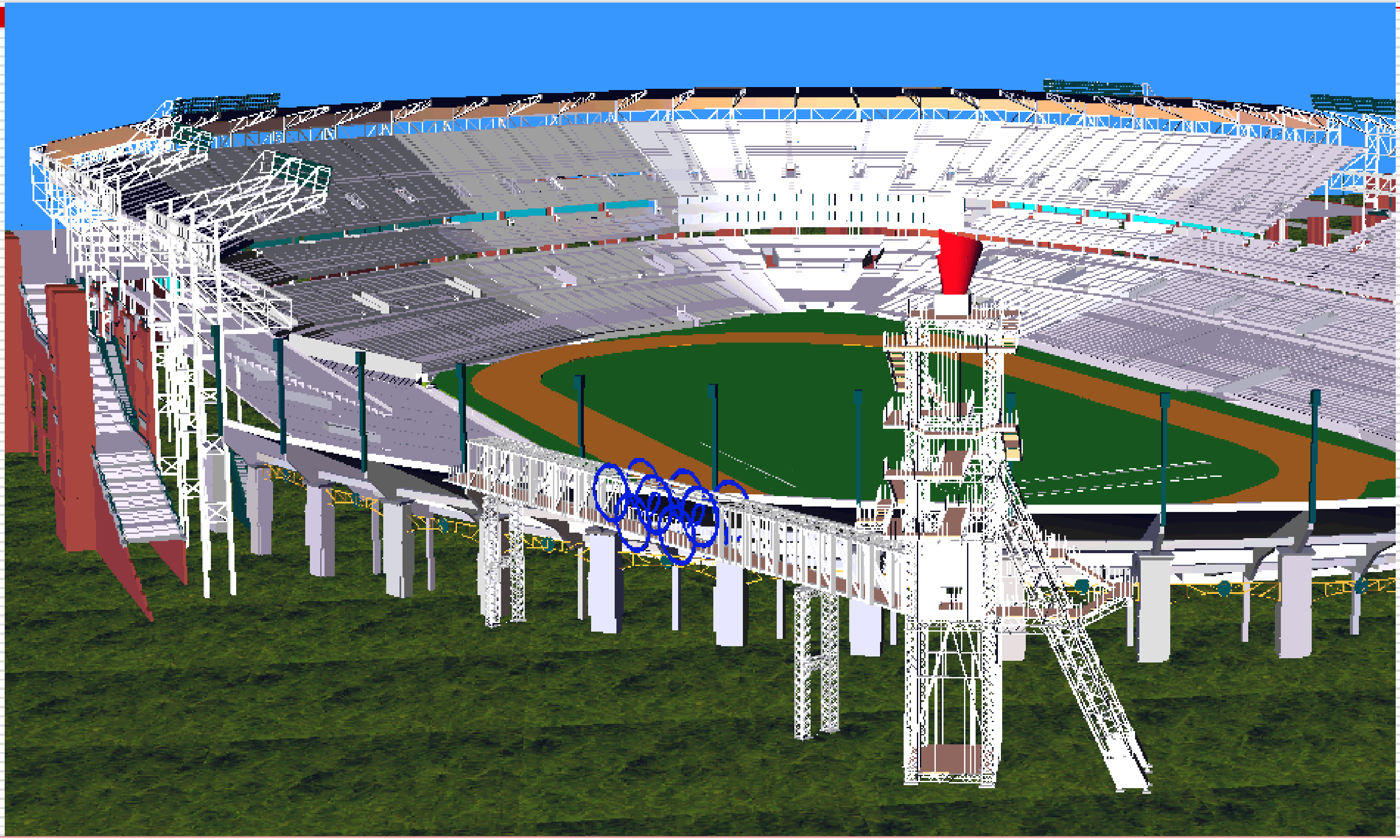
Discrete LOD: Advantages

- Fits modern graphics hardware well
  - Easy to compile each LOD into triangle strips, display lists, vertex arrays, …
  - These render *much* faster than unorganized polygons on today’s hardware (3-5 x)

Discrete LOD: Disadvantages

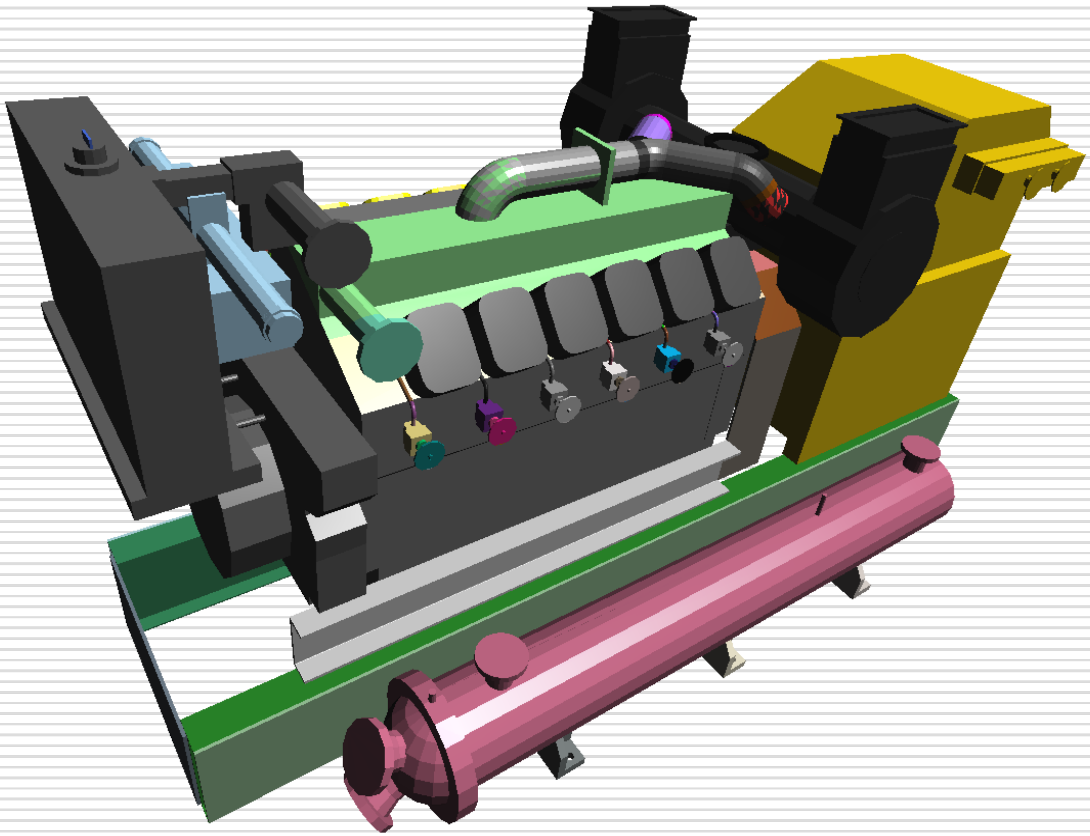
- So why use anything but discrete LOD?
- Answer: sometimes discrete LOD not suited for *drastic simplification*
- Some problem cases:
  - Terrain flyovers
  - Volumetric isosurfaces
  - Super-detailed range scans
  - Massive CAD models

Drastic Simplification: The Problem With Large Objects



Courtesy IBM and ACOG

Drastic Simplification: The Problem With Small Objects



Courtesy Electric Boat

Drastic Simplification: The Problem With Topology

Courtesy University of Utah

Drastic Simplification

- For drastic simplification:
  - Large objects must be subdivided
  - Small objects must be combined
  - Topology must be simplified
- Difficult or impossible with discrete LOD

Continuous Level of Detail

- A departure from the traditional static approach:
  - Discrete LOD: create individual LODs in a preprocess
  - Continuous LOD: create data structure from which a desired level of detail can be extracted at *run time*.

Continuous LOD: Advantages

- Better granularity → better fidelity
  - LOD is specified exactly, not chosen from a few pre-created options
  - Thus objects use no more polygons than necessary, which frees up polygons for other objects
  - Net result: better resource utilization, leading to better overall fidelity/polygon

Continuous LOD: Advantages

- Better granularity → smoother transitions
  - Switching between traditional LODs can introduce visual “popping” effect
  - Continuous LOD can adjust detail gradually and incrementally, reducing visual pops
    - Can even *geomorph* the fine-grained simplification operations over several frames to eliminate pops [Hoppe 96, 98]

Continuous LOD: Advantages

- Supports progressive transmission
  - *Progressive Meshes [Hoppe 97]*
  - *Progressive Forest Split Compression [Taubin 98]*
- Leads to *view-dependent LOD*
  - Use current view parameters to select best representation for *the current view*
  - Single objects may thus span several levels of detail

Typical Curve & Surface Simplification Problems

- Typical Curve Simplification Problem
  - Given curve with *n* vertices, find an accurate approximation using *m* vertices.
  - Given curve with *n* vertices, find a compact approximation with error < ε.
- Typical Surface Simplification Problem
  - Given surface with *n* vertices, find accurate approximation using m vertices.
  - Given surface with *n* vertices, find a compact approximation with error < ε.

Simplification Problem Characteristics - What problem do you want to solve?

- topology of output
  - curve or surface
- topology & geometry of input
  - points, function f(x), curve, height field f(x,y), manifold, surface
- other attributes:
  - color, texture
- domain of output
  - subset of input vertices?

Simplification Problem Characteristics - What problem do you want to solve?

- topology of triangulation
  - uniform, hierarchical, general
- approximating elements
  - linear, quadratic, cubic, …,other
- error metric
  - L2 = sum of squared, L∞ = maximum
- constraints
  - most accurate using a given number of elements or amount of memory
  - smallest satisfying a given error tolerance

Simplification Problem Characteristics - How do you want to solve the problem?

- speed / quality tradeoff
  - optimal (& slow) or sub-optimal (& fast)?
- refinement / decimation
  - top down or bottom up?
- number of passes
  - one pass or multiple passes?
- triangulation
  - hierarchical triangulation, Delaunay triangulation, data-dependent triangulation, or other?

Performance Requirements

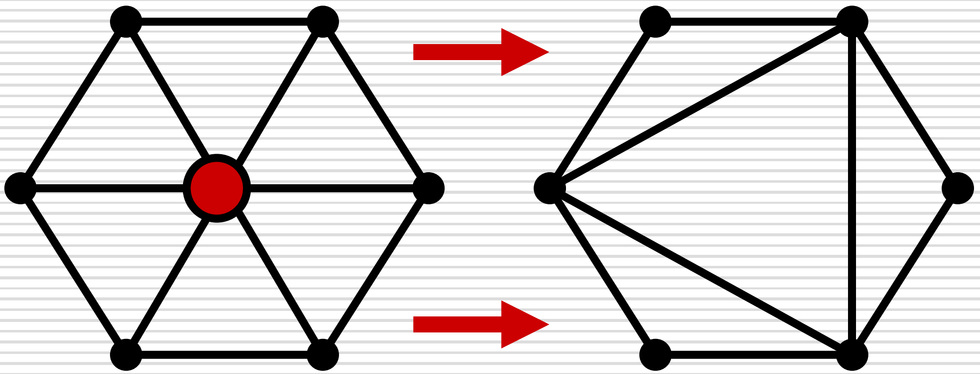
- Offline
  - Generate model at given level(s) of detail
  - Focus on quality
- Real-time
  - Generate model at given level(s) of detail
  - Focus on speed
  - Requires preprocessing
  - Time/space/quality tradeoff

Taxonomy of Surface Simplification Methods

- Height Field / Parametric Simplification
  - subsampling, pyramid, quadtree methods
  - greedy insertion [Garland95]
- Manifold Simplification
  - vertex decimation [Schroeder92]
  - vertex decimation with point lists [Eck95] [Lounsbery94]
  - wavelet [Hoppe93]
  - edge collapse [Ronfard96] [Hoppe96] [Gueziec95] [Garland97]
- Non-Manifold Simplification
  - vertex clustering [Rossignac93]

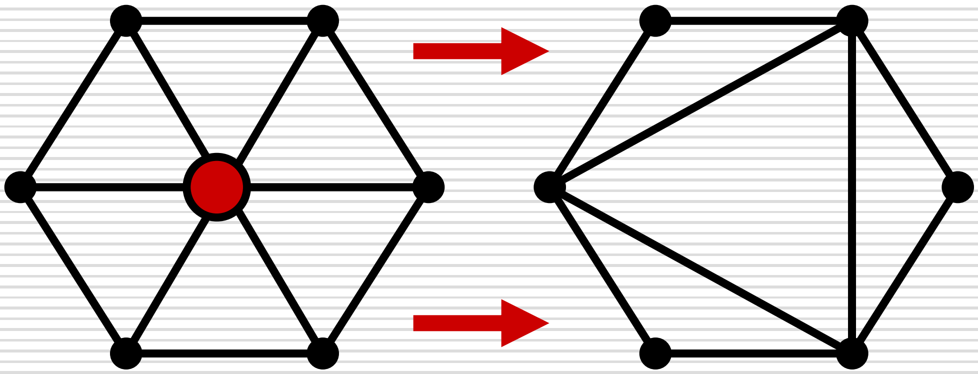
Methodology

- Sequence of local operations
  - Involve near neighbors - only small *patch* affected in each operation
  - Each operation introduces error
  - Find and apply operation which introduces the least error



