Level of Detail (LOD) Models

Part Two

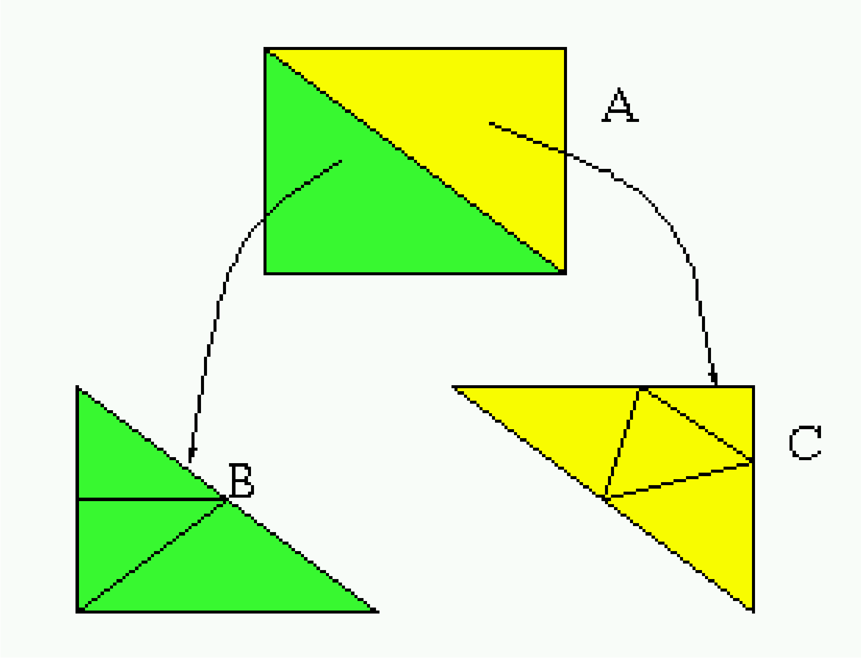
Classification

LOD MODELS (2)

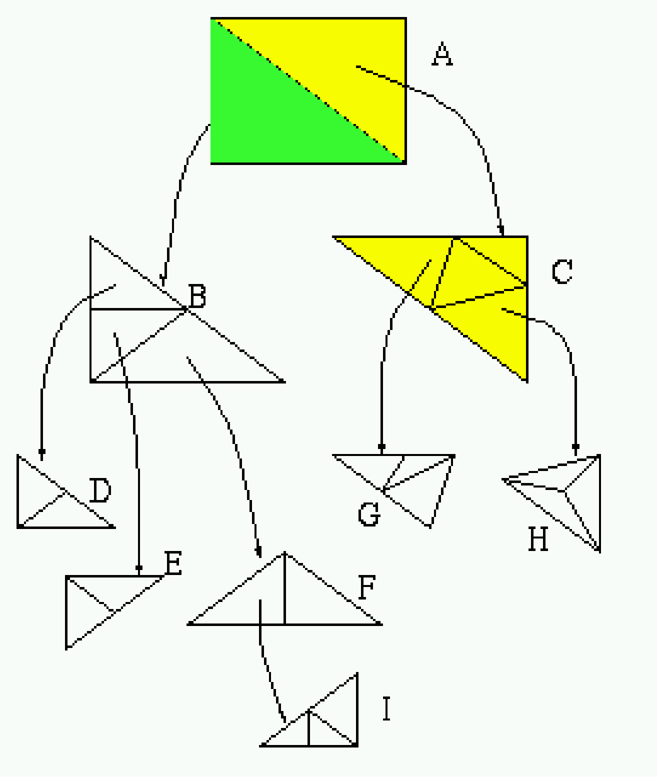
- Nested models
  - based on a nested subdivisions of the surface domain
  - each cell in the subdivision is refined independently
- Evolutionary models
  - based on the evolution of a mesh through local modifications
  - different meshes are obtained by combining different groups of modifications

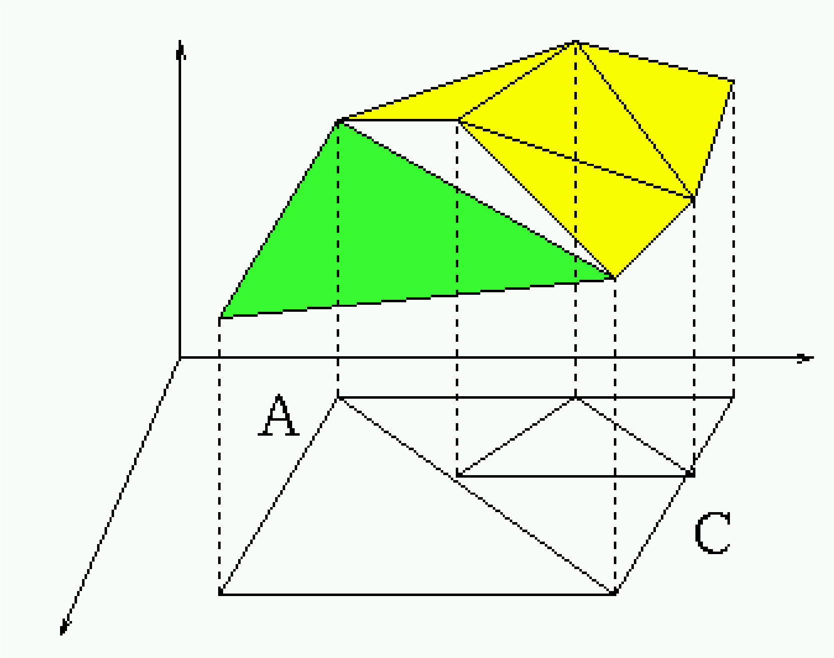
Nested models

- Root mesh at coarse resolution
- General refinement rule: each region in the mesh is refined independently into a local mesh
- An arc links a region to the local mesh refining it



- Cracks can appear in the surface if one node is refined independently from its neighbors
- Cracks are due to edges that split during refinement



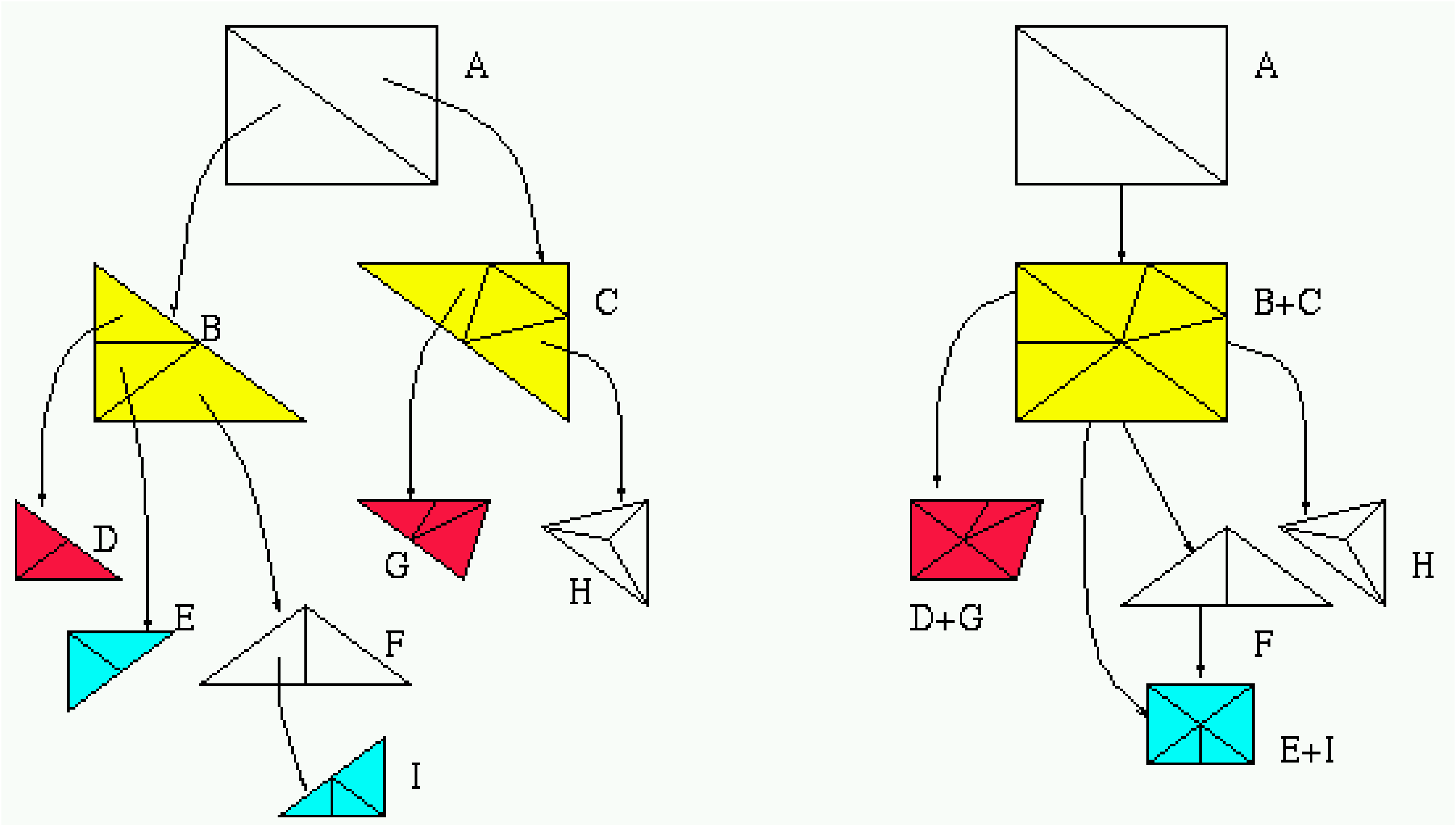


Nested models as MTs

- A node of the tree is not necessarily a node of the MT
- Nodes of the MT are obtained by node clustering
- *Clustering rule:* if an edge *e* of a triangle *t* splits during refinement, then
  - the same split must occur in refining triangle *t'* adjacent to *t* along *e*
  - the meshes refining *t* and *t'* must be clustered
- *Propagation:* many nodes of the tree can be clustered to form one node of the MT because of edge splits

...Nested models as MTs...

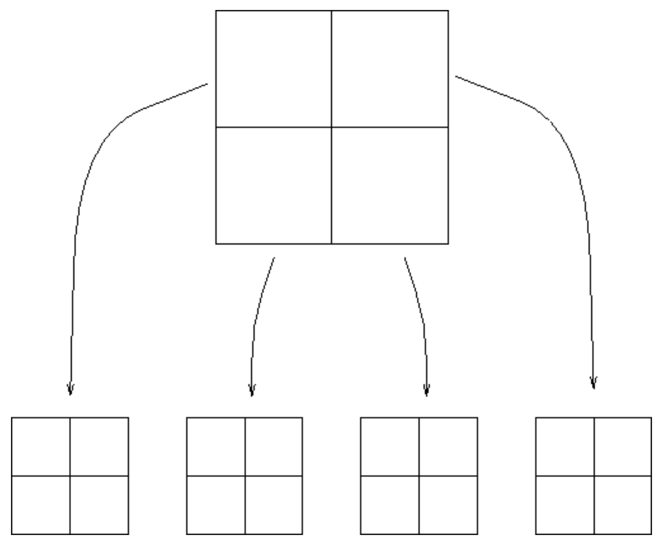
LOD MODELS (2)



Quadtree-like models

- Models based on regular nested subdivisions
- Suitable for explicit surfaces, data on a regular grid

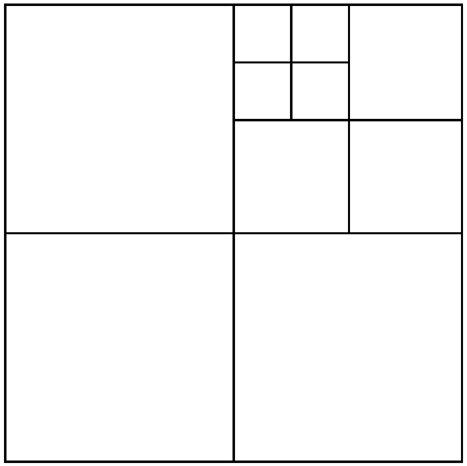
- **Quadtree:** recursive subdivision of a square universe into quadrants