Simplification Operations

- Decimation
  - Vertex removal
    - v ← v-1
    - f ← f-2

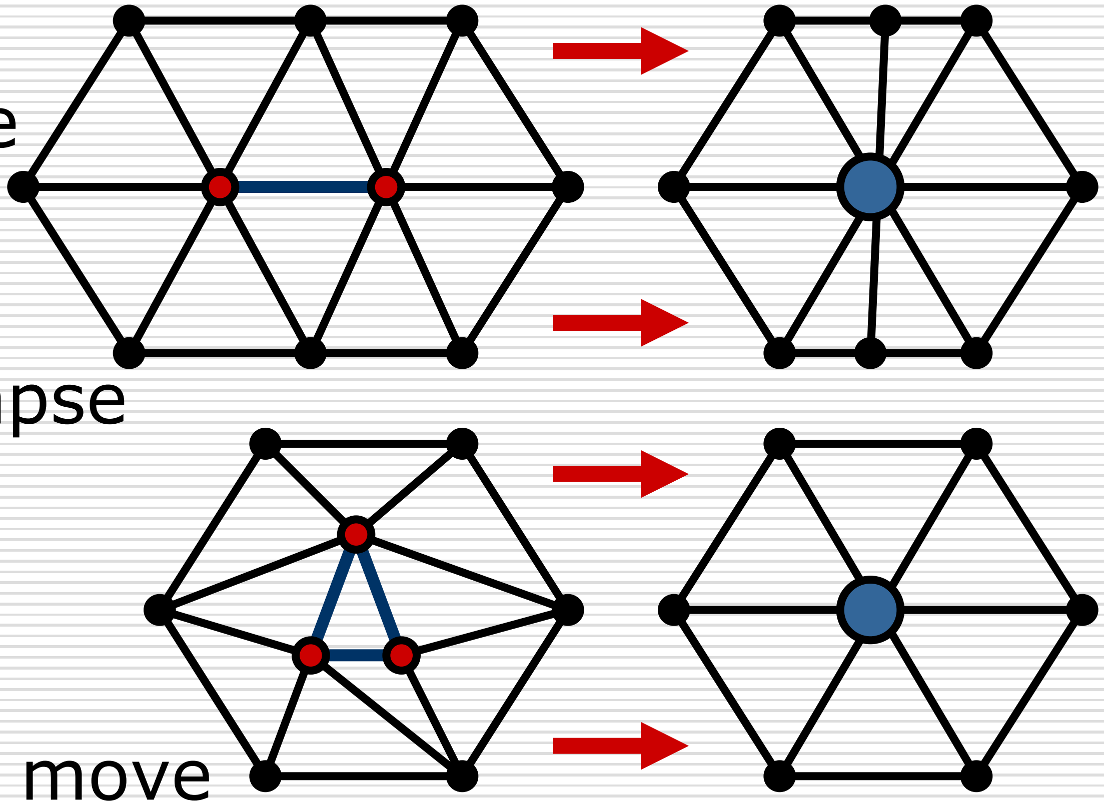


  - Remaining vertices - subset of original vertex set

Simplification Operations

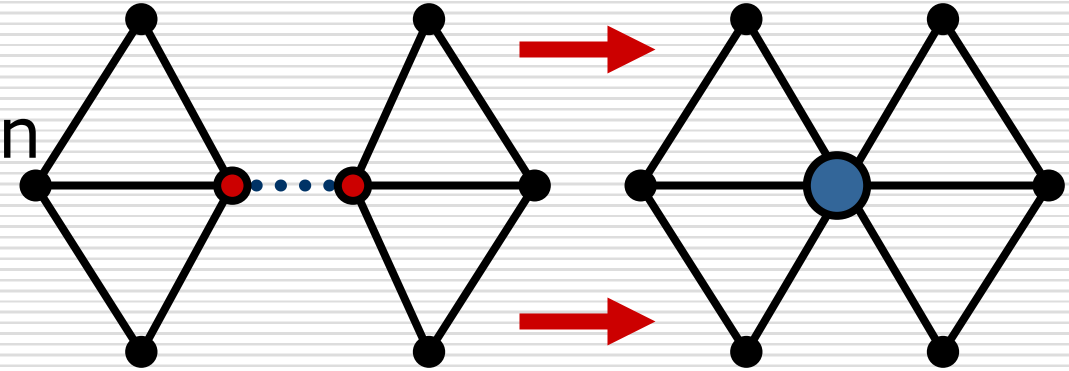
- Decimation
  - Edge collapse
    - v ← v-1
    - f ← f-2
  - Triangle collapse
    - v ← v-2
    - f ← f-4

  - Vertices may move

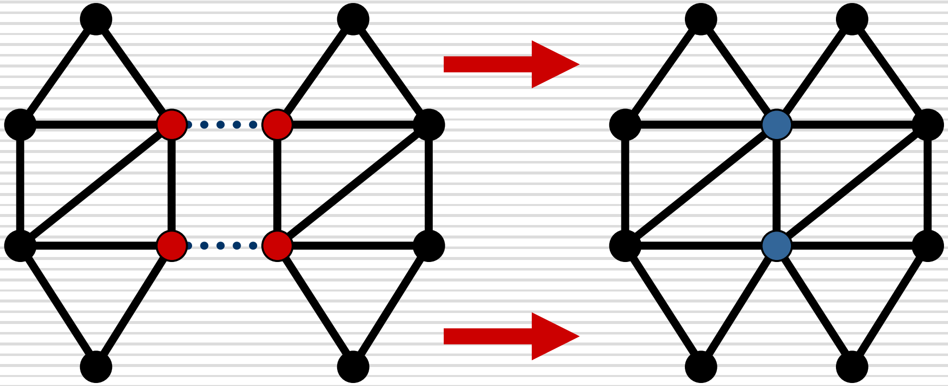


Simplification Operations

- Contraction
  - Pair contraction



  - Cluster contraction (set of vertices)

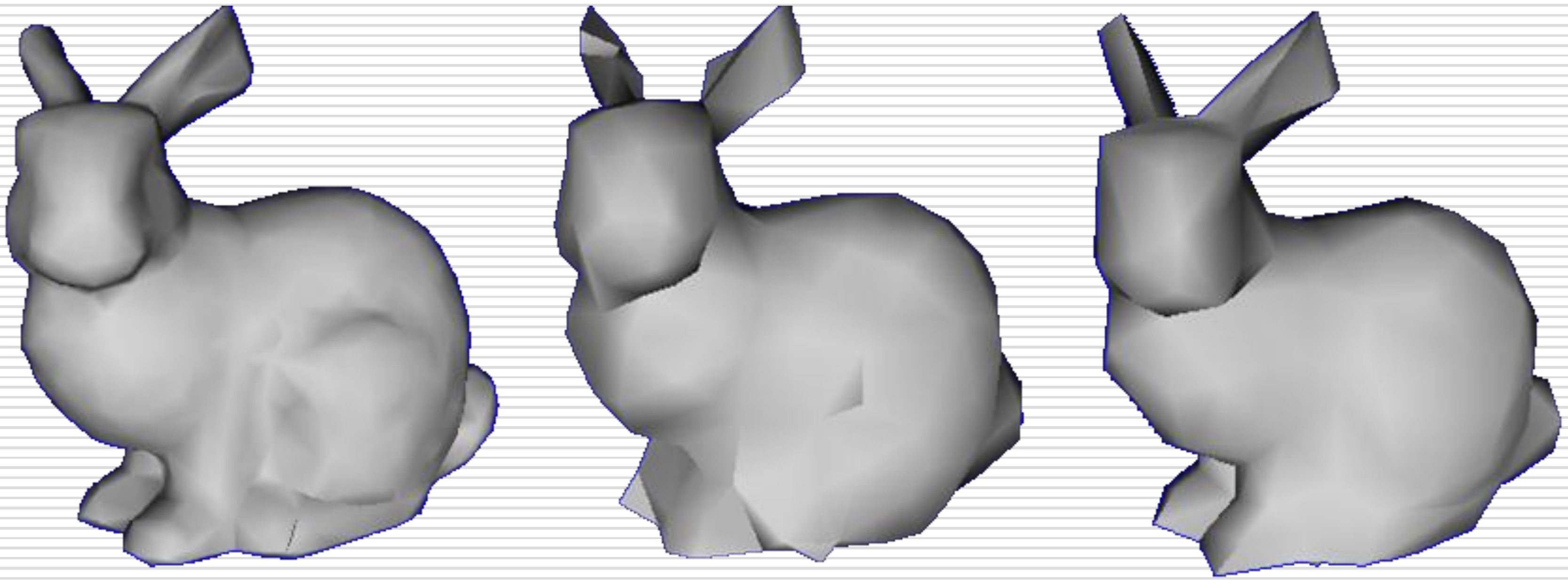


  - Vertices may move

Error Control

- Local error:
  - Compare new patch with previous iteration
    - Fast
    - Accumulates error
    - Memory-less
- Global error:
  - Compare new patch with original mesh
    - Slow
    - Better quality control
    - Can be used as termination condition
    - Must remember the original mesh throughout the algorithm

Local vs. Global Error



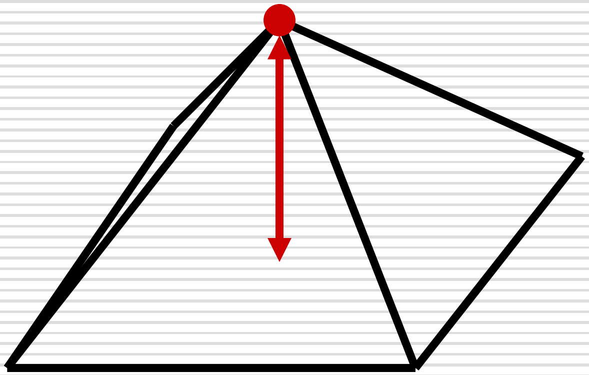
2000 faces

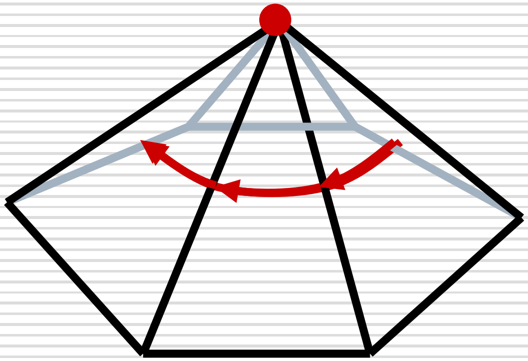
488 faces
(local error)

488 faces
(global error)

Simplification Error Metrics

- Measures
  - Distance to plane
  - Curvature
- Usually approximated
  - Average plane
  - Discrete curvature





$$\sum \alpha / 2\pi$$

The Basic Algorithm

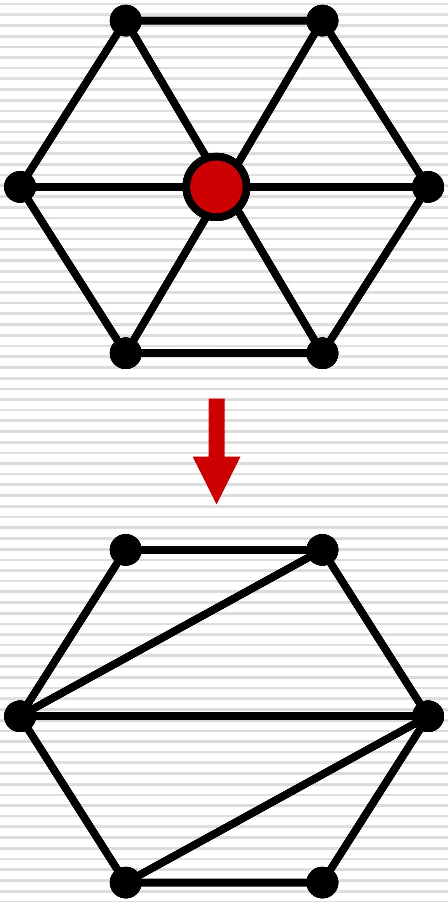
- Repeat
  - Select the element with minimal error
  - Perform simplification operation
    - (remove/contract)
  - Update error
    - (local/global)
- Until mesh size / quality is achieved

Implementation Details

- Vertices/Edges/Faces data structure
  - Easy access from each element to neighboring elements
- Use priority queue (e.g. heap)
  - Fast access to element with minimal error
  - Fast update

Vertex Removal Algorithm

- Simplification operation:
  - Vertex removal
- Error metric:
  - Distance to average plane
- May preserve mesh features (creases)

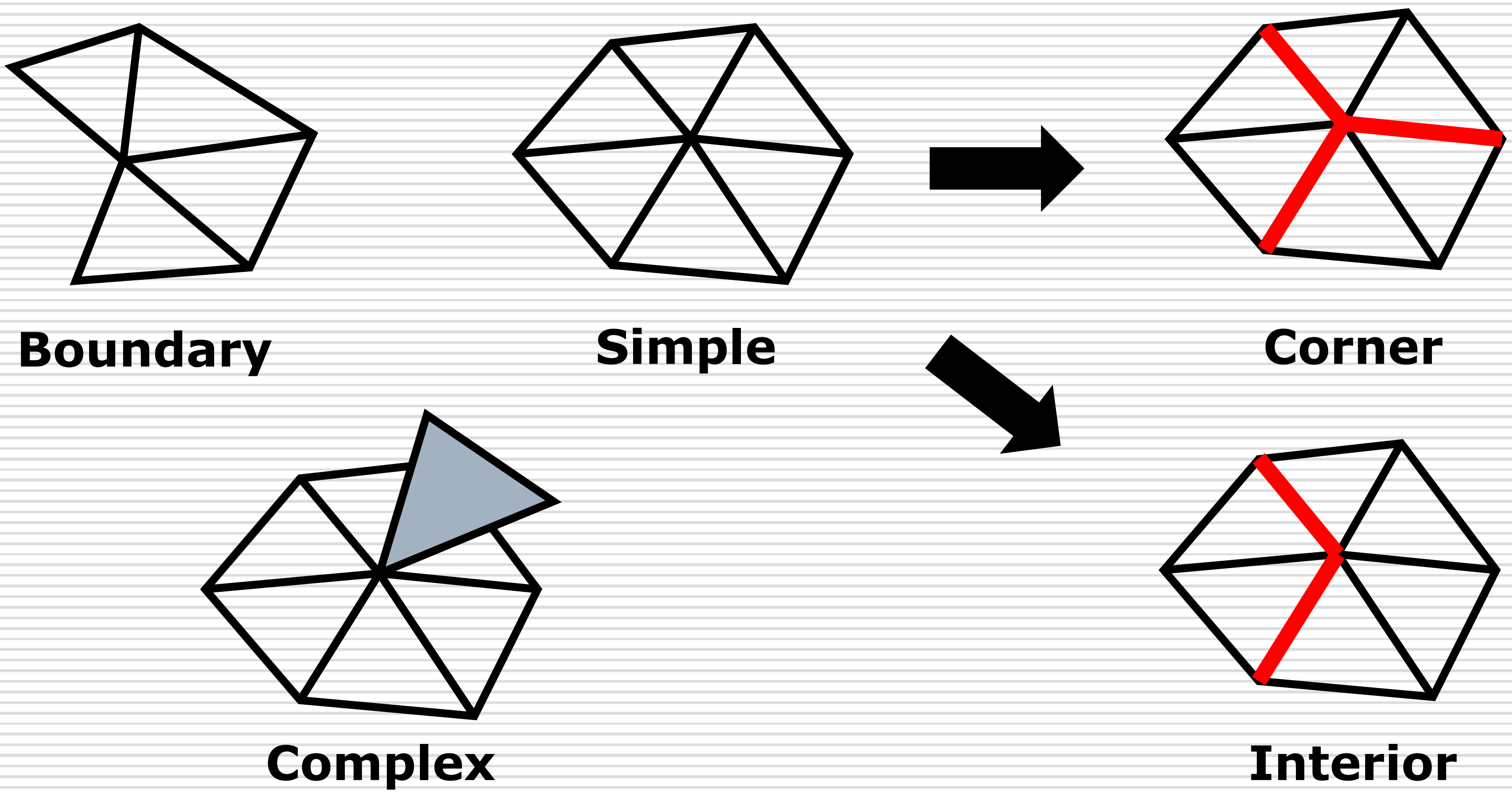


Algorithm Outline

- Characterize local topology/geometry
- Classify vertices as removable or not
- **Repeat**
  - Remove vertex
  - Triangulate resulting hole
  - Update error of affected vertices
- **Until** reduction goal is met

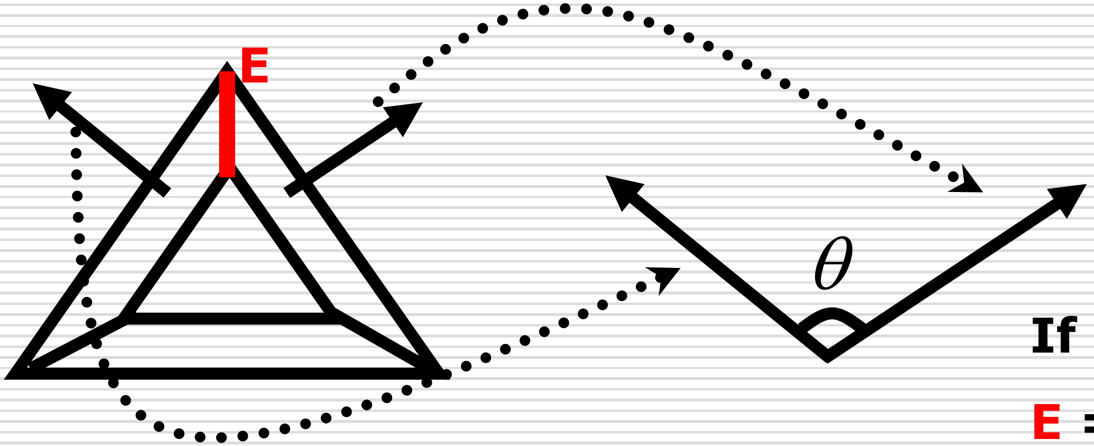
Characterizing Local Topology / Geometry

- Vertex Classification



Characterizing Local Topology / Geometry

- Feature edge exists if the angle between the surface normals of two adjacent triangles is greater than a user-specified "feature angle".



If $\theta > \theta_{fa}$

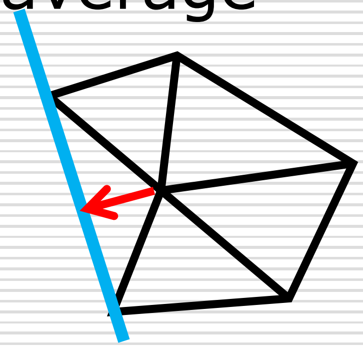
E = Feature Edge

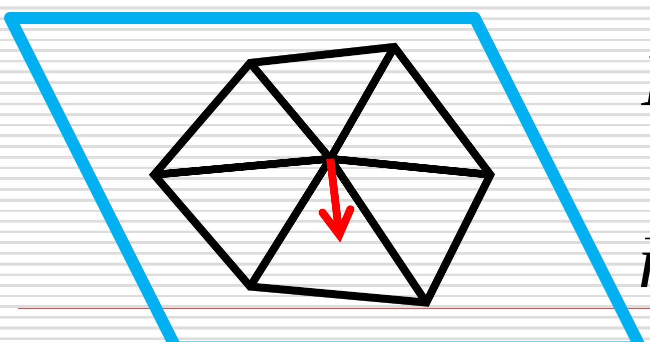
Characterizing Local Topology / Geometry

- Determine whether the given vertex is a potential candidate for deletion
- All vertices except complex vertices become candidates for deletion

Decimation Criterion

- E_{MAX} – user defined parameter
- Simple Vertex
  - Distance of vertex to the face loop average plane < E_{MAX}
- Boundary & Interior Vertex
  - Distance of vertex to the new boundary/edge < E_{MAX}





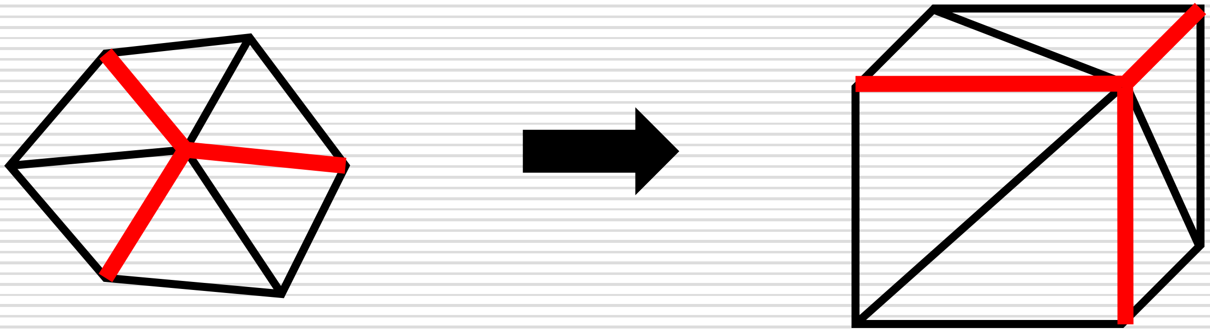
$$\vec{N} = \frac{\sum \vec{n}_i A_i}{\sum A_i}, \vec{n} = \frac{\vec{N}}{|\vec{N}|}, \vec{x} = \frac{\sum \vec{x}_i A_i}{\sum A_i}$$

$\vec{n}_i$:triangle normal, $\vec{x}_i$:centers, A_i:areas

$d = |\vec{n} \bullet (\vec{v} - \vec{x})|$, $\vec{v}$:vertex of considered

Decimation Criterion

- Corner Vertex ?



  - Corner vertices are usually not deleted to keep the sharp features.

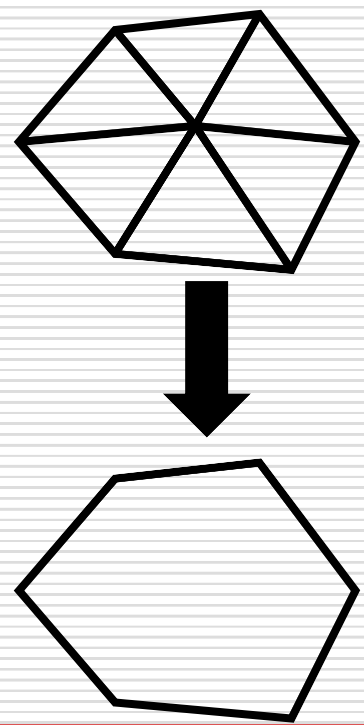
Triangulation

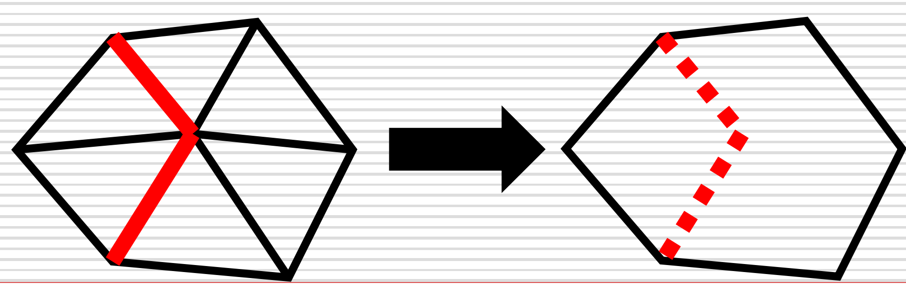
- If a vertex is eliminated, the loop created by removing the vertex is re-triangulated.
- Every loop is star shaped: recursive loop splitting triangulation schemes are used.
- If a loop cannot be re-triangulated, the vertex generating the loop is not removed.

Definition: A polygon *P* in which there exists an interior point *p* such that all the boundary points of *P* are *visible* from *p*.

Triangulation

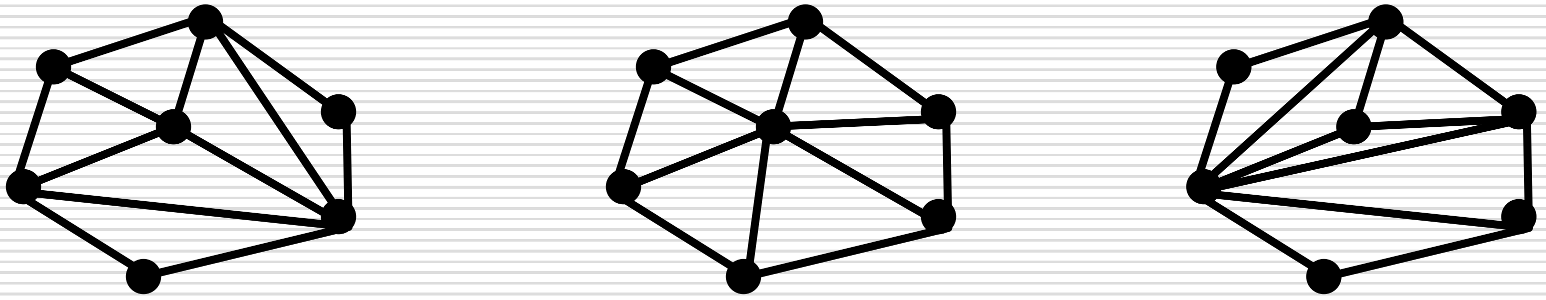
- After deleting a vertex and associated triangles create 1 or 2 loops
- 1 loop
  - Simple or Boundary Vertex
- 2 loops
  - Interior Edge Vertex





Triangulation

- A *triangulation* of set of points in the plane is a *partition* of the convex hull to triangles whose vertices are the points, and are empty of other points.
- There are an exponential number of triangulations of a point set.



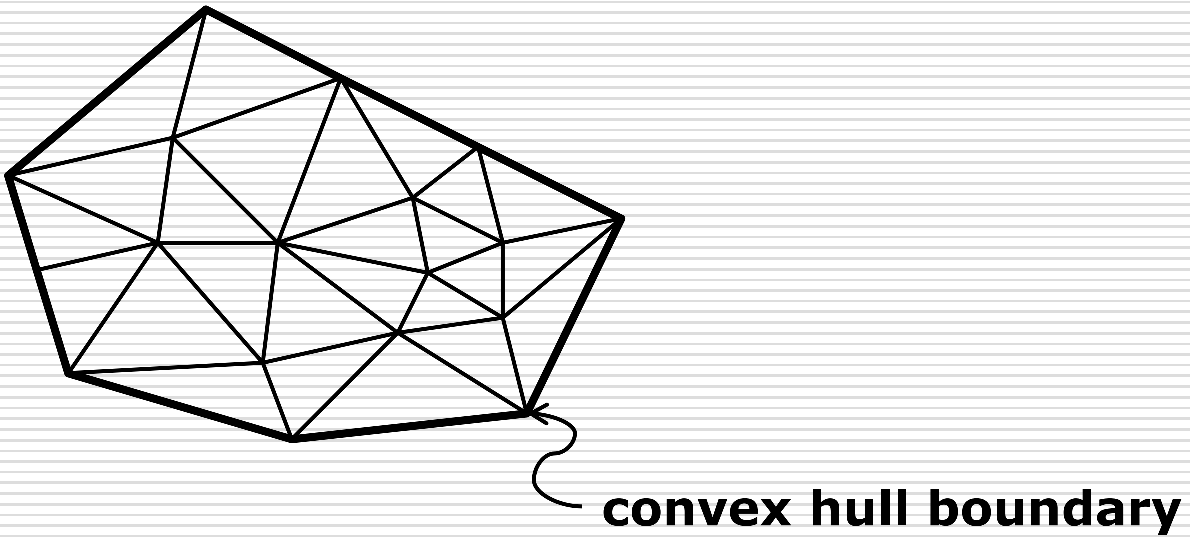
Definition: the minimal *convex set* containing a set of points *P*.

Triangulation

- Formal Definition
  - *maximal planar subdivision*
    - a subdivision S such that no edge connecting two vertices can be added to S without destroying its planarity
  - *triangulation* of set of points *P*
    - a maximal planar subdivision whose vertices are elements of *P*

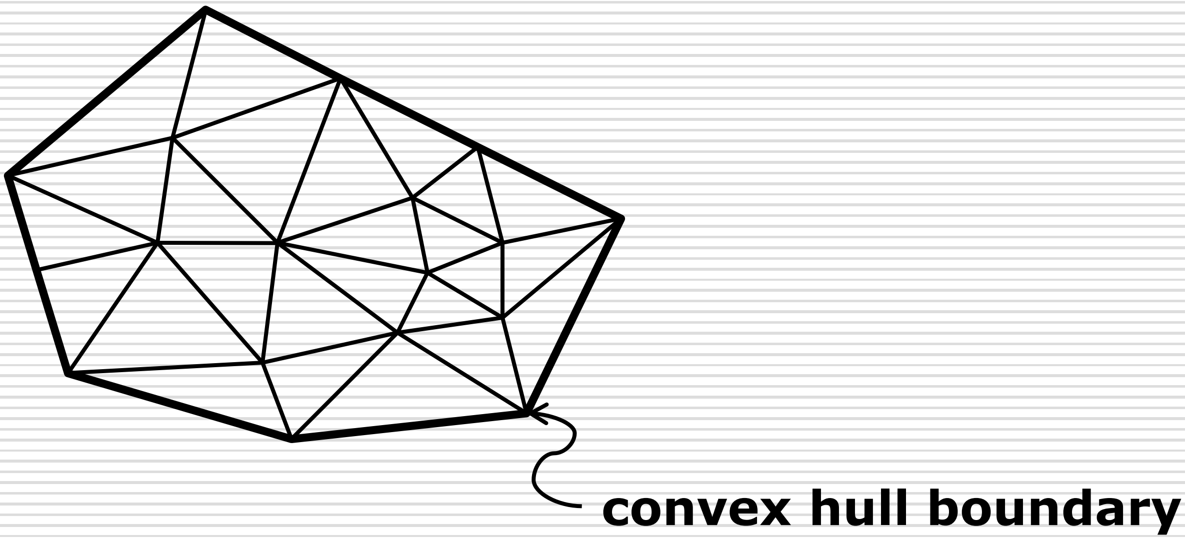
Triangulation

- Outer polygon must be convex hull
- Internal faces must be triangles, otherwise they could be triangulated further



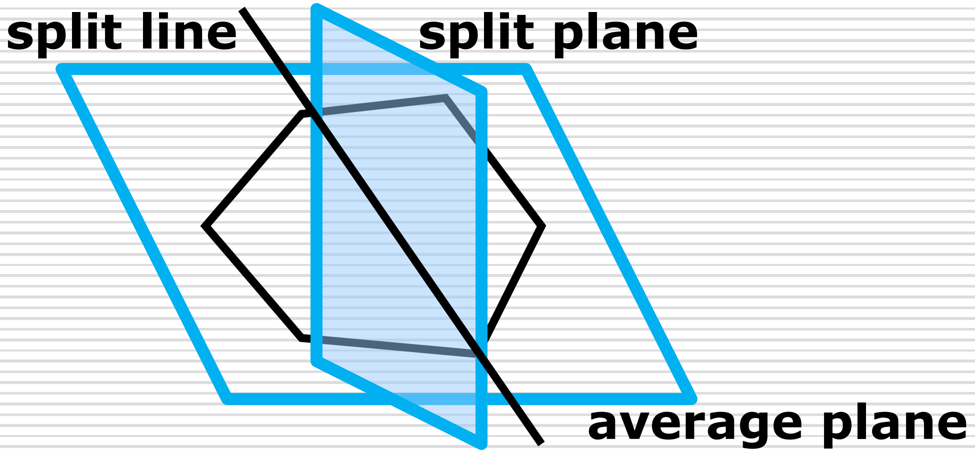
Triangulation

- For P consisting of n points, all triangulations contain $2n$-2-k triangles and $3n$-3-k edges
  - n = number of points in P
  - k = number of points on convex hull of P



Recursive Splitting Triangulation

- A split plane orthogonal to average plane is determined.