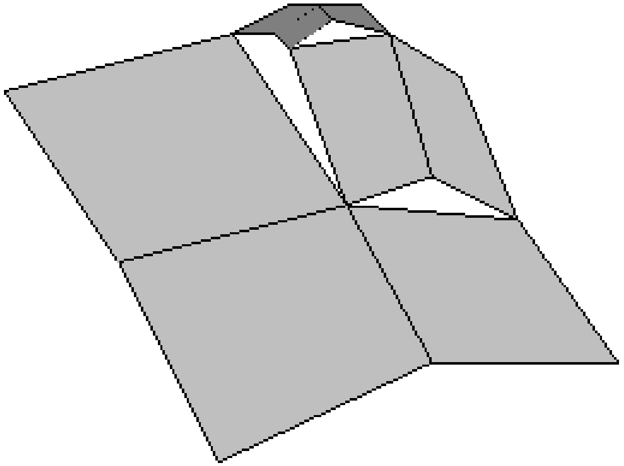


LOD MODELS (2)

Quadtree surface

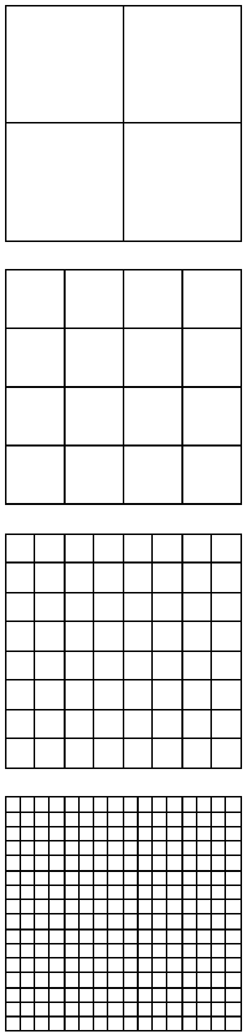
- Surface within each quadrant approximated through a bilinear patch
- Each patch interpolates data at its four vertices
- A mesh formed of quadrants from different levels has cracks





... Quadtree surface ...

- All quadrants of a given level must be clustered to form a single component
- Complete levels are the only possible conforming meshes
- It is not possible to change resolution through space



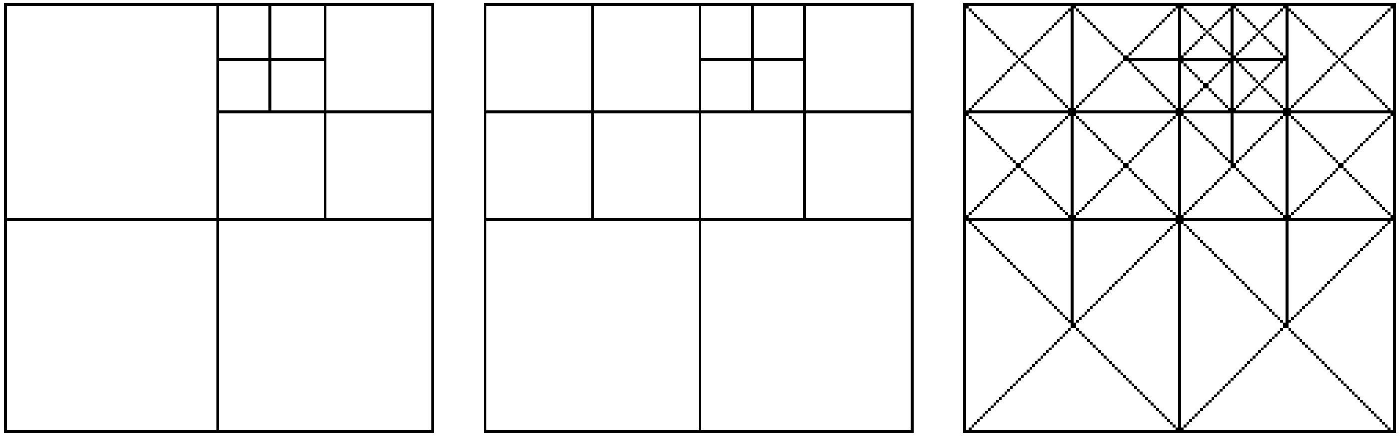
Octree [Wilhelms and Van Gelder, 1994]

- Extension of quadtree to volume data
  - Subdivision of a cubic universe into octants
  - Data field within each octant approximated through tri-linear patch
- Same problems as quadtree with cracks between octants of different levels

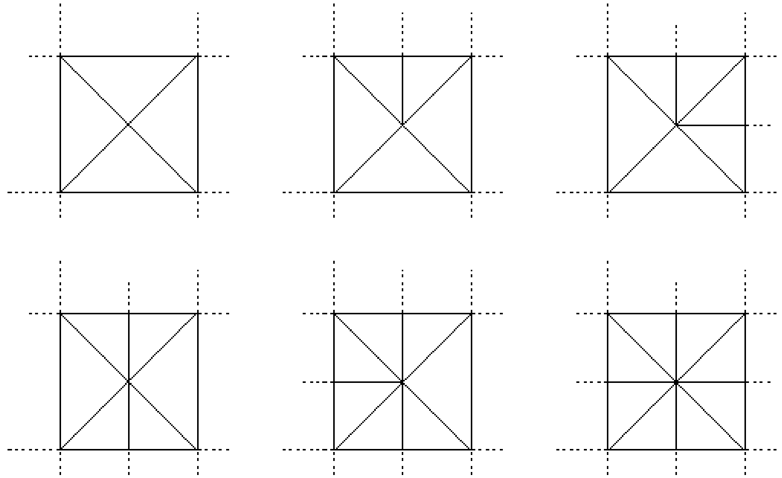
Restricted quadtree [Von Herzen & Barr, 1987]

Merging triangles from different levels without cracks:

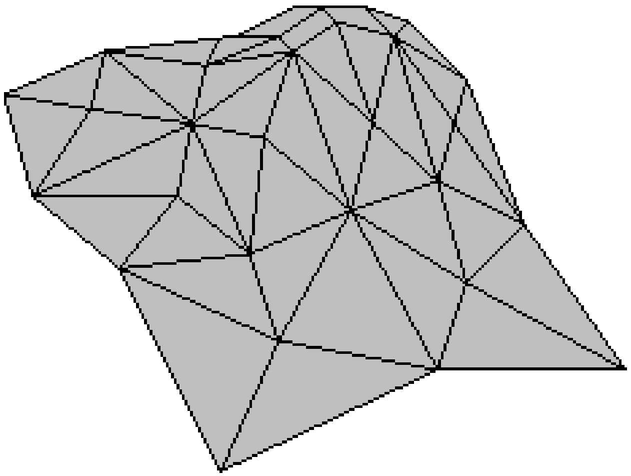
- Adjacent quadrants can differ by one level
- Each quadrant is triangulated
- Linear interpolation is used on each triangle



...Restricted quadtree...



- Each quadrant can be triangulated in 16 possible patterns
- Triangulation pattern depends on levels of adjacent quadrants



- Mesh is always conforming: no cracks
- The number of regions is higher than in the original quadtree

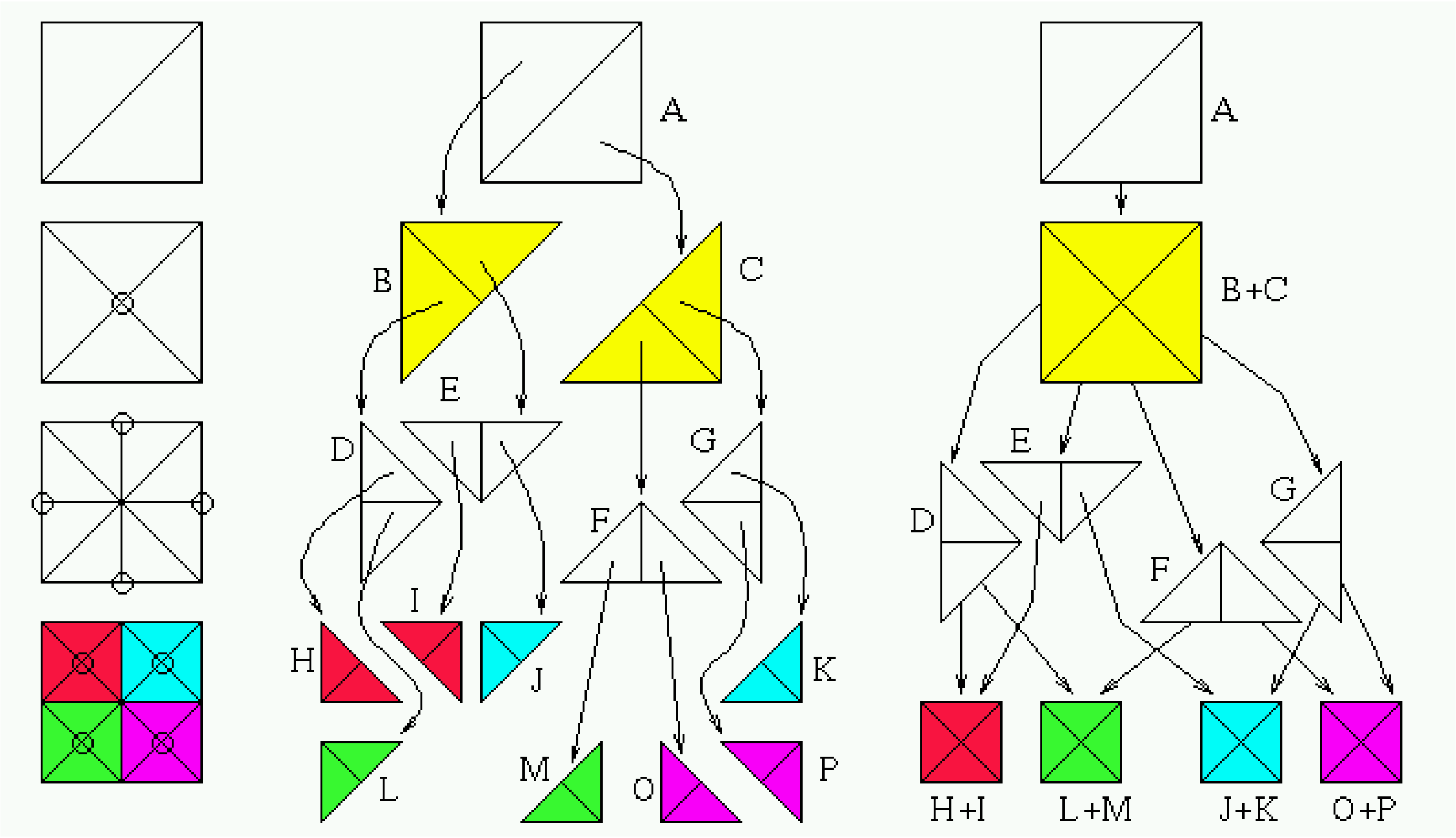
Restricted quadtree and wavelets [Gross et al., 1996]

- Quadtree subdivision naturally adapt to computation of wavelet coefficients at grid vertices
- LOD refers to detail relevance in wavelet space, rather than absolute approximation error on surface
- Selective refinement: vertices are selected according to their LOD, and the resulting quadtree is triangulated a posteriori
- Two quadtree levels are allowed between adjacent quadrants
- More complex lookup table is necessary to obtain all triangulation patterns

Hierarchy of right triangles
[Lindstrom et al., 1996, Evans et al., 1997, Duchaineau et al., 1997, Pajarola, 1998]

L O D M O D E L S (2)

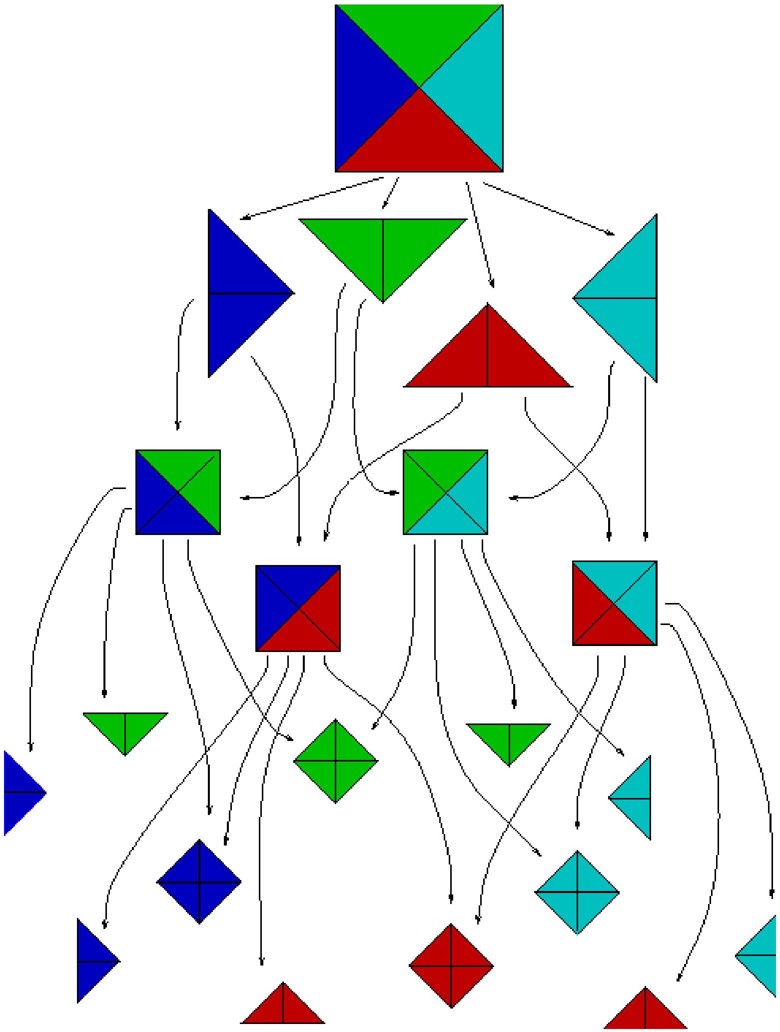
Each triangle is recursively bisected by splitting it along its longest edge



Binary tree representation MC representation

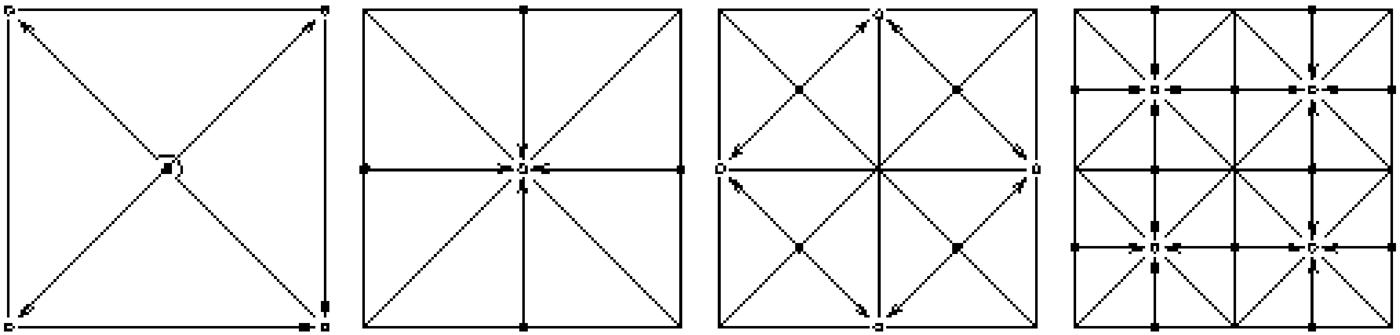
MT corresponding to a hierarchy of right triangles:

- cluster triangles of the same level that share a short edge
- each node is formed of four triangles (except at the boundary)
- two types of nodes:
  - squares
  - diamonds
- each node has two parents and four sons (except at the boundary, root, and leaves)

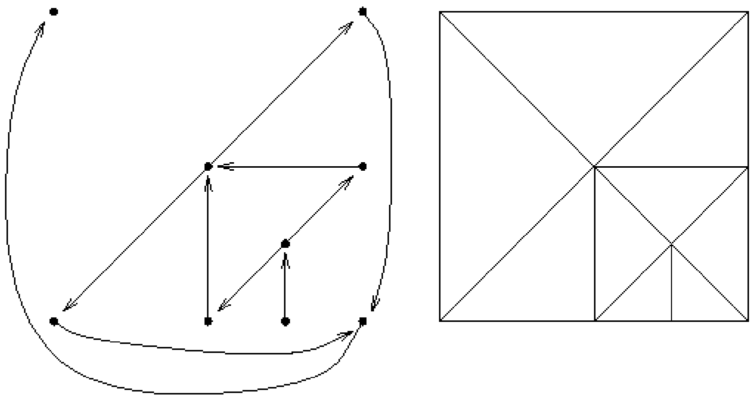


Corresponding hierarchy of vertices:

- each node in the MT is identified with central vertex of fragment
- each vertex depends on two other vertices



- selective refinement: mesh must contain all vertices needed to achieve the LOD, plus all their ancestors



...Hierarchy of right triangles...

- Different models are characterized by:
  - data structures
  - error evaluation
  - traversal algorithms

- Trade-off between space complexity and time efficiency

[Lindstrom et al., 1996]

- Data structure: matrix of input values
- Error: on-the-fly estimate of error in screen space
- Selective refinement: bottom-up vertex reduction

▲ no overhead for multiresolution data structure

▼ on-the-fly error estimate is expensive, speed-up techniques do not warrant exact error evaluation

...Hierarchy of right triangles...

[Duchaineau et al., 1997, Evans et al., 1997]

- Data structure: binary tree of triangles
- Error: a priori evaluation
- Selective refinement: top-down traversal of the tree, by forcing splits where necessary

▲ no need for numerical computation during refinement

▼ expensive data structure

An implicit data structure:

- Matrix of input values
- An array of errors, one for each triangle

- Each triangle, and each component can be identified by a unique code
- All topological and hierarchical relations involving vertices, triangles, and fragments can be evaluated by algebraic manipulation of codes
- The array of errors is addressed directly through codes

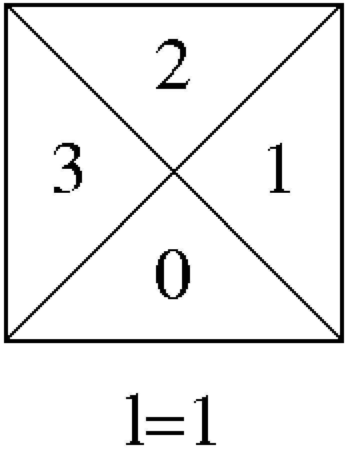
▲ Very efficient time performance with a moderate overhead

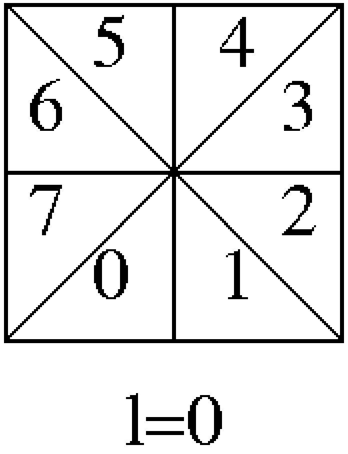
- Codes for triangles [Hebert, 1995]
  - Reference domain: unit square [0,1]x[0,1]
  - Quadtree subdivision:
    - a quadrant is identified by its center
    - the center of a quadrant at level *m* is encoded by a sequence of *m* quaternary digits (*2m* bits)

$$\sum_{i=1}^{m} 2^{-i} \sigma_i$$

where all σ_i are pairs of signs: (-1,-1) (-1,1) (1,-1) (1,1)

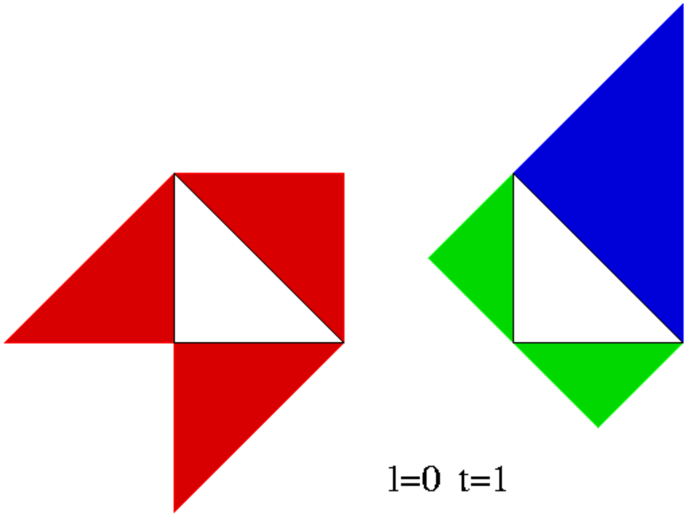
- For each level of the quadtree there are two levels in the tree of triangles: each quadrant is subdivided in two possible ways
- Triangles in a quadrant are identified by a pair of digits *(l,t)* where *l* is the type of subdivision, and *t* is the index of a triangle in the subdivision (total 4 bits)





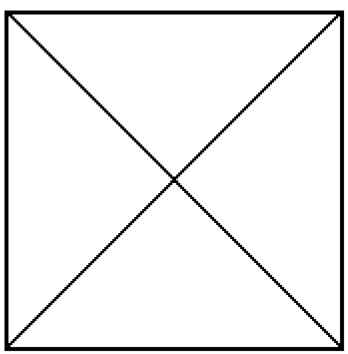
...Hierarchy of right triangles...

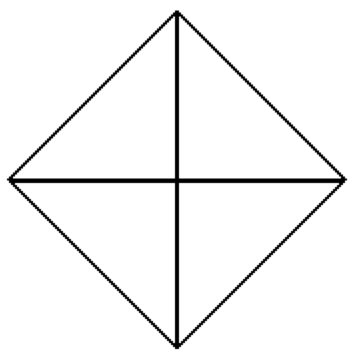
- Topological relations are evaluated by algebraic manipulation of codes. From a triangle we can obtain:
  - vertices
  - parent triangle
  - sons
  - adjacent triangles at the same level
  - adjacent triangle at the previous level
  - adjacent triangles at the next level



...Hierarchy of right triangles...

- Codes for nodes of the MT:
  - two kinds of nodes: squares and diamonds
  - a node is encoded by its center
  - a square coincides with a quadtree quadrant ➜ same code
  - a diamond is encoded with a similar formula that identifies its center vertex



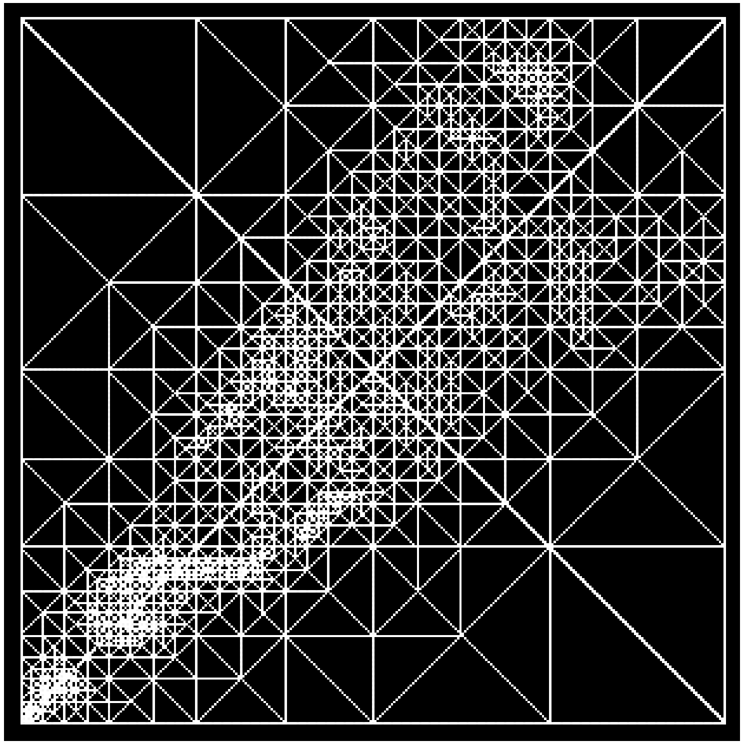


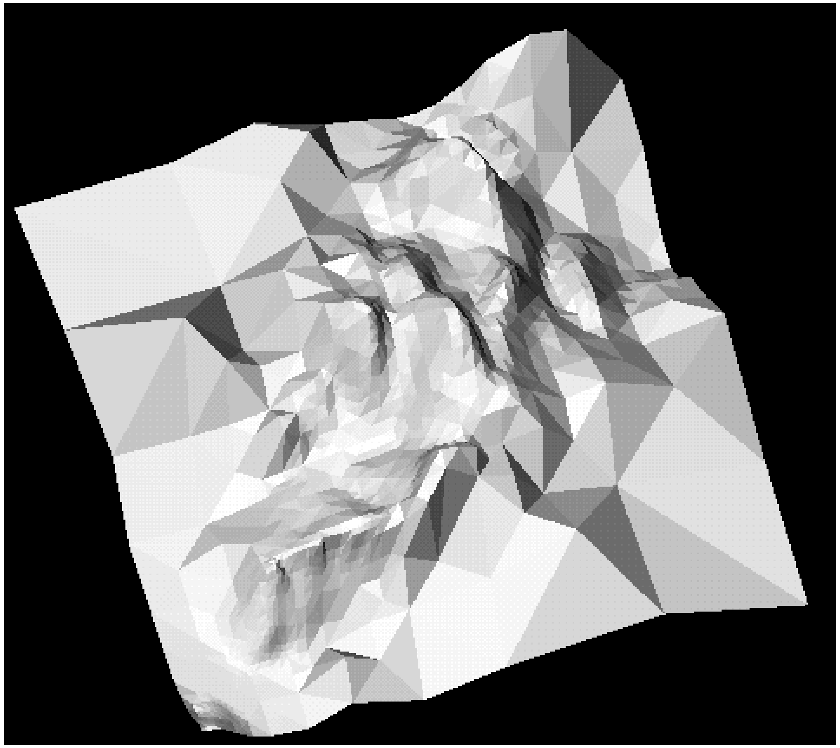
...Hierarchy of right triangles...