- If two loops do not overlap, the split plane is acceptable.

Recursive Splitting Triangulation

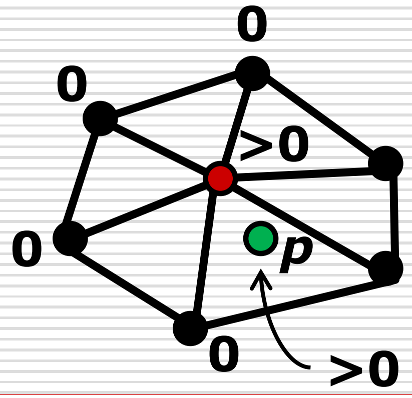
- Best splitting plane is determined using an aspect ratio:

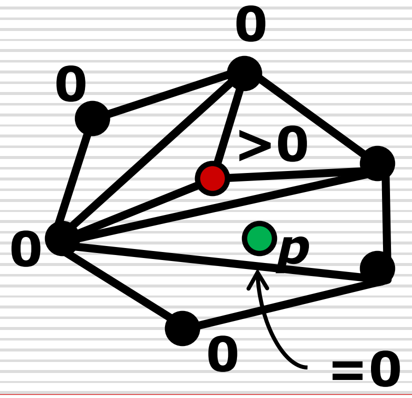
$$\frac{\text{minimum distance of the loop vertices to the split line}}{\text{the length of the split line}}$$

- Maximum aspect ratio gives best splitting plane.

Piecewise Linear Interpolation

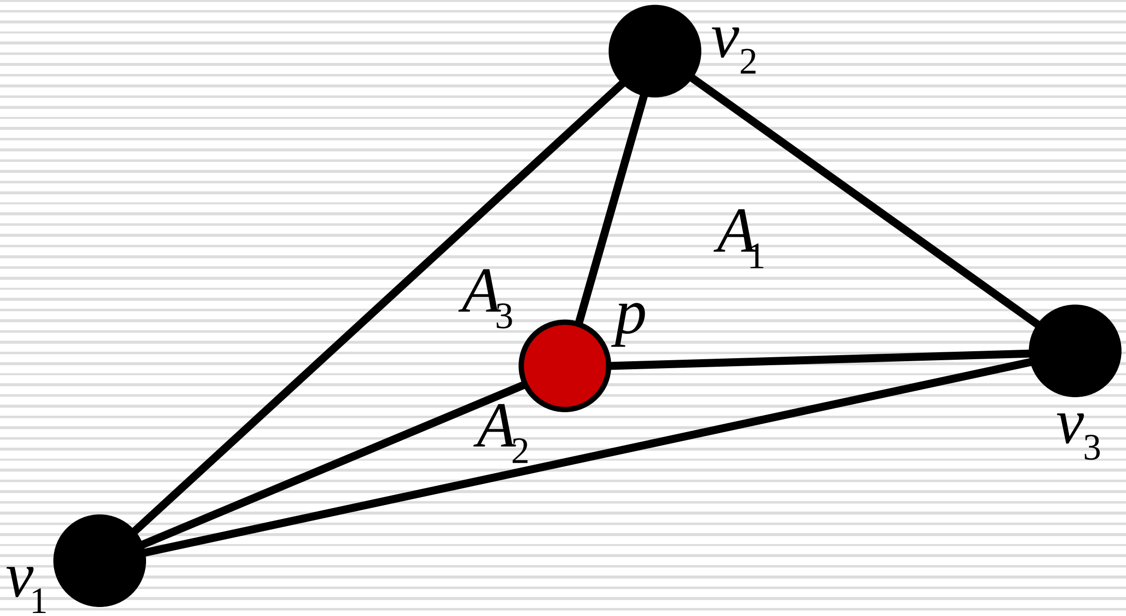
- The height of a point *p* inside a triangle is determined by the height of the triangle vertices, and the location of *p*.
- The result depends on the triangulation.





Barycentric Coordinates

- Any point inside a triangle can be expressed *uniquely* as a *convex* combination of the triangle vertices.



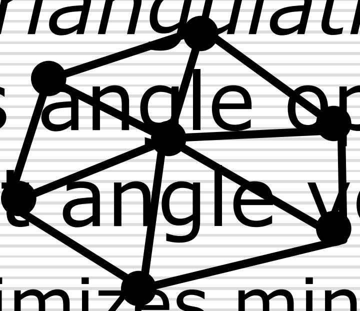
$$p = a_1 v_1 + a_2 v_2 + a_3 v_3$$

$$a_i = \frac{A_i}{A_1 + A_2 + A_3}$$

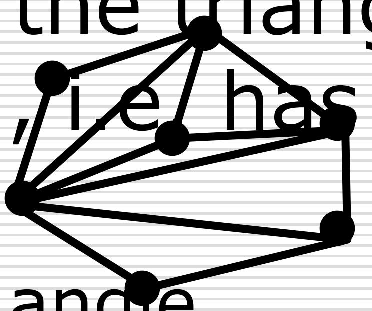
$$a_i \geq 0, a_1 + a_2 + a_3 = 1$$

Quality Triangulation

- Let $A(T) = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ be the *angle vector* in the triangulation T, in increasing order.
- $A(T) > A(T')$ iff there exists an i such that $\alpha_j = \alpha'_j$ for all $j < i$ $\alpha_i > \alpha'_i$
- *Best triangulation* is the triangulation that is angle optimal, i.e. has the largest angle vector.
  - Maximizes minimum angle.



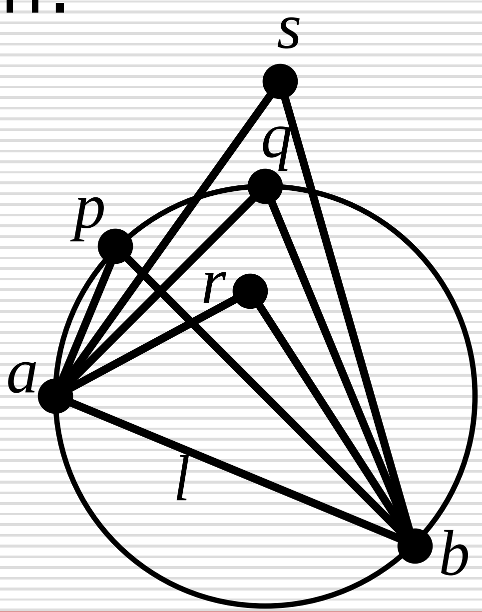
good



bad

Thales' Theory

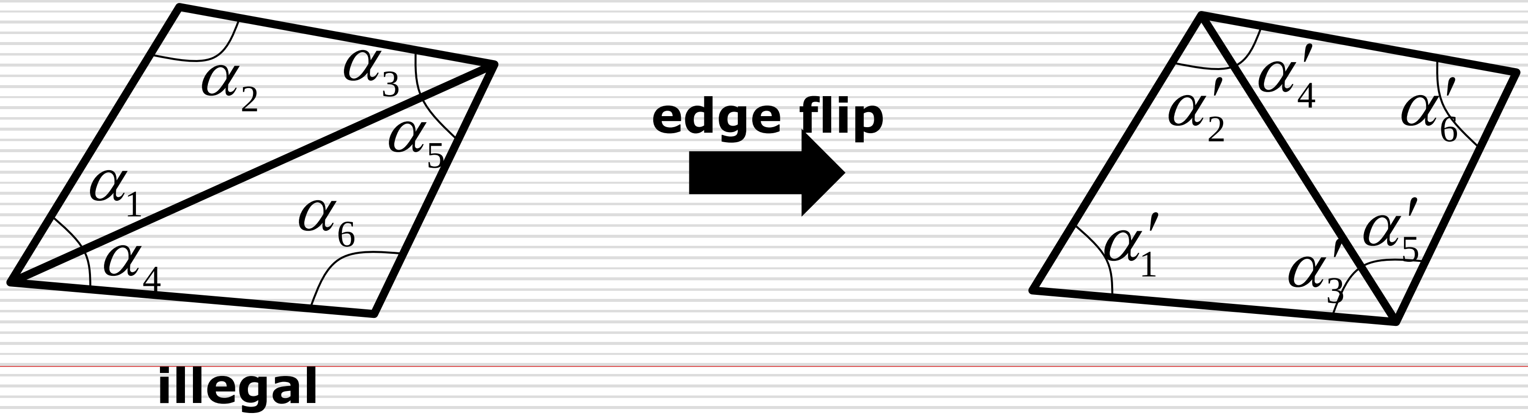
- Let C be a circle, and l be a line intersecting C at points a and b. Let p, q, r and s be points lying on the same side of l, where p and q are on C, r inside C and s outside C. Then:



$$\angle arb > \angle apb = \angle aqb > \angle asb$$

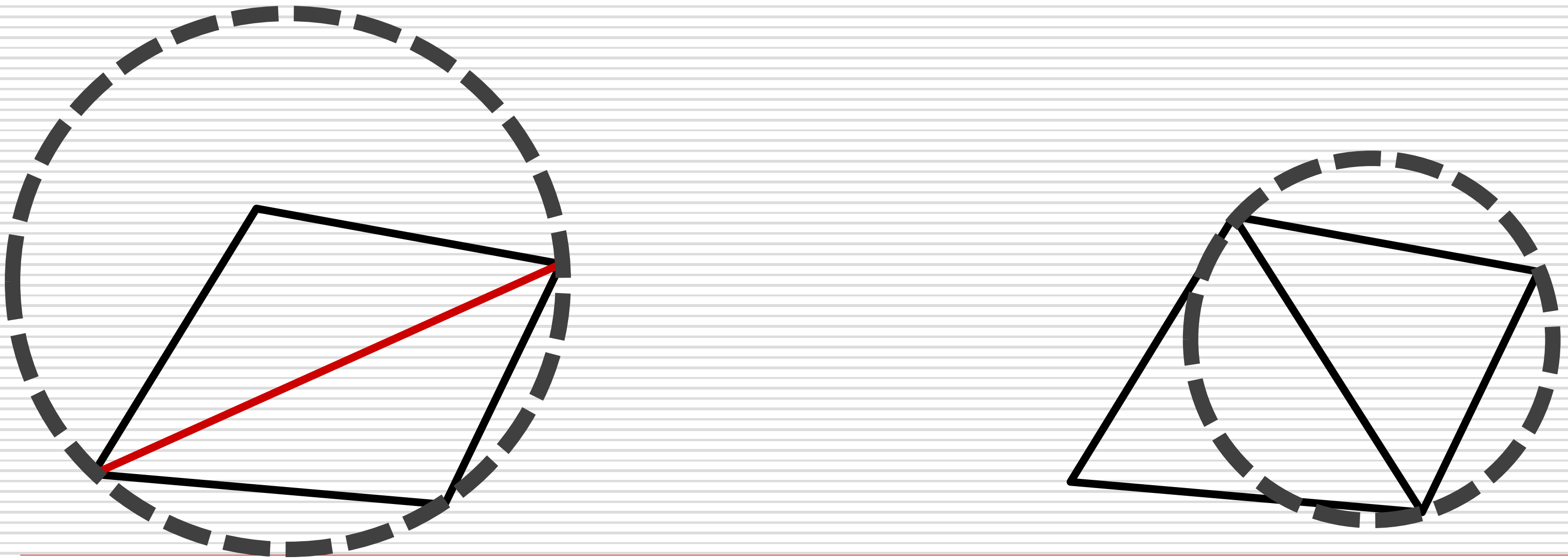
Improving a Triangulation

- Consider two adjacent triangles of *T*:
- If the two triangles form a convex quadrilateral, we could have an alternative triangulation by performing an edge flip on their shared edge.



Illegal Edges

- Lemma: An edge is illegal iff one of its opposite vertices is inside the circle defined by the other three vertices.
- Proof: By Thales' theorem.

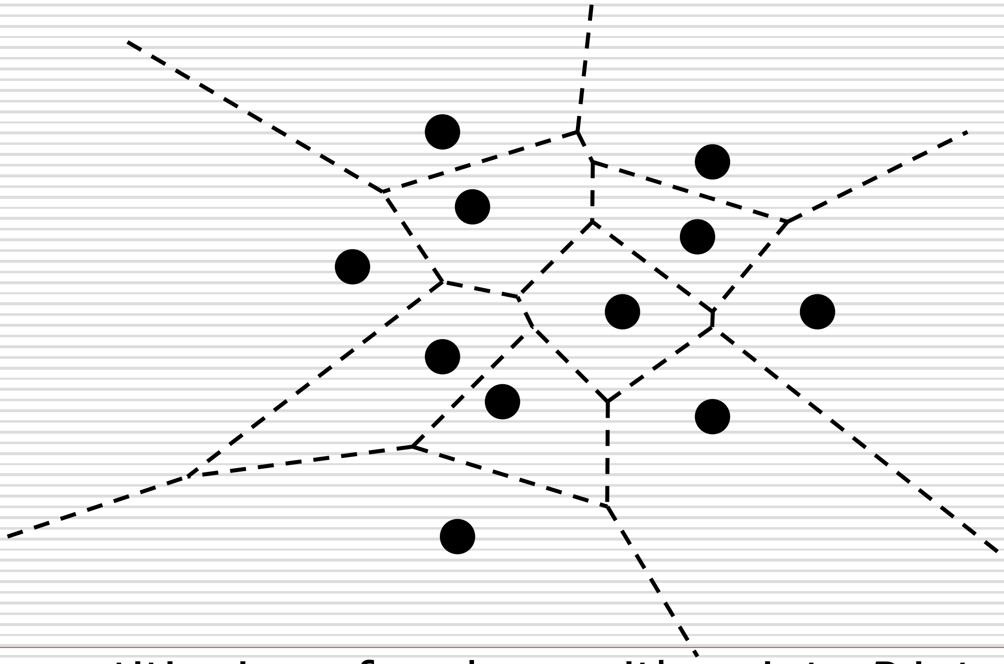


Illegal Edges

- Theorem: A Delaunay triangulation does not contain illegal edges.
- Corollary: A triangle is Delaunay iff the circle through its vertices is empty of other sites (the *empty-circle* condition).
- Corollary: The Delaunay triangulation is not unique if more than three sites are co-circular.