- Relations among triangles and nodes are evaluated by algebraic manipulation of codes. From a node we can obtain:
  - triangles in it
  - triangles in its floor
  - parent nodes
  - child nodes

The algorithm for selective refinement for MT can be implemented efficiently on a hierarchy of right triangles encoded by the implicit data structure

Results of selective refinement with view frustum, and error increasing with distance from viewpoint



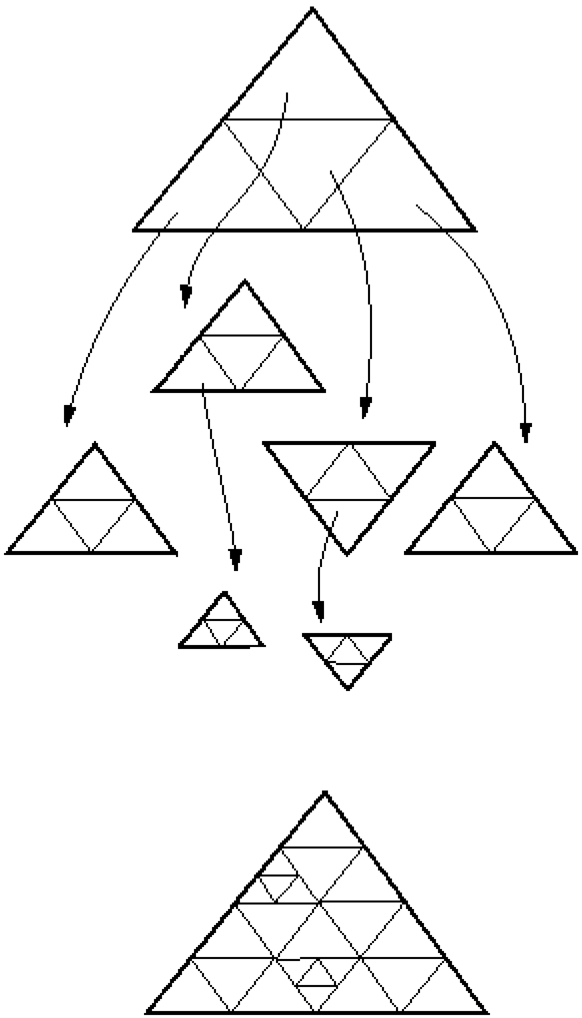


Multi-tetra framework [Maubach, 1994, Zhou et al., 1997]

- Extension of hierarchy of right triangles to volume data:
  - Subdivision of a cubic universe into twelve tetrahedra
  - Recursive bisection of each tetrahedron at the midpoint of its longest edge
- Implicit data structure as in the 2D case [Hebert, 1994]

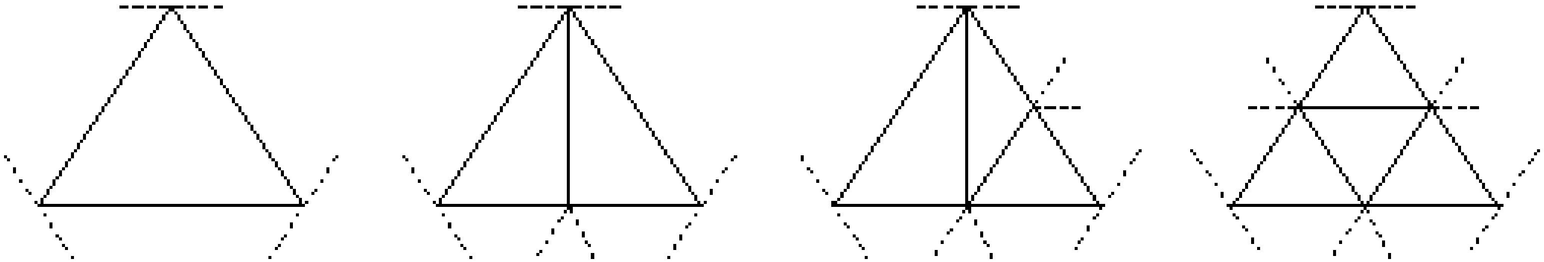
Quaternary triangulations

- Recursive subdivision of a triangular domain into four triangles by joining the edge midpoints
- Applicable as a refinement scheme to an arbitrary surface mesh at low resolution
- Topological constraints on positions of vertices (needs re-meshing)
- Supports methods based on wavelets
- A mesh made of triangles from different levels has cracks



...Quaternary triangulations...

- Different levels of the hierarchy can be merged through a posteriori triangulation of non-conforming meshes
- Eight triangulation patterns
- Patterns less regular than those used for restricted quadtrees
- Model completed with such triangulation patterns is **not** an MT



Extension to 3D [Grosso & Ertl, 1997, Grumpf, 1997]

- Recursive partition of a tetrahedron into eight tetrahedra by splitting each edge at its midpoint
- 8-ary tree
- Tetrahedra from different levels made conforming through subdivision patterns analogous to those in 2D

Adaptive hierarchical triangulations

- Based on irregular triangulations
- Suitable for sparse data sets
- Error driven subdivision rule: refine a triangle by inserting vertices that cause the largest errors
- Vertices can be inserted inside a triangle and/or on its edges

▲ More adaptive than models based on fixed subdivision rules

▼ May contain elongated triangles (slivers)

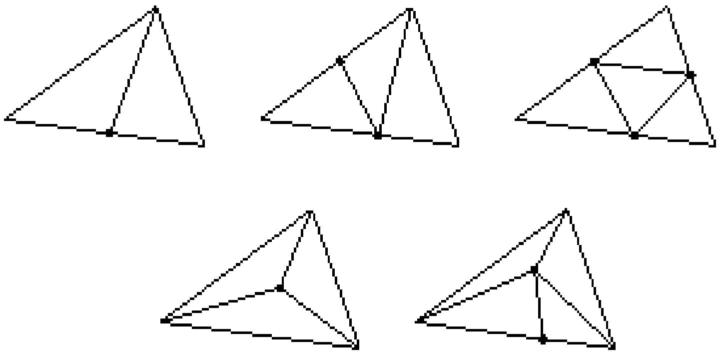
...Adaptive hierarchical triangulations...

L O D   M O D E L S

(2)

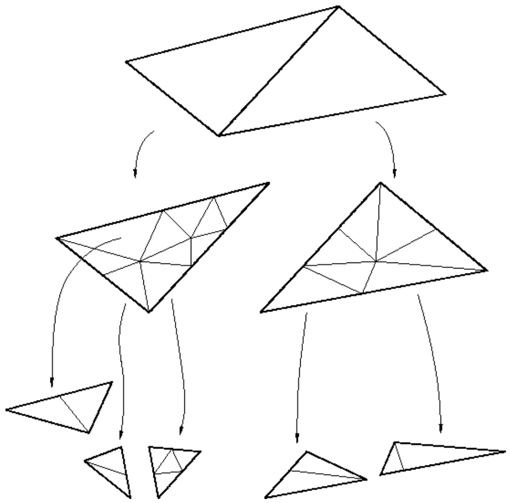
[Pavlidis and Scarlatos, 1990/92]

- Find the vertex causing the largest error inside the triangle and the vertices causing the largest error along each edge
- Select only vertices whose error is beyond a given threshold
- Use predefined subdivision patterns



[De Floriani and Puppo, 1992/95]

- Insert the vertex causing the largest error until a given accuracy is achieved
- At each insertion compute the Delaunay triangulation



Evaluation of tree-like models

Regular subdivisions

- Pros:
  - easy to handle
  - compact data structures
  - regular shape of regions
  - support wavelets
- Cons:
  - only regular data: topological constraints
  - quadtrees and right triangles only for terrain
  - less adaptive than irregular triangulations

Irregular subdivisions

- Pros:
  - suitable for arbitrary data and for arbitrary surfaces
  - more adaptive than regular subdivisions
- Cons:
  - elongated triangles
  - cumbersome data structures
  - selective refinement not easy

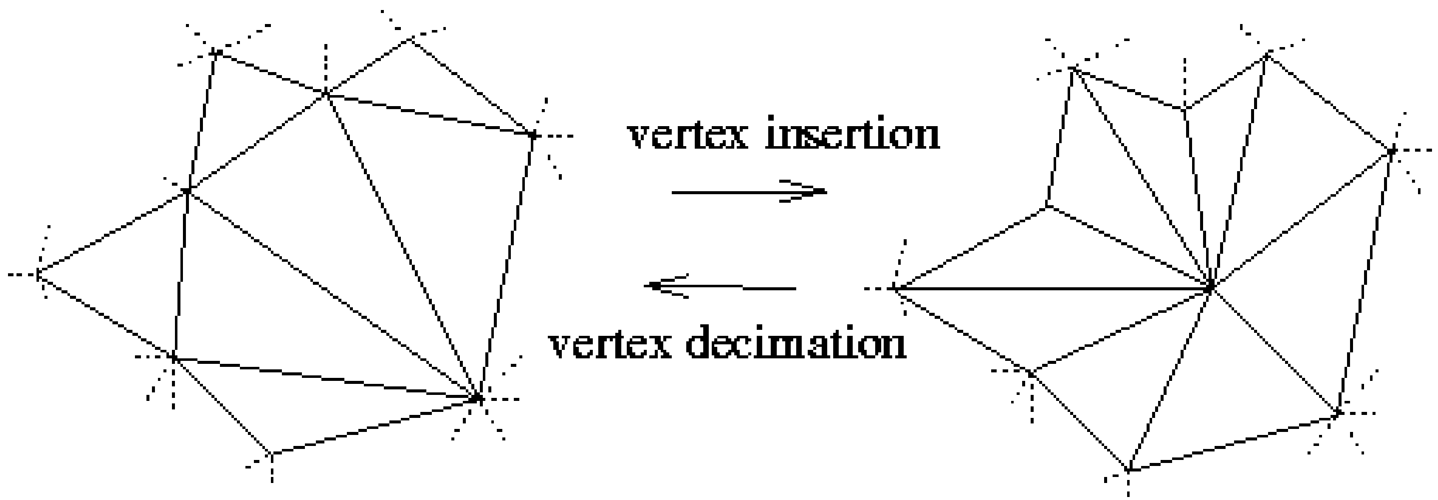
Evolutionary models

Store the evolution of a mesh through either refinement or simplification algorithm based on *local modifications*

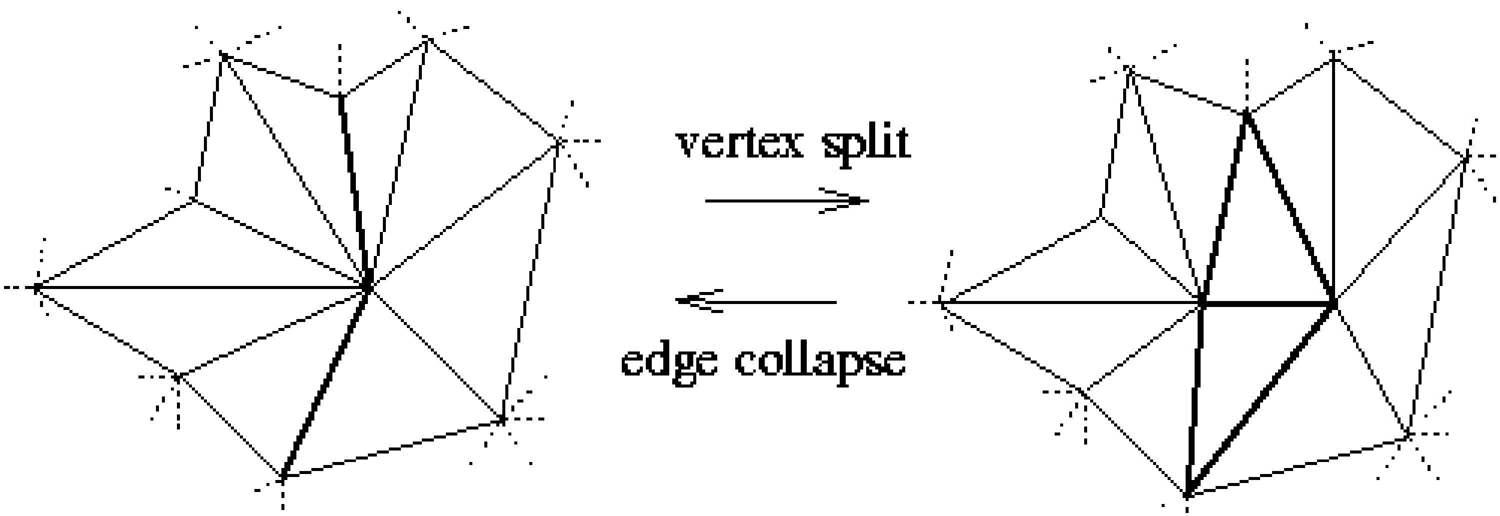
L O D M O D E L S (2)

- Partial order is given by relations among components of the mesh (vertices, faces, etc…) before and after each local modification
- Different models characterized by:
  - types of surfaces supported
  - construction method
  - information stored (geometry, connectivity, topology, interference, attributes, error, etc…)
  - operations supported - efficiency of algorithms

- Models based on vertex insertion / vertex decimation
  - [De Floriani, 1989]
  - [de Berg and Dobrindt, 1995]
  - [Cignoni et al., 1995/97]
  - [Brown, 1996/97]
  - [Klein and Strasser, 1996]
  - [De Floriani et al., 1996/97/98]



- Models based on vertex split / edge collapse
  - [Hoppe, 1996/97/98]
  - [Xia et al., 1996/97]
  - [Maheswari et al., 1997]
  - [Gueziec et al., 1998]
  - [Kobbelt et al., 1998]



Construction through refinement

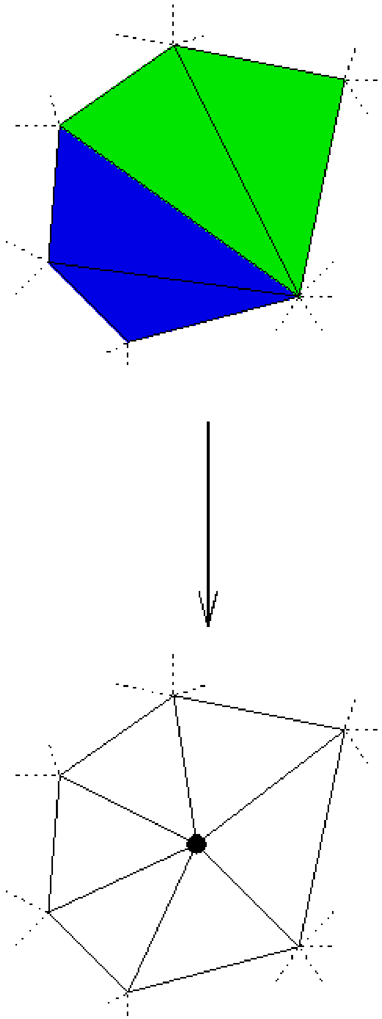
L O D M O D E L S (2)

- Method applied only to build models based on vertex insertion
  - Start from a coarse mesh at low resolution built on a small subset of data
  - Perform iterative local refinements until all data have been inserted as vertices of the mesh
- The initial mesh is the root of an MT
- Each local refinement generates a node of an MT formed of new triangles inserted in the mesh
- Difficult to apply to generic manifold surfaces

...Construction through refinement...

- Node generated by vertex insertion: a star of triangles surrounding the new vertex
- Floor of a node: a star-shaped triangulated polygon

- Key issues:
  - selection of vertices to insert
  - triangulation method
  - degree of vertices (size of fragments)
  - error estimation
  - height of the resulting hierarchy



...Construction through refinement...

- Greedy refinement:
  - at each step, insert vertex causing the largest error
  - mesh update based on either Delaunay or data dependent triangulation

▲ good heuristic to reduce the number of points to achieve a given accuracy

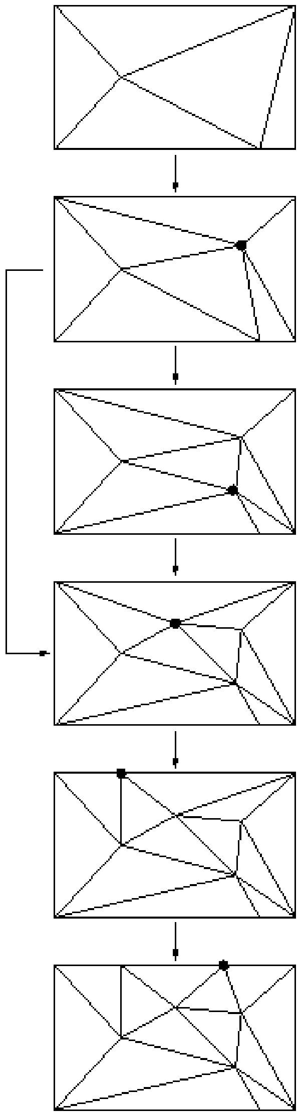
▲ inserting vertices of bounded degree guarantees linear growth

▼ method cannot guarantee that accuracy improves at every refinement step

▼ fragments may pile-up in a high hierarchy:

  low expressive power

  low performance of traversal algorithms



...Construction through refinement...

Extension to 3D [Cignoni et al., 1994/1997]

- Iterative insertion of vertices in a Delaunay tetrahedrization
- Vertex selection rule as in 2D
- Applicable to convex and curvilinear volume data sets

Construction through simplification

L
O
D

M
O
D
E
L
S

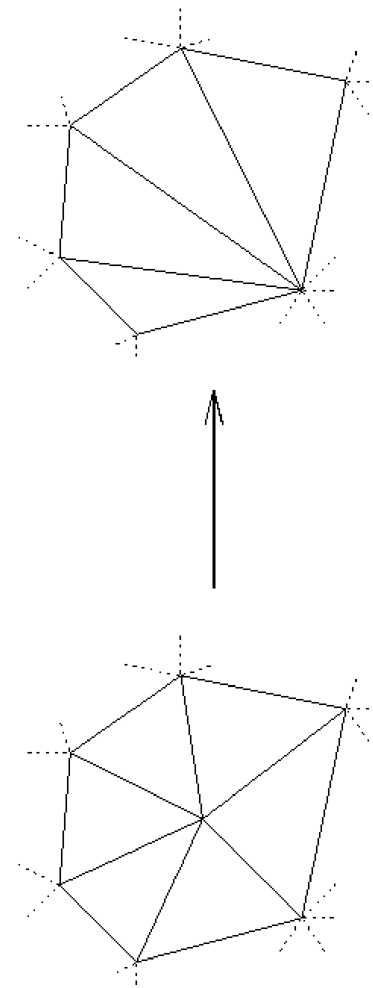
(2)

- Any local simplification rule can be used (vertex decimation, edge collapse, etc.)
  - Start from mesh at full resolution, based on all data
  - Perform iterative local simplifications
- The final mesh is the root of an MT
- Each simplification step generates a node formed of triangles eliminated from the mesh
- The new portion of mesh generated by a simplification step is the floor of the corresponding node
- Applicable to generic manifold surfaces

...Construction through simplification...

- Vertex decimation:
  - node: a star of triangles surrounding the removed vertex
  - floor of a node: a star-shaped triangulated polygon

- Key issues:
  - selection of vertices to remove
  - triangulation method
  - degree of vertices (size of fragments)
  - error estimation
  - height of the resulting hierarchy

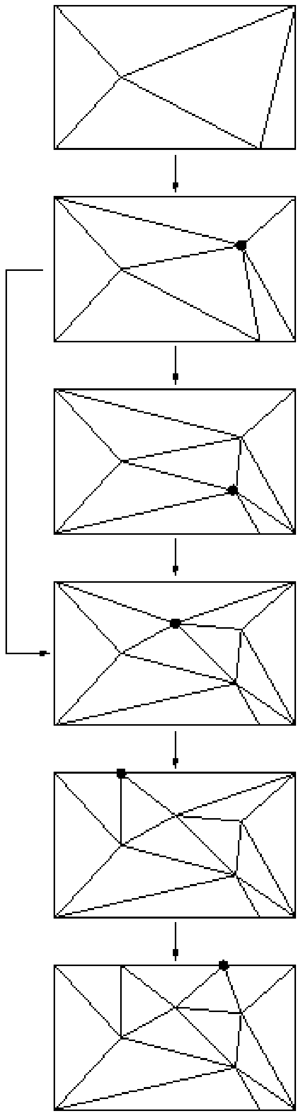


LOD MODELS (2)

...Construction through simplification...

- Greedy decimation:
  - at each step, remove vertex causing the least error increase
  - mesh update based on either Delaunay triangulation or heuristics

- result similar to greedy refinement
- removing vertices of bounded degree guarantees linear growth
- components may pile-up in an unbalanced DAG



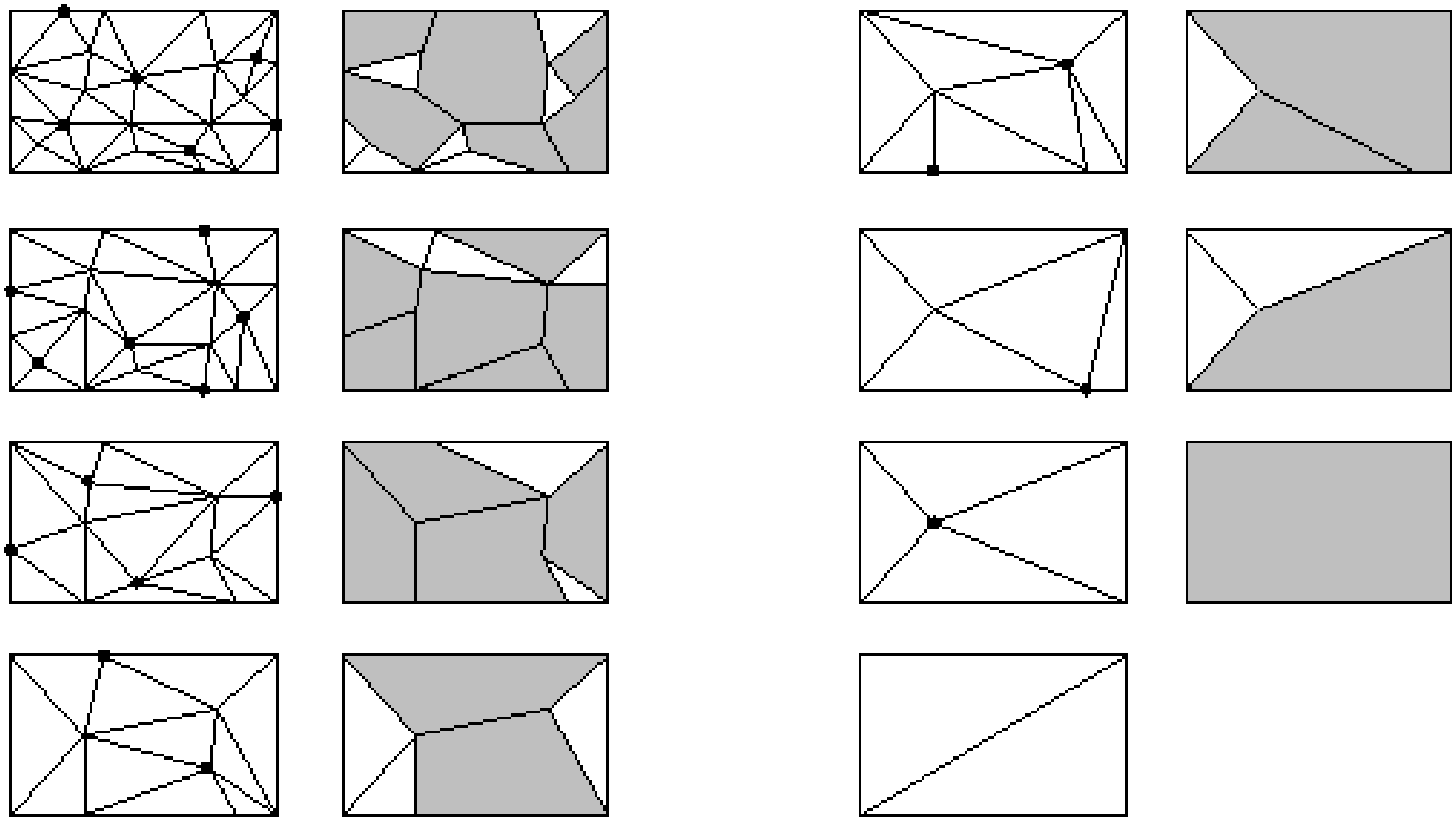
...Construction through simplification...