Delaunay Graph & Voronoi Diagram

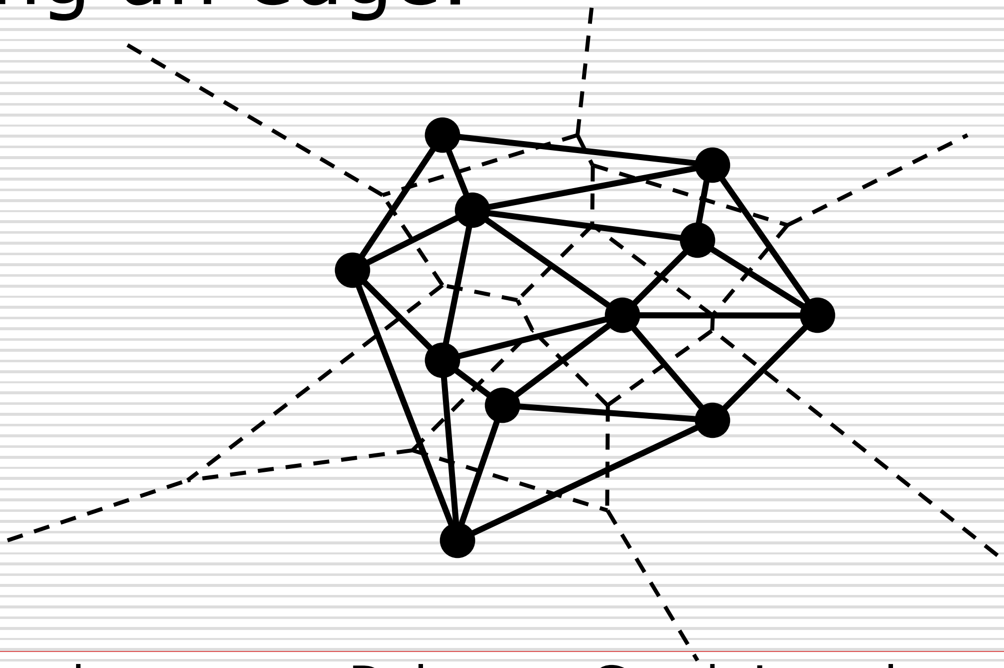
- Delaunay Graph of a set of points *P* is the dual graph of the Voronoi Diagram of *P*



Definition: the partitioning of a plane with points *P* into *convex polygons* such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other.

Delaunay Graph

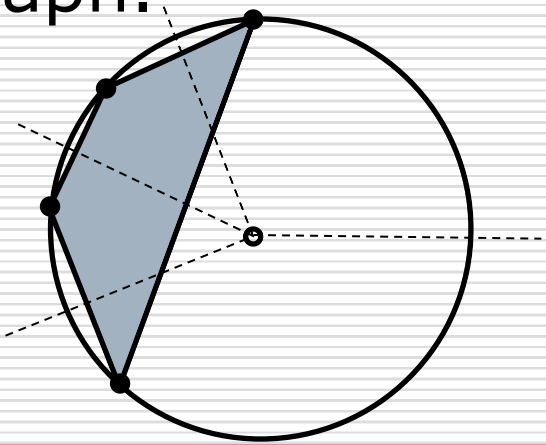
- Constructing Delaunay Graph by connecting the adjacent vertices sharing an edge.



Note: no two edges cross; Delaunay Graph is a planar graph.

Delaunay Triangulation

- Some sets of more than 3 points of Delaunay graph may lie on the same circle.
- These points form empty convex polygons, which can be triangulated.
- *Delaunay Triangulation* is a triangulation obtained by adding 0 or more edges to the Delaunay Graph.

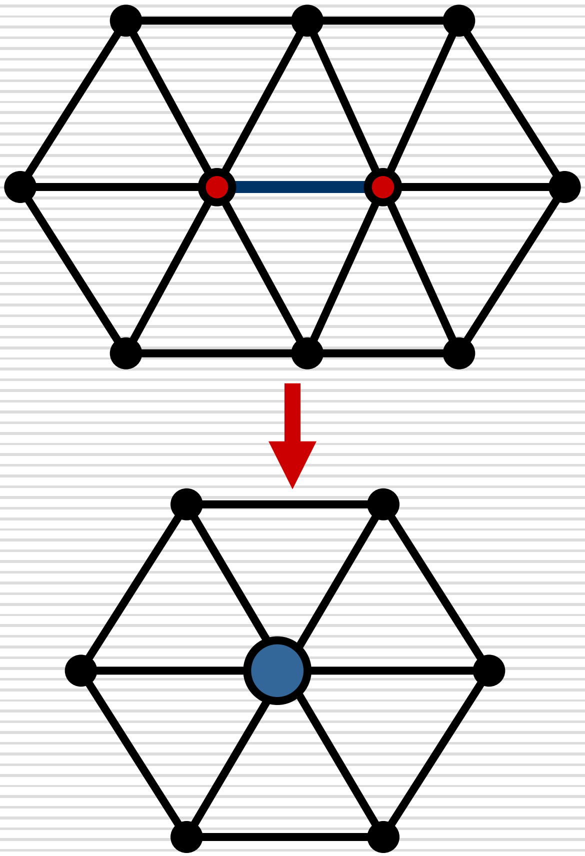


Pros and Cons

- Pros:
  - Efficient
  - Simple to implement and use
    - Few input parameters to control quality
  - Reasonable approximation
  - Works on very large meshes
  - Preserves topology
  - Vertices are a subset of the original mesh
- Cons:
  - Error is not bounded
    - Local error evaluation causes error to accumulate

Edge Collapse Algorithm

- Simplification operation:
  - Pair contraction
- Error metric:
  - distance, pseudo-global
- Simplifies also topology

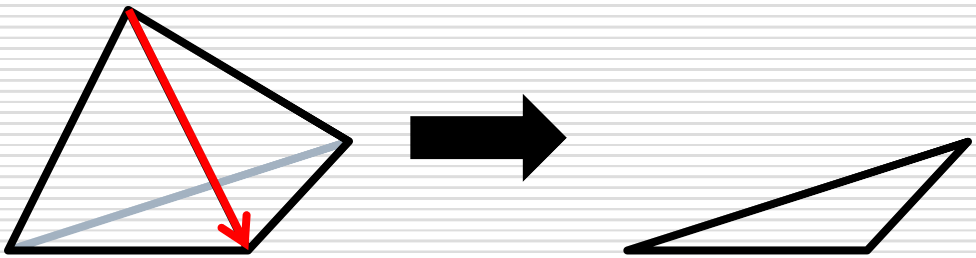


Pros and Cons

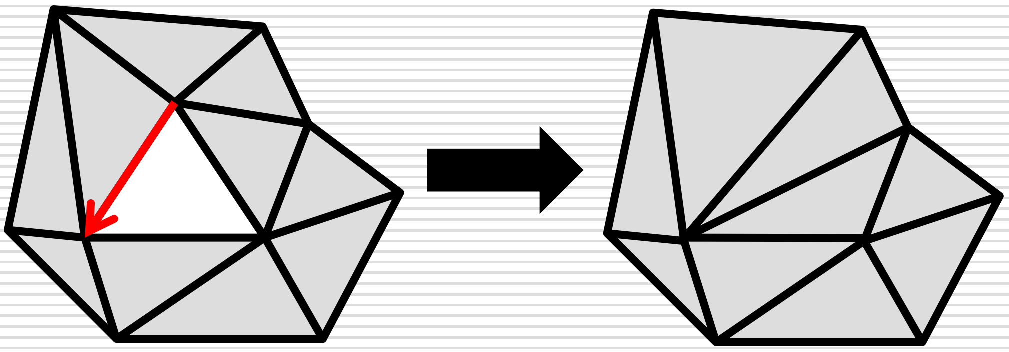
- Pros
  - Error is bounded
  - Allows topology simplification
  - High quality result
  - Quite efficient
- Cons
  - Difficulties along boundaries
  - Difficulties with coplanar planes
  - Introduces new vertices not present in the original mesh

Special Cases

- Modification of topology of a closed structure

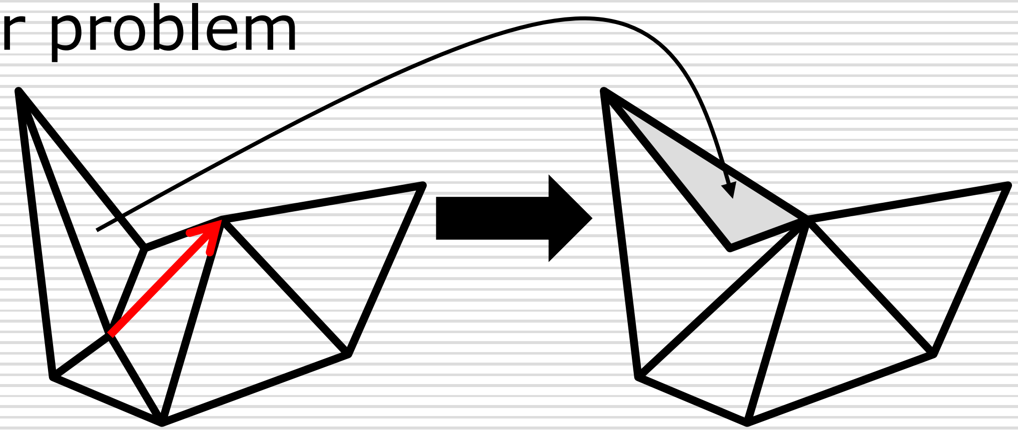


- Topological ‘holes’ problem

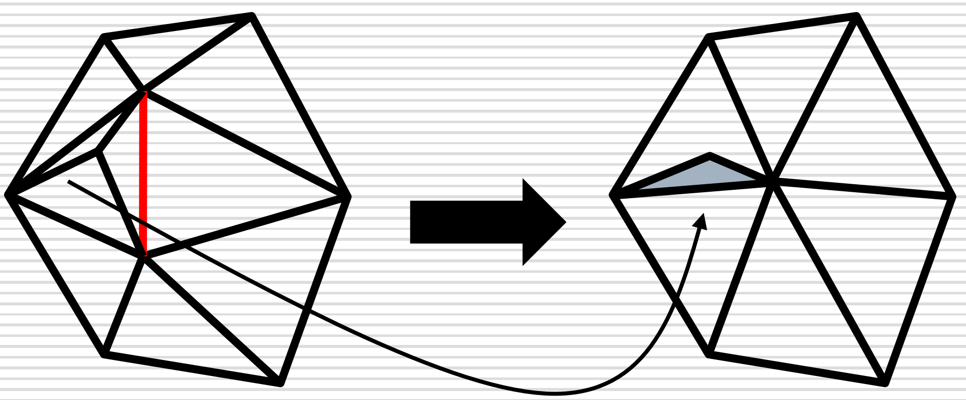


Special Cases

- Foldover problem



- Topological inconsistency problem



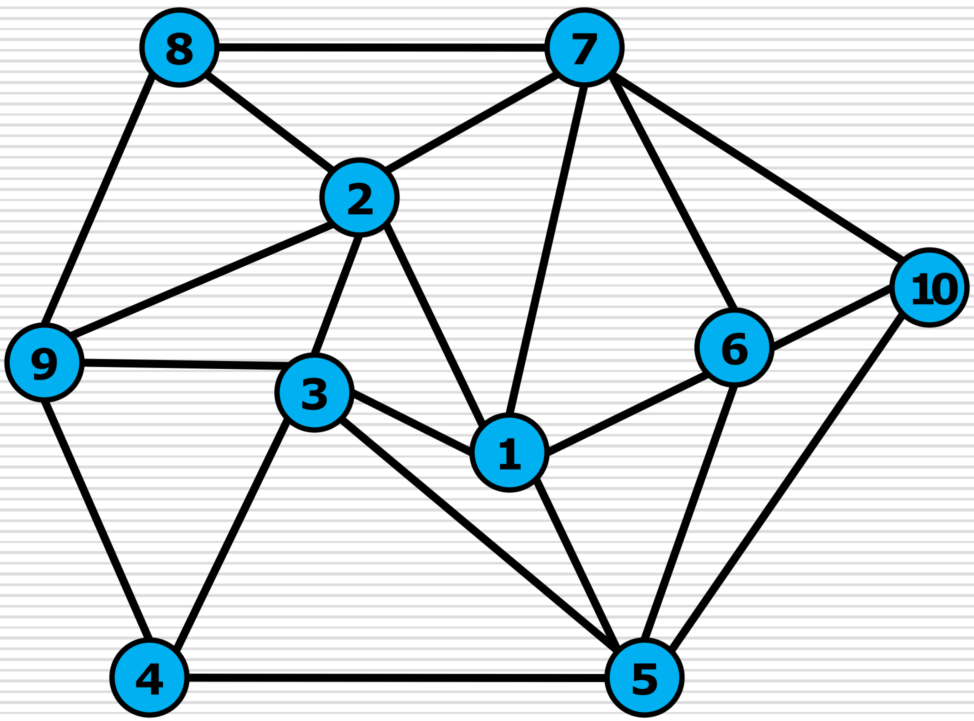
Vertex Tree & Active Triangle List

- The Vertex Tree
  - represents the entire model
  - a hierarchical clustering of vertices
  - queried each frame for updated scene
- The Active Triangle List
  - represents the current simplification
  - list of triangle to be displayed

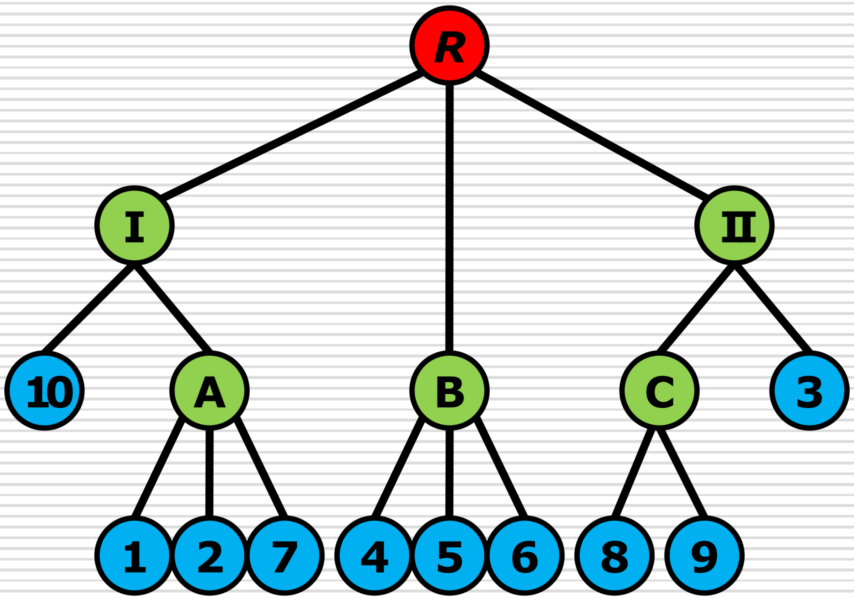
The Vertex Tree

- Each vertex tree node contains:
  - a subset of model vertices
  - a representative vertex or repvert
- *Folding* a node collapses its vertices to the repvert
- *Unfolding* a node splits the repvert back into vertices

Vertex Tree Example

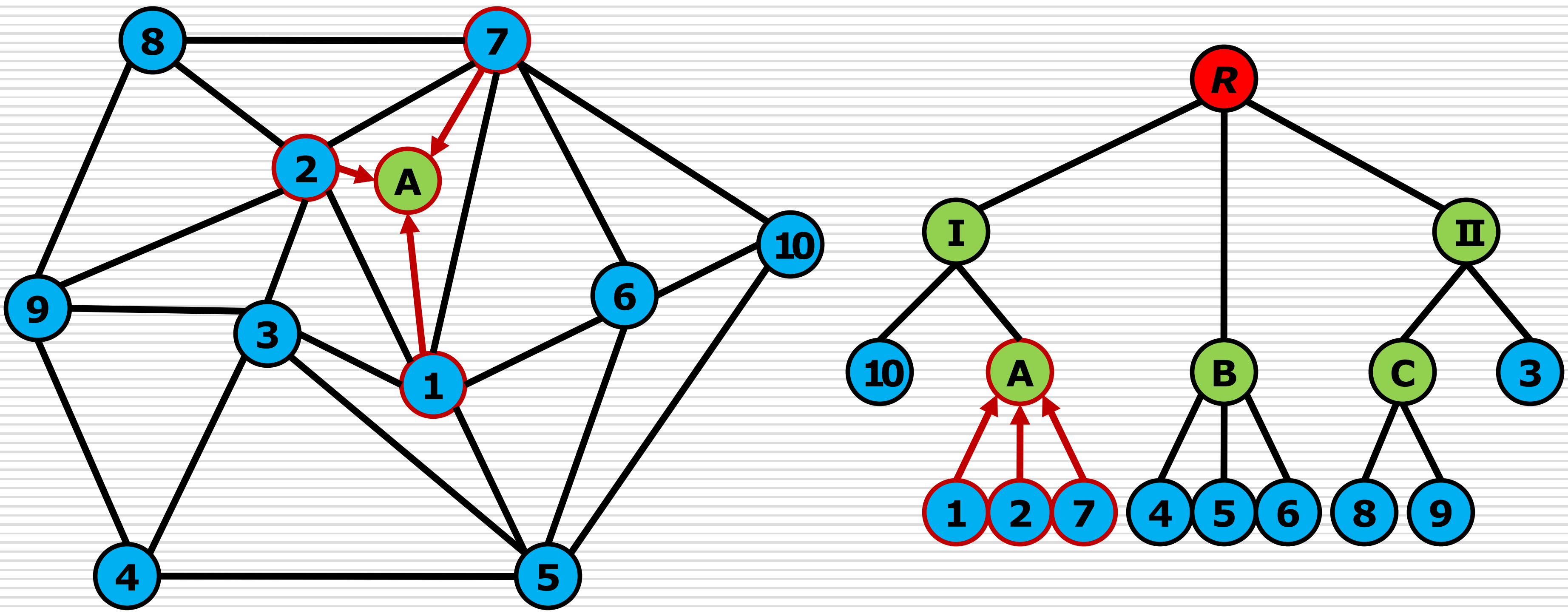


Triangles in Active List



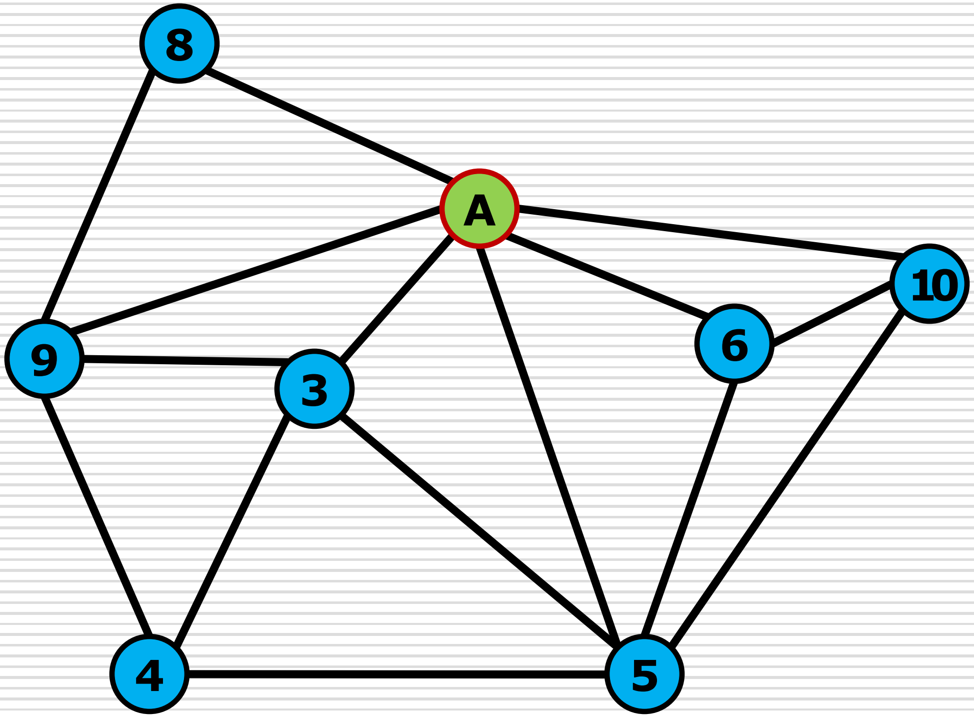
Vertex Tree

Vertex Tree Example

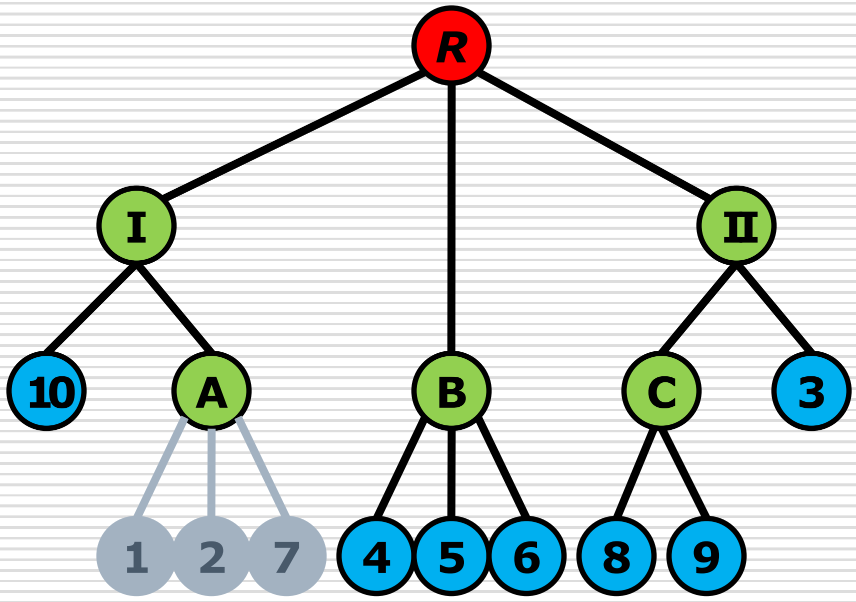


Triangles in Active List **Vertex Tree**

Vertex Tree Example

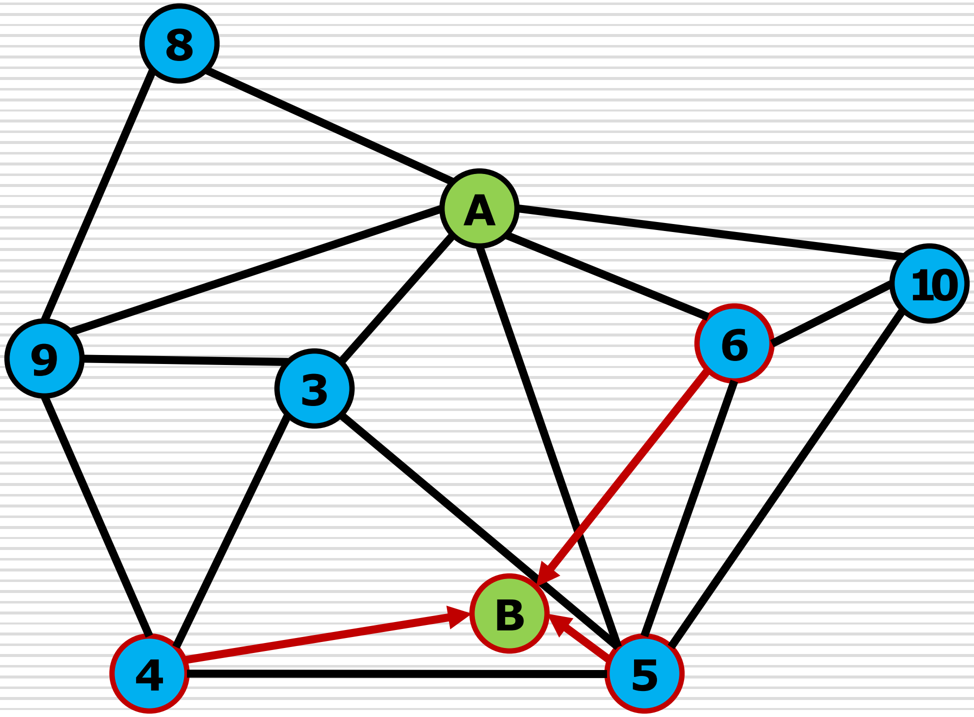


Triangles in Active List

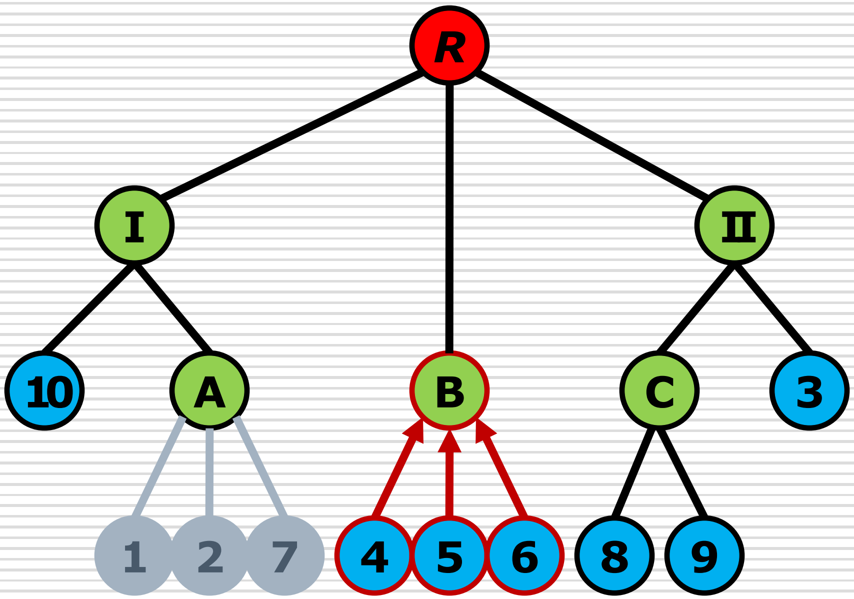


Vertex Tree

Vertex Tree Example

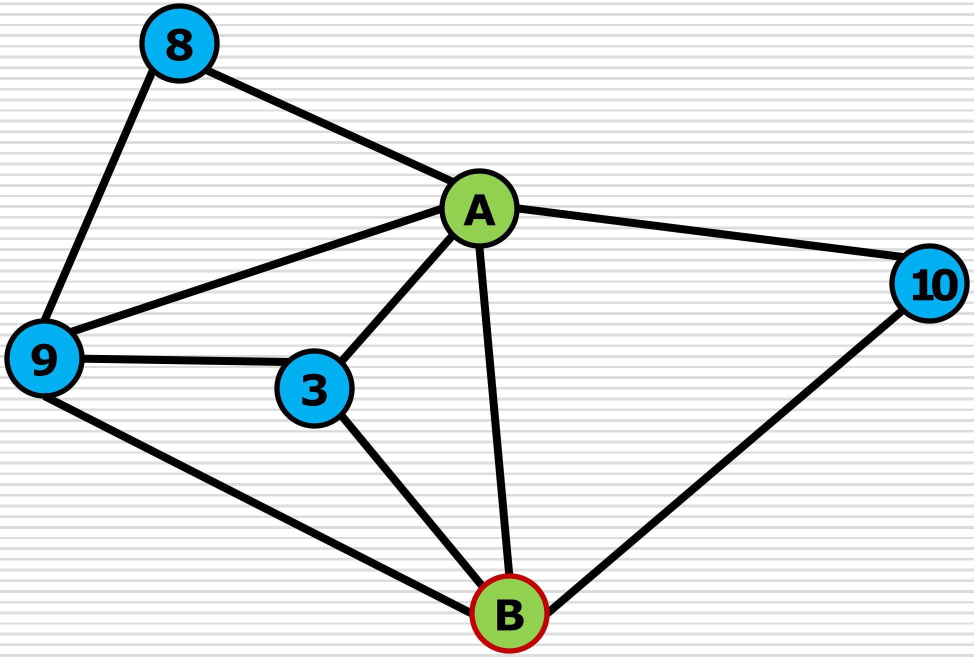


Triangles in Active List

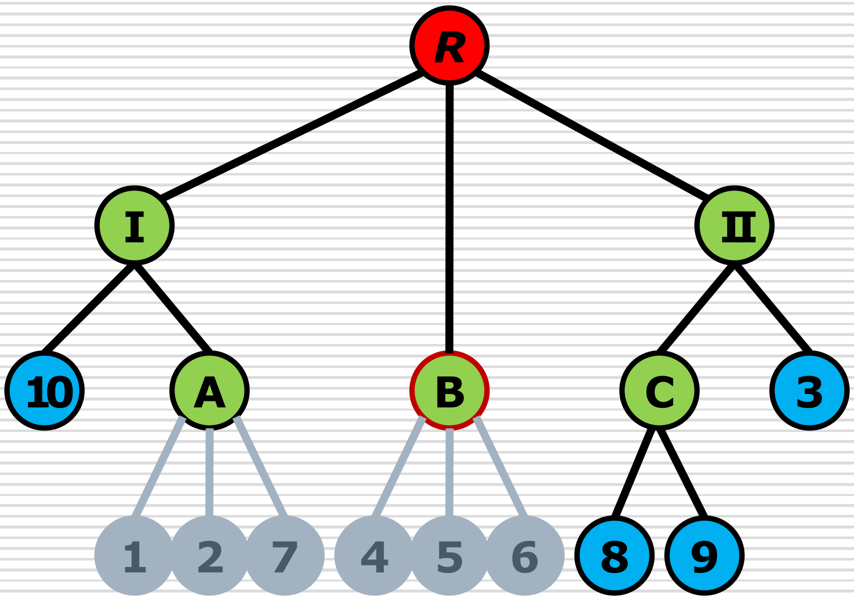


Vertex Tree

Vertex Tree Example

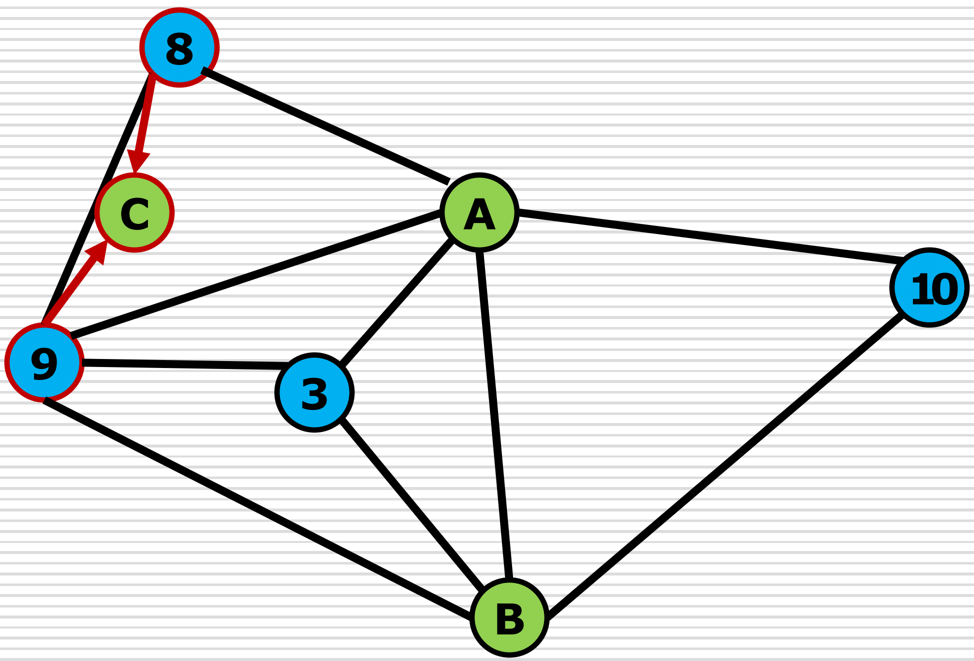


Triangles in Active List

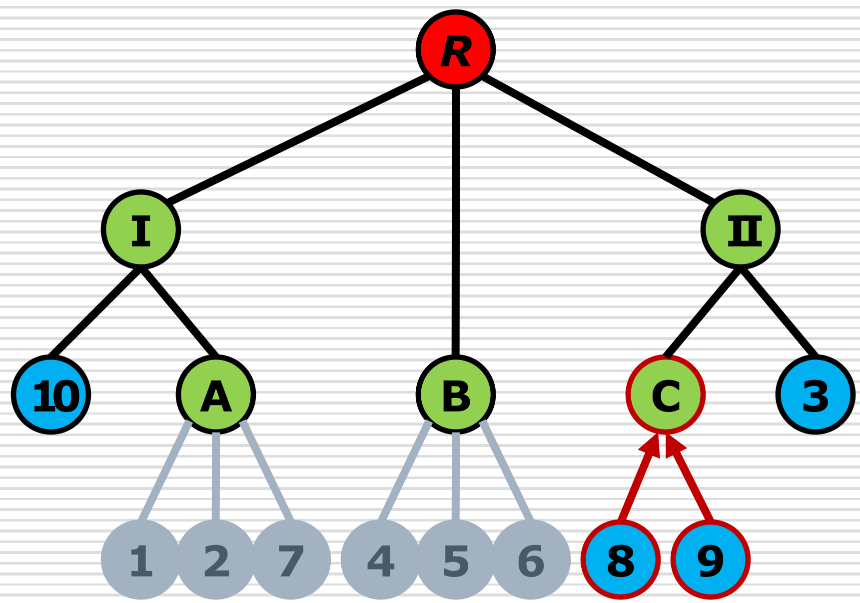