- Decimation of independent sets of vertices:
  - at each iteration remove a set vertices such that all vertices
    - have **bounded degree**
    - are **mutually independent** (no two vertices share an edge)
  - vertices to remove selected with a greedy technique giving highest priority to vertices causing the least error increase
  - each vertex defines a different fragment
  - holes triangulated with Delaunay or data-dependent rule

- method guarantees **linear growth, bounded width** and **logarithmic height**

...Construction through simplification...

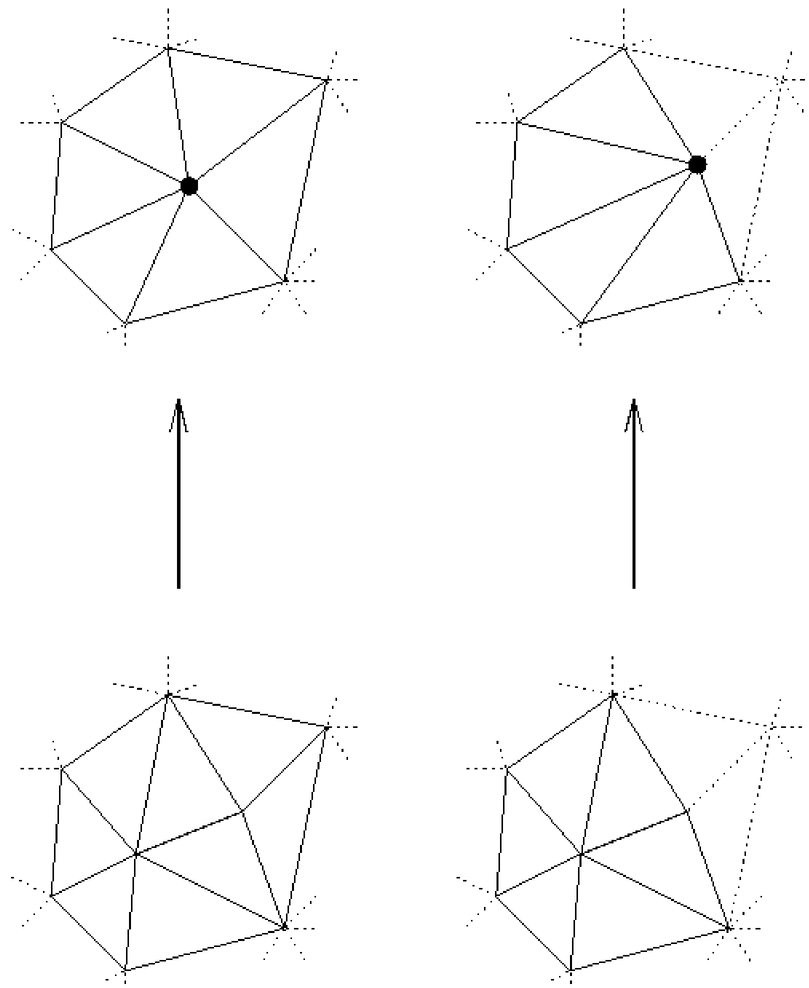
LOD MODELS (2)



...Construction through simplification...

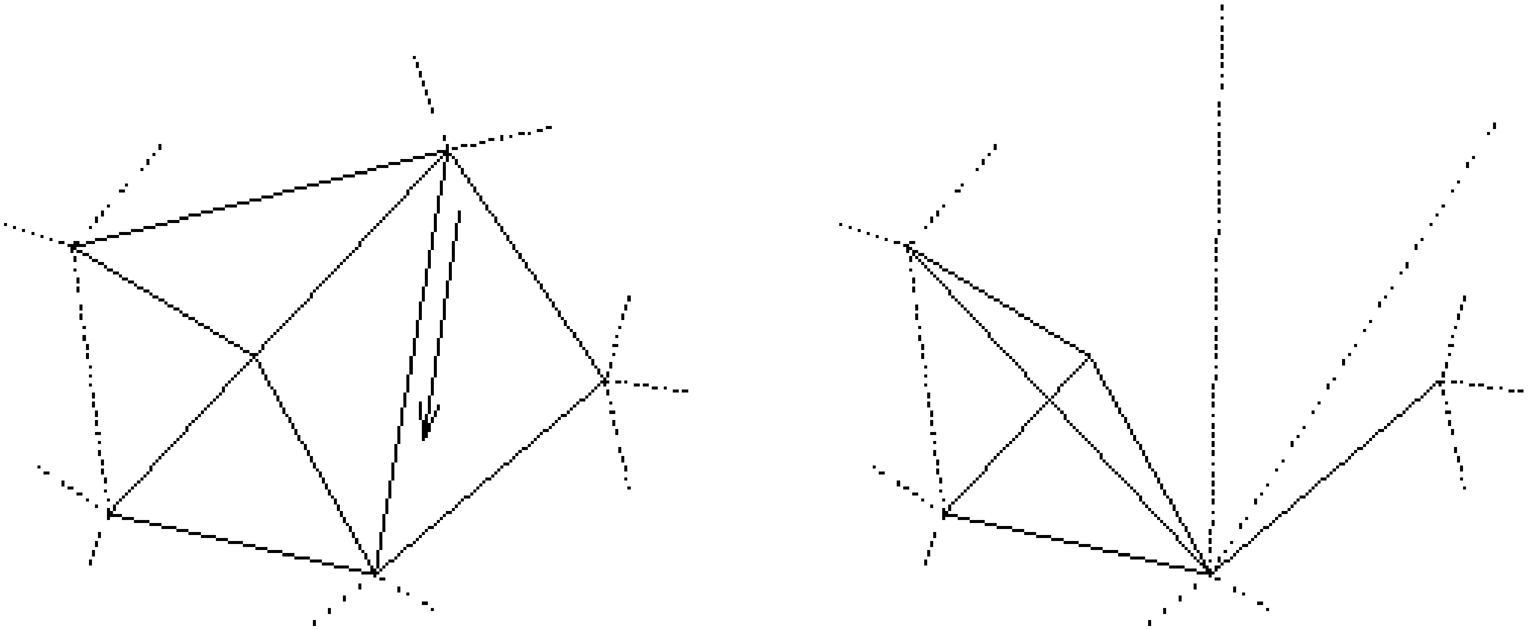
- Edge collapse on midpoint:
  - node: cycle of triangles surrounding the collapsed edge
  - floor: star of triangles surrounding the vertex resulting from collapse

- Edge collapse on endpoint:
  - node: star of triangles surrounding the endpoint
  - floor: fan of triangles centered at endpoint
  - equivalent to decimation with special update rule



...Construction through simplification...

Illegal edge collapse: one or more triangles *flip over* because of collapse operation.



...Construction through simplification...

- Edge collapsing rules:
  - Greedy: collapse an edge at each step:
    - the shortest edge
    - the edge causing the least error increase
    - an edge surrounded by almost coplanar faces
  - Independent set:
    - two edges are independent if they have disjoint influence regions
    - select a maximal set of independent edges and collapse them all together
- Results similar to decimation:
  - ▲ removing vertices of bounded degree guarantees bounded width and linear growth
  - ▲ removing an independent set guarantees logarithmic height
  - ▼ in greedy collapse fragments may pile-up in a high hierarchy

...Construction through simplification...

Extension to 3D [Cignoni et al., 1997]

- Edge collapse in a tetrahedrization: collapse an edge and the star of tetrahedra surrounding it
- Edge selection as in 2D
- Applicable to all kinds of volume data sets

...Construction through simplification...

Decimation

- ▲ smaller influence regions
- ▲ more regular triangles
- ▲ always possible for any vertex

- ▼ more complex update rules
- ▼ geo-morphing difficult

Collapse

- ▲ simple update rules
- ▲ supports geo-morphing

- ▼ larger influence regions
- ▼ slivers
- ▼ collapse not always legal

Data structures

Relevant information on evolutionary models

- **Geometry:** coordinates of vertices
- **Connectivity:** triples of vertices forming triangles
- **Topology:** adjacency, boundary, co-boundary relations
  - local topology: among elements of a single node
  - global topology: among components of different nodes
- **Spatial interference:** relations among nodes and triangles that have spatial interference
- **Additional information:** accuracy, material, surface normal, etc.

...Data structures...

- Different data structures characterized by the amount of information stored
- Trade-off between spatial complexity and efficiency
  - compact data structures more suitable to storage and transmission
  - extended data structures more suitable to complex operations:
    - selective refinement
    - spatial queries
- Compactness can be achieved by exploiting properties of special models

Linear sequences: store sequences of local modifications that produce a refined mesh starting at a coarse mesh

- List of vertices [Klein & Strasser, 1996] :
  - store initial mesh plus sequence of vertices in suitable order
  - each vertex in the sequence is inserted iteratively to refine mesh
  - mesh is updated with Delaunay (implicit) rule at each insertion

- ▲ very compact
- ▼ suitable only to explicit and parametric surfaces
- ▼ local update requires numerical computation
- ▼ selective refinement is computationally expensive
- ▼ no connectivity, topological, and interference information maintained

- List of triangles [Cignoni et al., 1995] :
  - store all triangles appearing during refinement/simplification
  - each triangle is tagged with a *life:* range of accuracies through which it "survives" during refinement/simplification
  - life is used to extract meshes at a given (uniform) LOD
  - extended to 3D for volume data [Cignoni et al., 1997]

▲ applicable to all kinds of surface
▲ extraction of a uniform LOD very efficient
▲ moderately compact
▼ selective refinement not possible
▼ no topology and interference maintained

- Progressive Meshes [Hoppe, 1996/98] :
  - store initial coarse mesh plus a sequence of vertex splits (inverse operation of edge collapse) in suitable order
  - each vertex split is maintained in compressed format
  - each vertex split gives a node of a corresponding MT
  - a uniform LOD is extracted by expanding the sequence up to the desired level

  - ▲ compact
  - ▲ extraction of a uniform LOD efficient without numerical computation
  - ▲ suitable additional structures to maintain attributes
  - ▼ selective refinement needs additional information
  - ▼ no connectivity, topology and interference maintained

...Data structures...