Vertex Tree

Vertex Tree Example

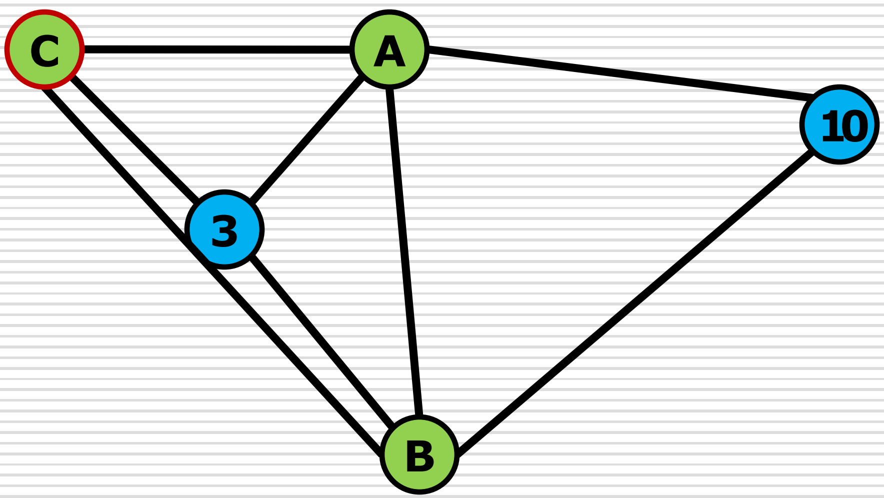


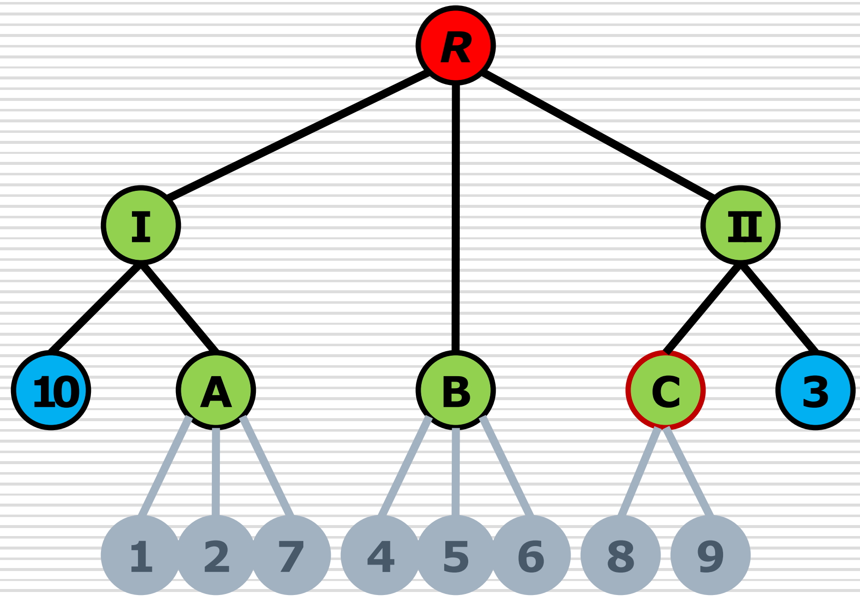
Triangles in Active List



Vertex Tree

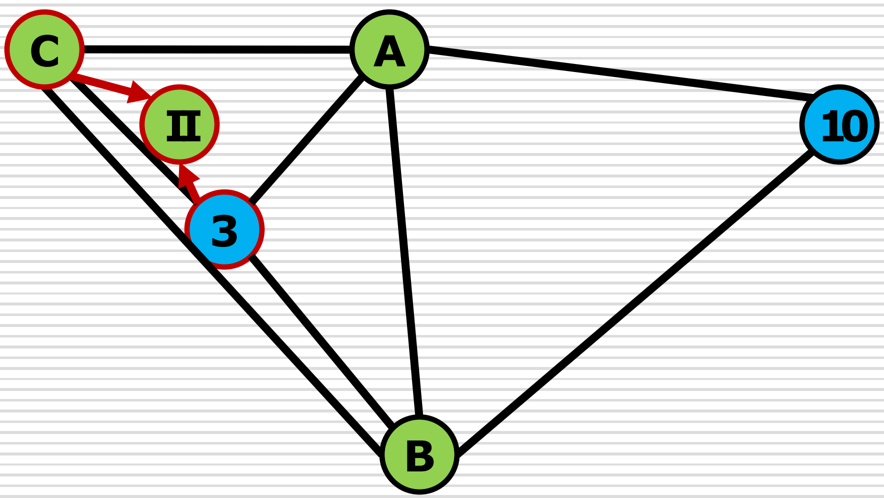
Vertex Tree Example



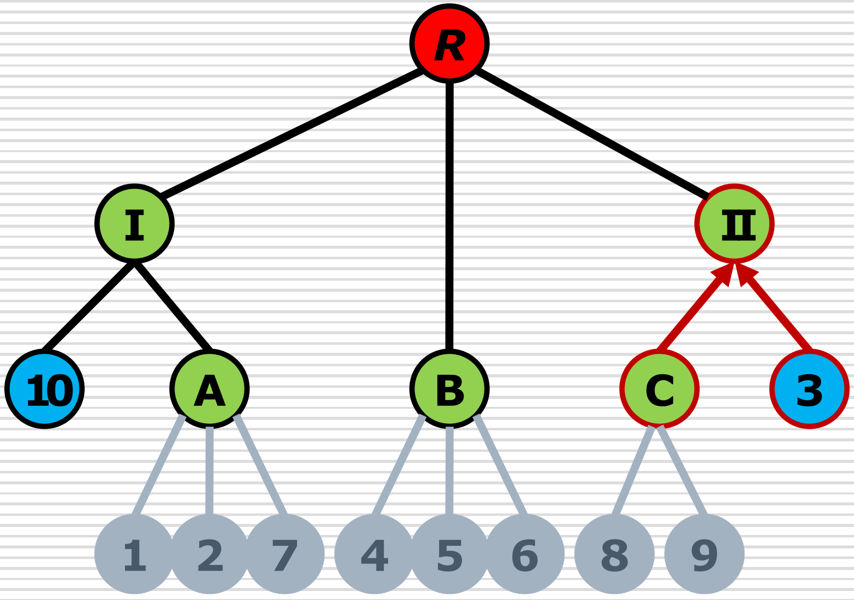


Triangles in Active List **Vertex Tree**

Vertex Tree Example

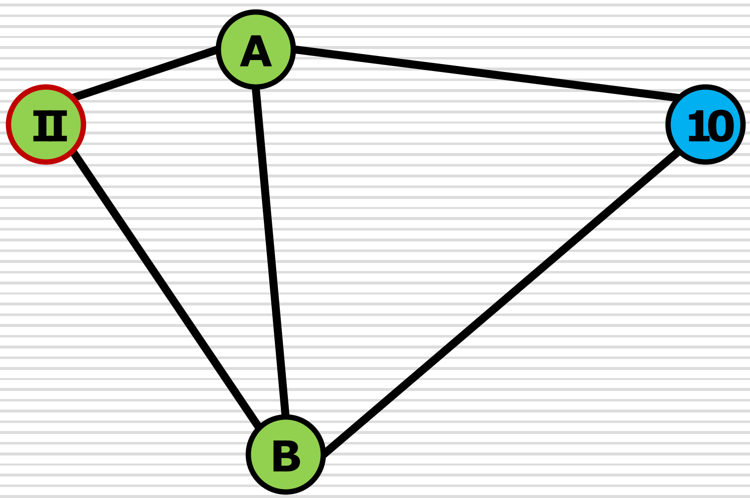


Triangles in Active List

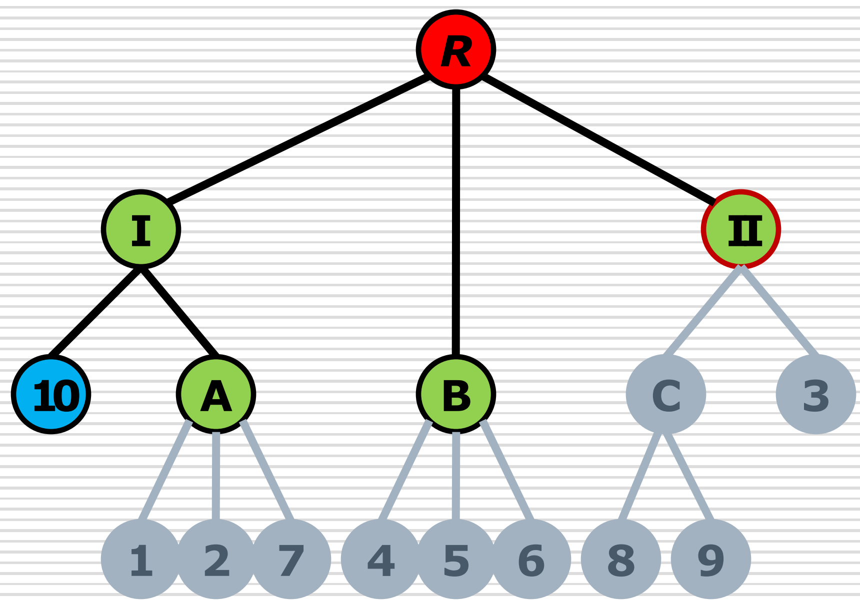


Vertex Tree

Vertex Tree Example

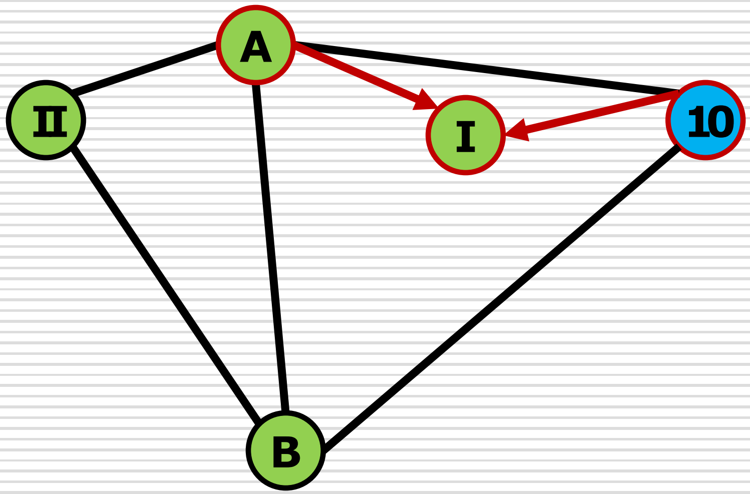


Triangles in Active List

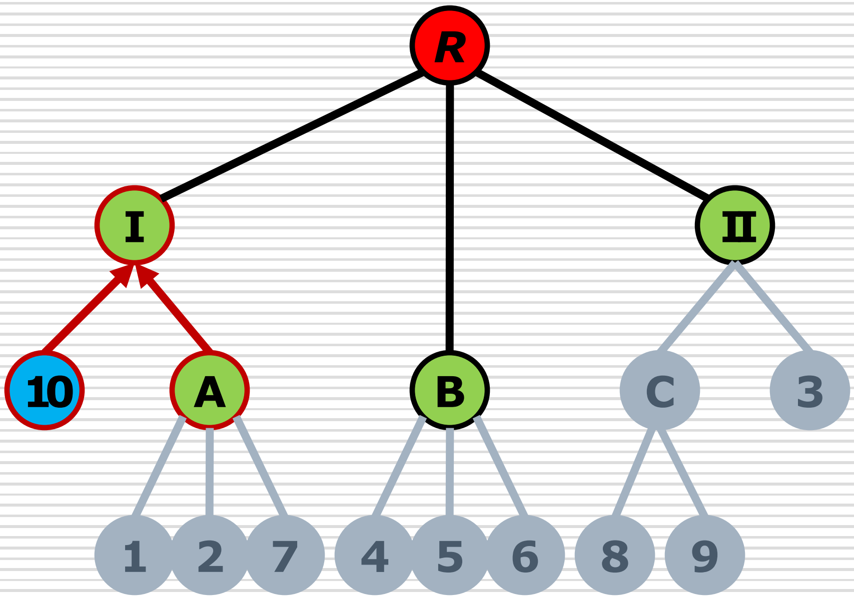


Vertex Tree

Vertex Tree Example

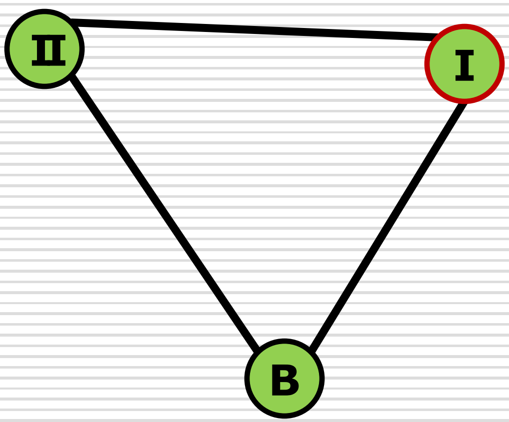


Triangles in Active List

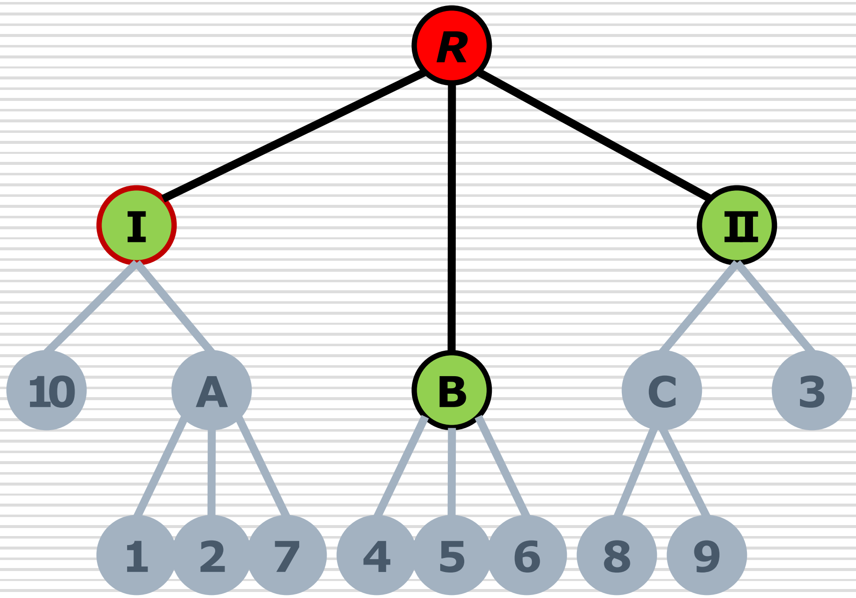


Vertex Tree

Vertex Tree Example

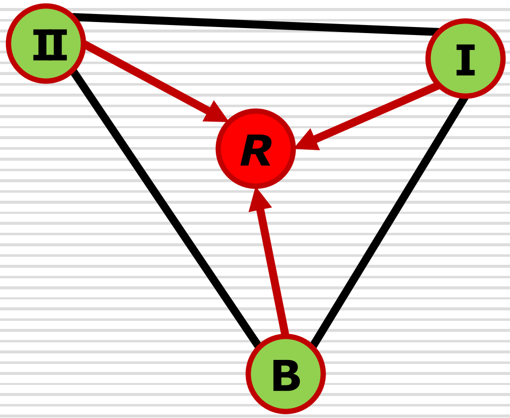


Triangles in Active List

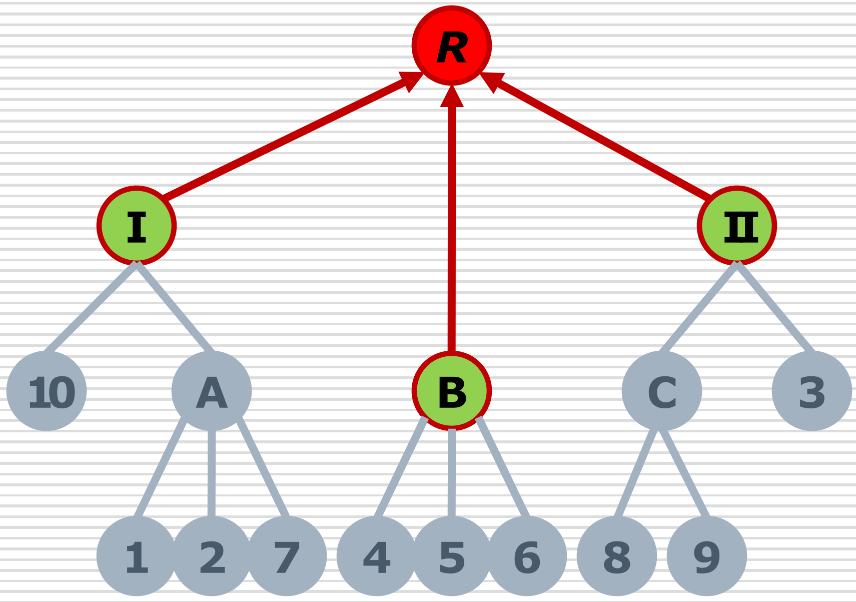


Vertex Tree

Vertex Tree Example

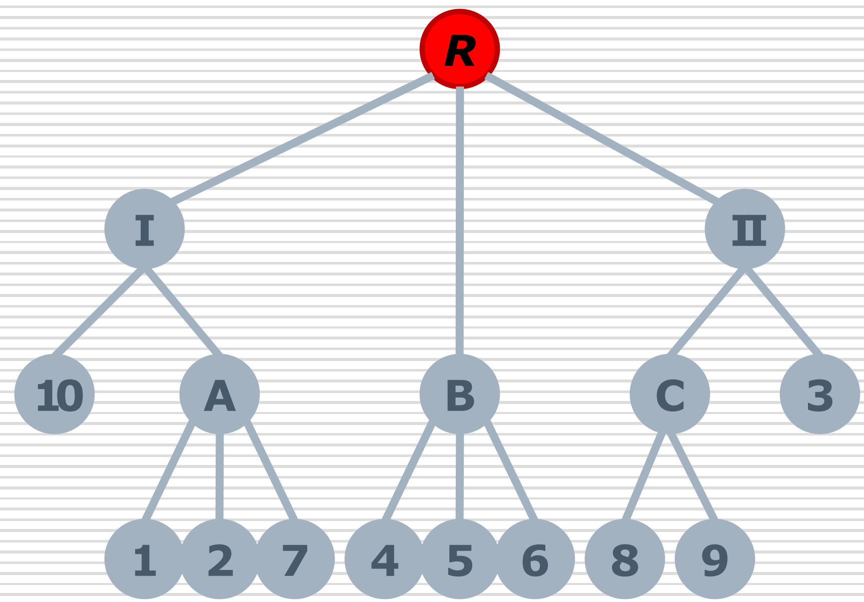


Triangles in Active List



Vertex Tree

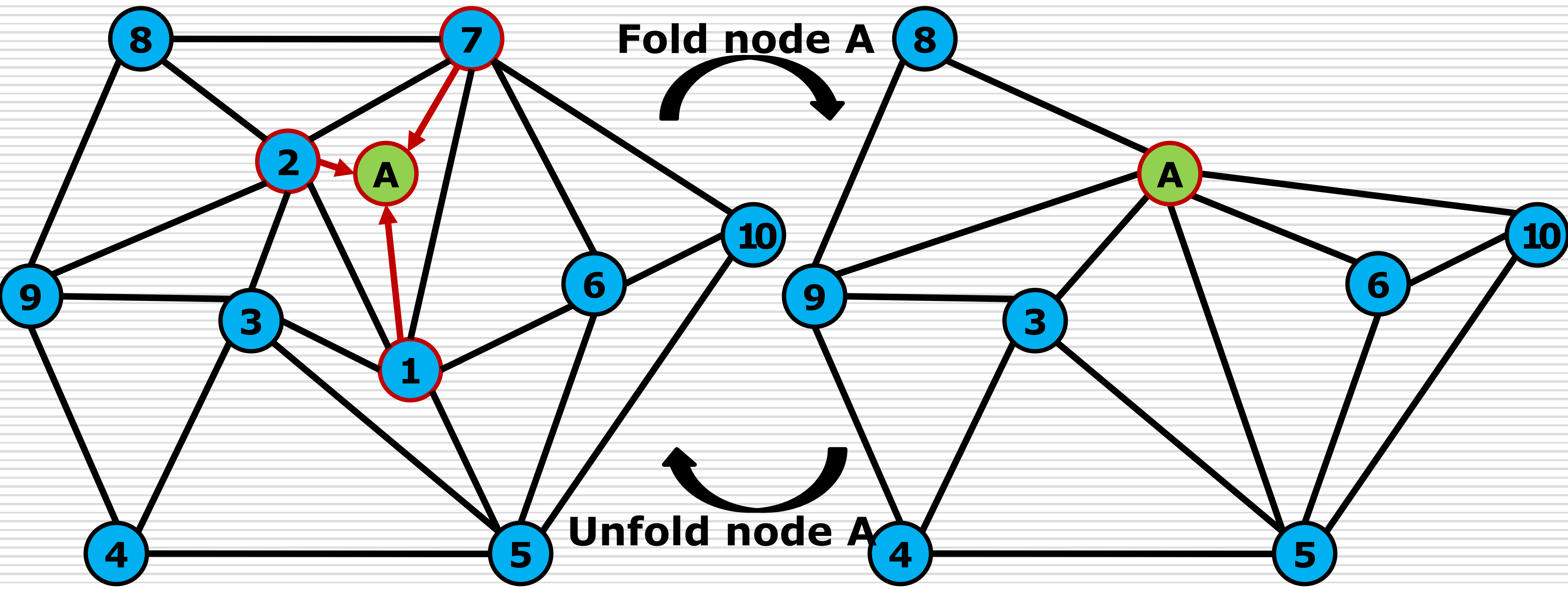
Vertex Tree Example



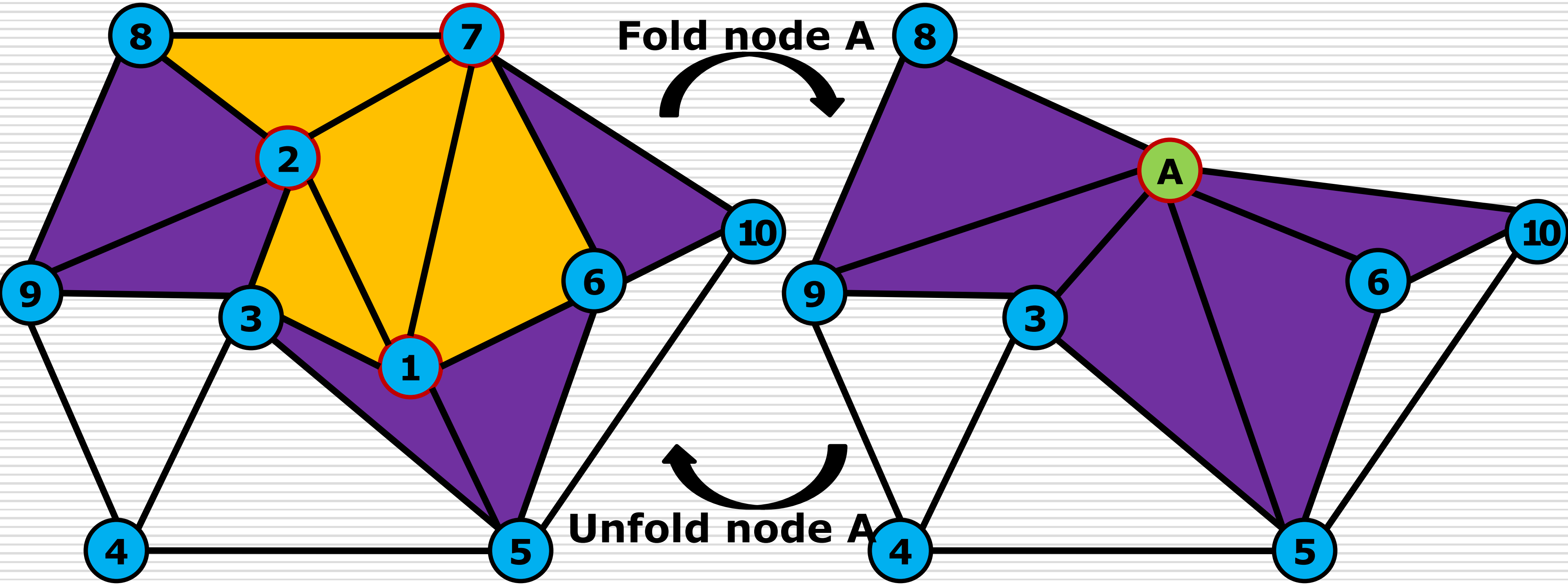
Triangles in Active List

Vertex Tree

The Vertex Tree: Folding & Unfolding



The Vertex Tree: Tris & Subtris



Tris: triangles that change shape upon folding

Subtris: triangles that disappear completely