- Sequence of vertex insertions [De Floriani et al., 1998]:
  - store initial mesh plus a set to vertex insertions in suitable order
  - analogous to PM but based on vertex insertion rather than split
  - supports arbitrary triangulations including:
    - Delaunay triangulation
    - Constrained Delaunay triangulation
    - Data dependent triangulations

▲ more flexible than PM

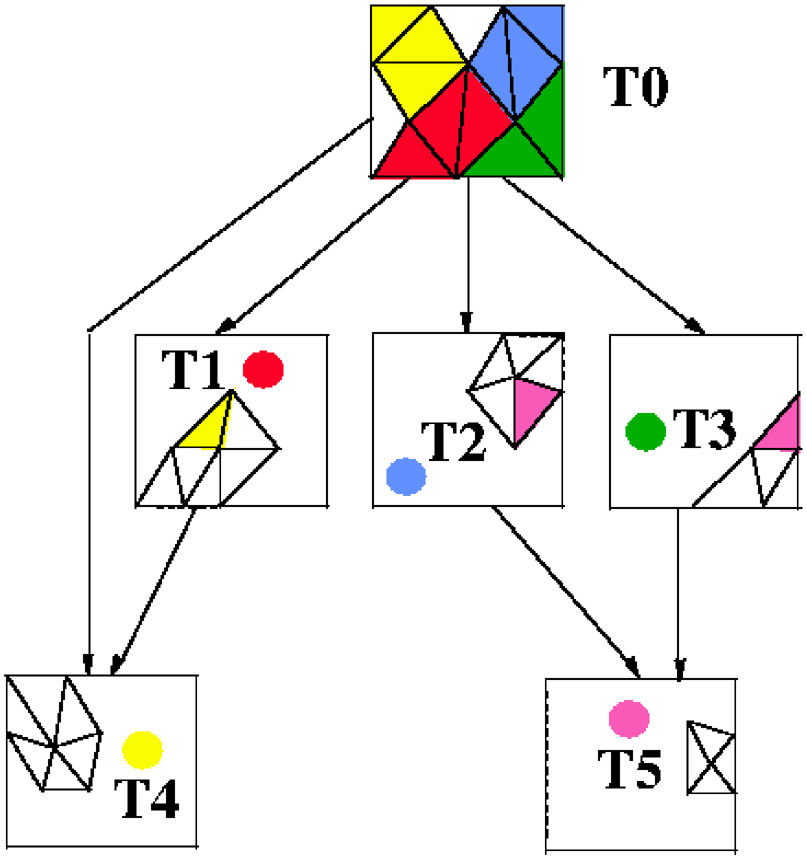
▼ slightly more expensive than PM

...Data Structures...

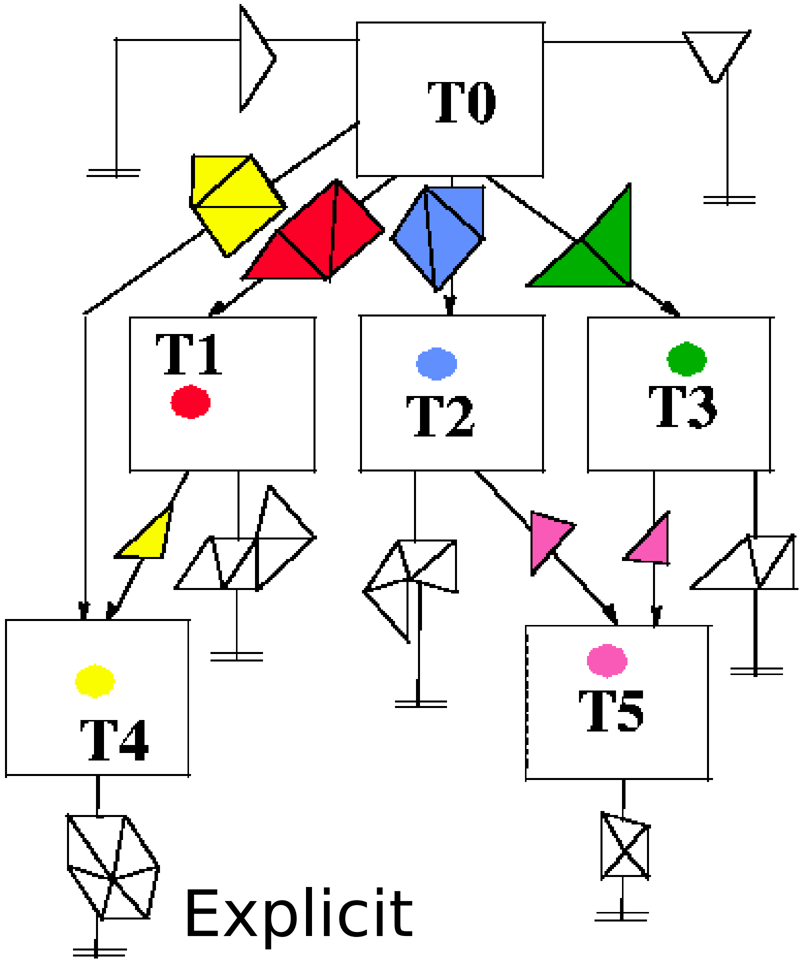
Explicit MT representation [De Floriani et al., 1996/98]

- Geometry: vertex coordinates
- Connectivity:
  - for each triangle: error + references to its three vertices
- DAG structure:
  - for every arc ***(T_j,T_i)***: links to source and destination node, link to the set of triangles of ***T_j*** which form the floor of ***T_i***
  - for every node: link to the sets of its incoming and outcoming arcs

LOD MODELS (2)



MT



Explicit representation

...Explicit MT representation...

- ▲ Supports selective refinement efficiently
- ▲ Supports spatial queries efficiently

- ▼ High storage cost
- ▼ No topology

Compressed hierarchies:

- Key ideas:
  - each node of an MT is a local modification that can be encoded in compressed form
  - hierarchical links among nodes are encoded explicitly
- Different structures for models based on edge collapse (PM):
  - [Xia et al., 1996/97]
  - [Hoppe, 1997]
  - [Gueziec et al., 1998]
- One structure for models based on vertex decimation:
  - [De Floriani et al., 1997/98]

Compressed hierarchies for PMs

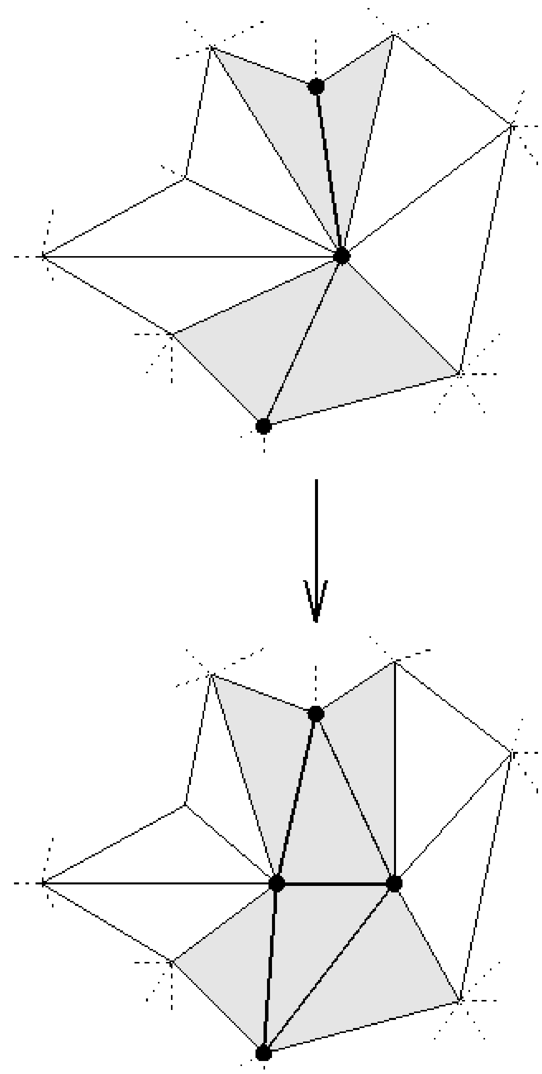
Vertex tree (forest) [Xia et al., 1996/97]

- Binary forest of vertices
  - topmost level: vertices of the base mesh
  - children of a vertex: vertices resulting from split
- Vertex split can be encoded in compressed form
- Rule for selective refinement: *a vertex can split if and only if all boundary vertices of the corresponding fragment belong to the current mesh* ⇒ need for additional interference links
- For each vertex:
  - parent-child relation in forest
  - additional links to vertices that must exist in order to allow split
- ▲ More compact than explicit MT
- ▼ Less general than explicit MT
- ▼ No control on accuracy of triangles

...Data structures...

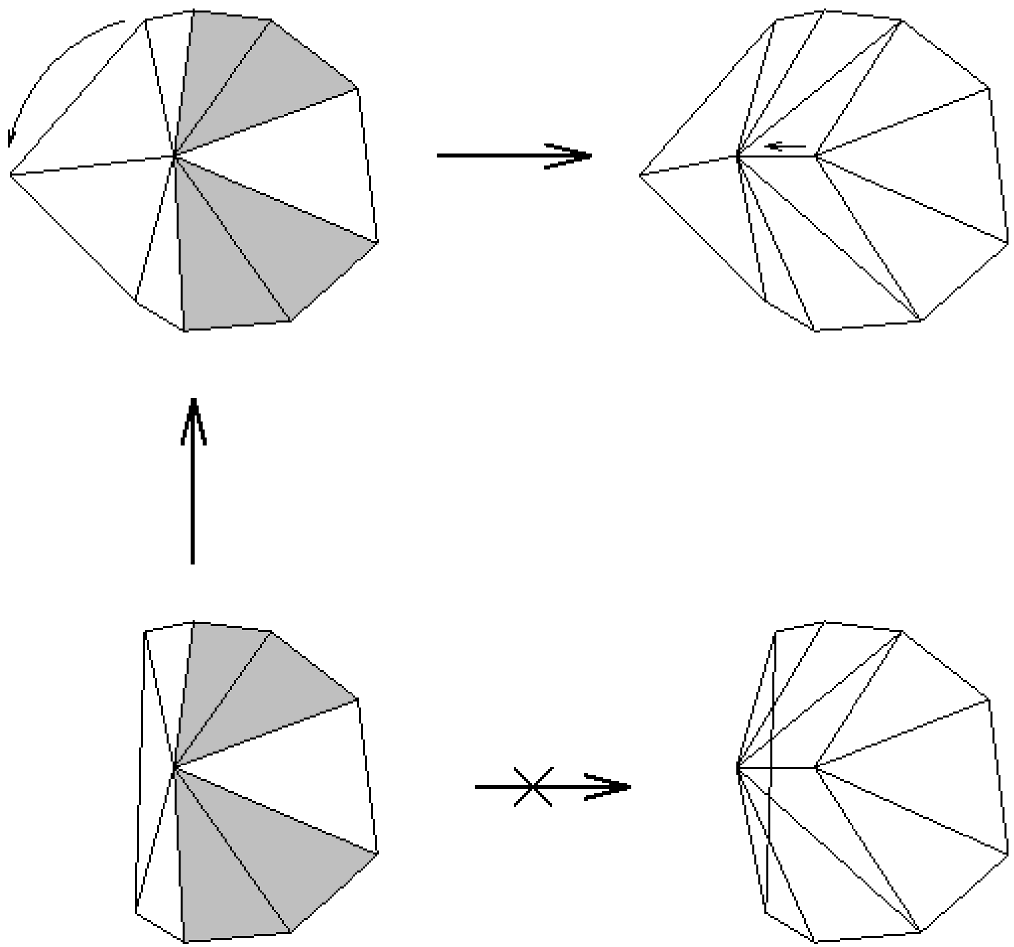
Extended PM structure [Hoppe, 1997]

- Same hierarchy as in [Xia et al.], but vertices are renamed at each split
- Array of vertices - for each vertex:
  - compressed split information
  - parent-child information
  - constant number of dependencies for split/collapse with triangles and vertices of influence region
- Array of faces - for each face:
  - indexes of vertices
  - indexes of (local) neighbors
- ▲ No variable-length fields
- ▲ Local topology
- ▼ High storage cost
- ▼ Illegal split may occur



L O D M O D E L S (2)

Example of illegal split



...Data structures...

Collapse DAG [Gueziec et al., 1998]

- DAG of edge collapse operations: one node per collapse
- Edge collapse always on endpoint one vertex survives
- Almost the same hierarchy as MT
- Dependencies maintained in hash tables

- Cost and performances similar to [Xia et al.]

Compressed hierarchies for MT based on decimation

Implicit MT [De Floriani et al., 1997/98]

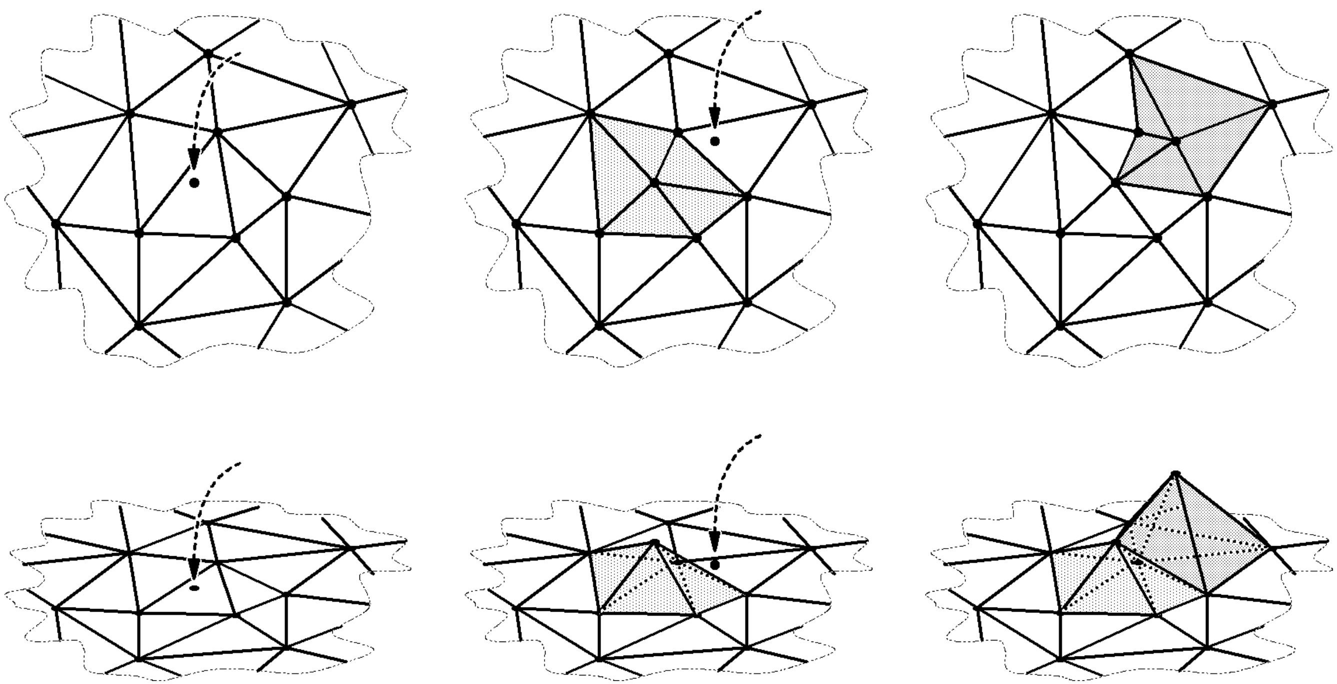
- Each node corresponds to vertex insertion/decimation
- Partial order of nodes is maintained
- Array of vertices, each entry storing vertex coordinates
- Array of arcs - for every arc *a:*
  - indexes of source and destination nodes
  - index of next arc with same destination node
- Array of nodes - for every node *n:*
  - index of first outgoing and incoming arcs
  - number of outgoing arcs
  - maximum error of its triangles
  - compressed information to perform vertex insertion/decimation

...Implicit MT...

- Storage cost:
  - ▲ 3.5 times cheaper than Explicit MT
  - ◆ Comparable with vertex tree, and collapse DAG
- Selective refinement:
  - ▼ Slower than Explicit MT
  - ▼ Accuracy not stored for each triangle resulting mesh not minimal (about twice the size of mesh obtained from Explicit MT)
  - ◆ Comparable with vertex tree, and collapse DAG

Hypertriangulation [Cignoni et al., 1995/97]

- Interpretation of an MT in a higher dimensional space: triangles of a new node are lifted along a "resolution axis" and welded on floor at the node boundary



L O D M O D E L S (2)

...Hypertriangulation...

- Data structure based on topological information (global adjacency)

  - ▲ topology is maintained explicitly
  - ▲ efficient support of queries requiring navigation of domain
  - ▲ interactive mesh editing
  - ▼ high storage cost
  - ▼ no interference information
  - ▼ selective refinement super-linear and possible only for some LOD functions

Selective refinement

- **Top-down traversal of hierarchy:**
  - on generic MT [Puppo, 1996, De Floriani et al., 1997/98]
  - on PM [Hoppe, 1997, Xia et al., 1997]
  - on restricted quadtrees [Von Herzen and Barr, 1987, Gross et al., 1996]
  - on hierarchy of right triangles [Evans et al., 1997, Duchaineau et al., 1997]
  - on hierarchy of irregular triangles [De Floriani & Puppo, 1995]
- **Bottom-up traversal of hierarchy:**
  - on PM [Xia et al., 1996]
  - on hierarchy of right triangles [Lindstrom et al., 1996]
- **Breadth-first traversal of surface:**
  - on hypertriangulation [Cignoni et al., 1995/97]
  - on hierarchical triangulation [De Floriani & Puppo, 1995]

Top-down on MT

- Visit DAG starting at cut just below its root
- Recursively move cut below a node *n* when a triangle labeling an arc entering *n* has accuracy worse than LOD

Algorithm on explicit MT:

- ▲ Optimally efficient (on MT with linear growth)
- ▲ Applicable to all models
- ▼ Needs expensive data structure

Algorithm on implicit MT:

- ▲ Lighter data structure
- ▼ Slower
- ▼ Output mesh larger
- ▼ Applicable only to special models

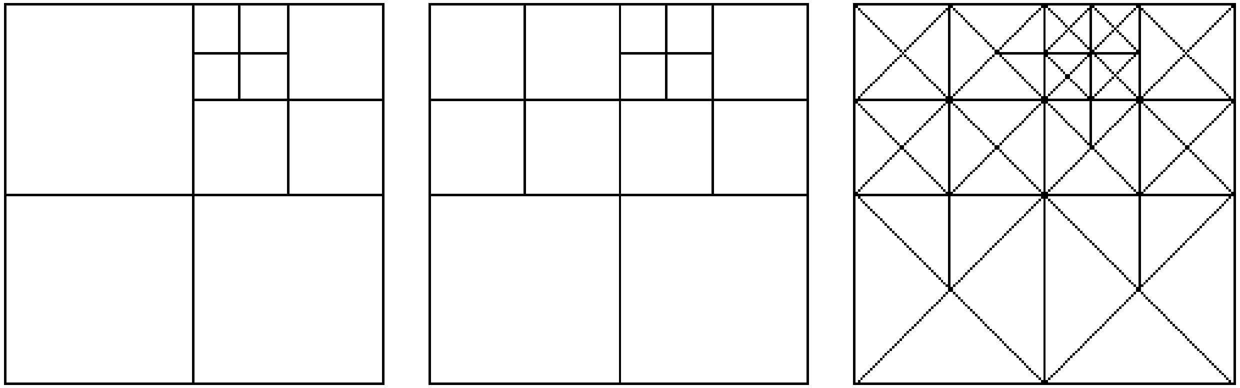
Top-down on PM (vertex forest or DAG)

- Visit forest/DAG starting at topmost level
- Recursively expand a vertex when accuracy worse than LOD
- Find all vertices that constrain selected vertices
- Perform all splits corresponding to selected vertices in the default order

- ▲ Fast
- ▼ Needs expensive data structure

Top-down on restricted quadtrees

- Visit tree starting at its root
- Recursively expand a quadrant when accuracy worse than LOD
- Balance quadtree
- Triangulate each quadrant according to its neighbors
- ▲ Fast
- ▼ No control on error of triangles generated a posteriori
- ▼ Needs suitable procedures for neighbor finding



Top-down on tree of right triangles

- Visit tree starting at its root
- Recursively refine a triangle when accuracy worse than LOD
- Propagate vertex dependencies to obtain a conforming mesh

▲ Easy and fast if all vertex dependencies are available

Top-down on trees of irregular triangles

- Visit tree starting at its root
- Recursively expand a triangle when accuracy worse than LOD
- Triangulate mesh a posteriori to make it conforming

▲ Easy and fast

▼ No control on error of triangles generated a posteriori

Bottom-up on PM (vertex forest):

- Visit forest starting at leaves
- Recursively discard leaves that can be collapsed
- Perform splits corresponding to selected vertices in default order

Bottom-up on hierarchy of right triangles:

- Visit tree starting at leaves
- Recursively merge sibling leaves when possible

▼ All vertices must be analyzed even to extract a coarse LOD

Breadth-first traversal of domain on hypertriangulations and on hierarchical triangulations

- Start with a triangle where highest LOD is required
- Incrementally add triangles adjacent to the boundary of current triangulation through global adjacencies
- Each time a boundary edge *e* is crossed, select a triangle which is as coarse as possible, satisfied the LOD, and is compatible with current mesh

▲ Supports dynamic local refinement/abstraction of detail (resolution editing)

▲ Ideal for propagating LOD through the surface rather than through space

▼ Applicable only for LOD monotonically decreasing with distance from a given point

▼ Computational complexity is super-linear

▼ Needs expensive data structure

Discussion

Taxonomy

- **Modeling issues:**
  - types of surfaces supported
  - data distribution
  - expressive power
  - shape of triangles
  - storage
- **Processing issues:**
  - selective refinement
  - progressive transmission
  - navigation
  - geometric queries
  - construction

Types of surfaces supported

- All types:
  - MT, PM, HyT, quaternary triangulations
- Explicit and parametric surfaces only:
  - quadtrees, restricted quadtrees, hierarchies of right triangles (problems with trimming curves)
  - Adaptive hierarchical triangulations

Data distribution

- Data distribution: irregular distribution vs regular grid

Irregular

MT, HyT, PM, Adaptive hierarchical triangulations

Quaternary triangulations (with remeshing)

Quadtree, hierarchy of right triangles

Regular

Expressive power

- Number of different meshes that can be extracted
- Possibility to adapt a mesh to arbitrary LOD
- Ratio accuracy/size

More expressive, higher ratio	↑	Explicit MT
		PM, Implicit MT
		HyT
		Quaternary triangulations
		Hierarchy of right triangles, quadtree,restricted quadtree,
Less expressive, lower ratio	↓	Adaptive hierarchical triangulations

...Discussion...

Shape

- Regions with regular shape are desirable
- Slivers cause numerical errors, and unpleasant visual effects

More regular	↑	Quadtree, restricted quadtree, hierarchy of right triangles
		Quaternary triangulation
		HyT, MT
		PM
Less regular	↓	Adaptive hierarchical triangulations

Storage

- Compact data structures are desirable
- Increasing complexity: geometry, connectivity, interference, topology

More expensive	↑	Adaptive hierarchical triangulations
		HyT
		Explicit MT, vertex hierarchies
		Implicit MT
		Linear sequences (linear PM)
Less expensive	↓	Quadtree, quaternary triangulation, hierarchy of right triangles

Selective refinement

- High efficiency is desirable

More efficient	↑	Explicit MT
		Hierarchy of right triangles
		PM vertex hierarchies
		Implicit MT
		HyT
		Quaternary triangulation, restricted quadtree
Less efficient	↓	Linear sequences (linear PM)

Progressive transmission

- Compact coding and fast decompression at client are relevant

More efficient ↑

Quadtree, hierarchy of right triangles

Linear PM

Implicit MT, PM vertex hierarchies

Other linear sequences

Explicit MT

Restricted quadtree, quaternary triangulation

Less efficient ↓

Navigation

- Ability to move across surface and through LOD

More efficient ↑	Quadtree, hierarchy of right triangles
	Restricted quadtree, quaternary
	Adaptive hierarchical triangulation
	HyT
	Explicit MT
	Implicit MT, PM vertex hierarchies
Less efficient ↓	Linear sequences

Geometric queries

- Point location, windowing, segment intersection, ray shooting, clip, etc…
- Bounded width and logarithmic height are relevant

More efficient	↑	Quadtree, hierarchy of right triangles
		Restricted quadtree, quaternary
		Adaptive hierarchical triangulation
		Explicit MT
		HyT
		Implicit MT, PM vertex hierarchies
Less efficient	↓	Linear sequences

Construction

- Lower time complexity of construction algorithm is desirable
- Easy-to-code algorithms are desirable

More complex, more difficult	↑	Adaptive hierarchical triangulations
		HyT
		MT, PM vertex hierarchies
		Linear sequences (linear PM)
		Quaternary triangulation
Less complex, less difficult	↓	Quadtree, hierarchy of right triangles