matfyz

Computer Graphics Charles University

Data for real-time graphics

© 2005-2018 Josef Pelikán

CGG MFF UK Praha

pepca@cgg.mff.cuni.cz

http://cgg.mff.cuni.cz/~pepca/

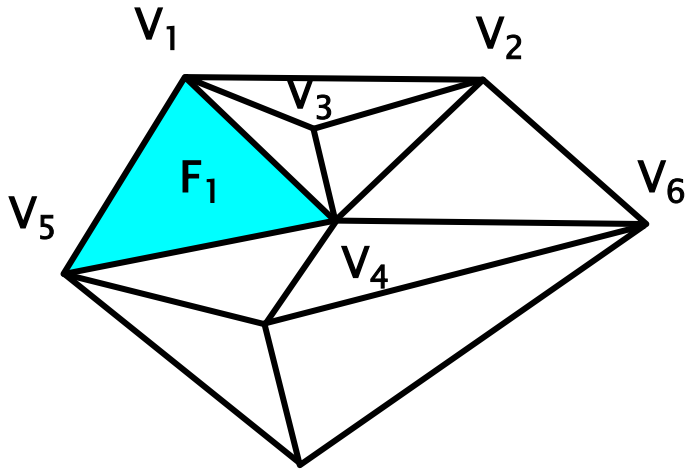
Content

- boundary representations, triangle meshes
  - efficient CPU → GPU transfer
- scene hierarchies
  - articulated hierarchy, skeleton
- "Level of Detail" (LoD)
  - discrete and continuous LoD
  - terrain LoD (height-map based)
- billboards, imposters, point-sprites

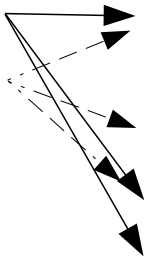
3D scene representation

- in general there are more approaches ..
- .. but in real-time graphics **boundary representation** (**B-rep**) is most popular
  - only object surface
  - the simplest forms: **triangle meshes** or point-based surfaces (colored 3D points)
- **geometry** + **topology**
  - vertex coordinates + primitive assembly
  - vertices are **shared** among adjacent faces

Triangle mesh I



F_1	V_1,V_4,V_5
F_2	V_1,V_3,V_4
F_3	V_2,V_3,V_1
F_4	V_2,V_4,V_3
...	

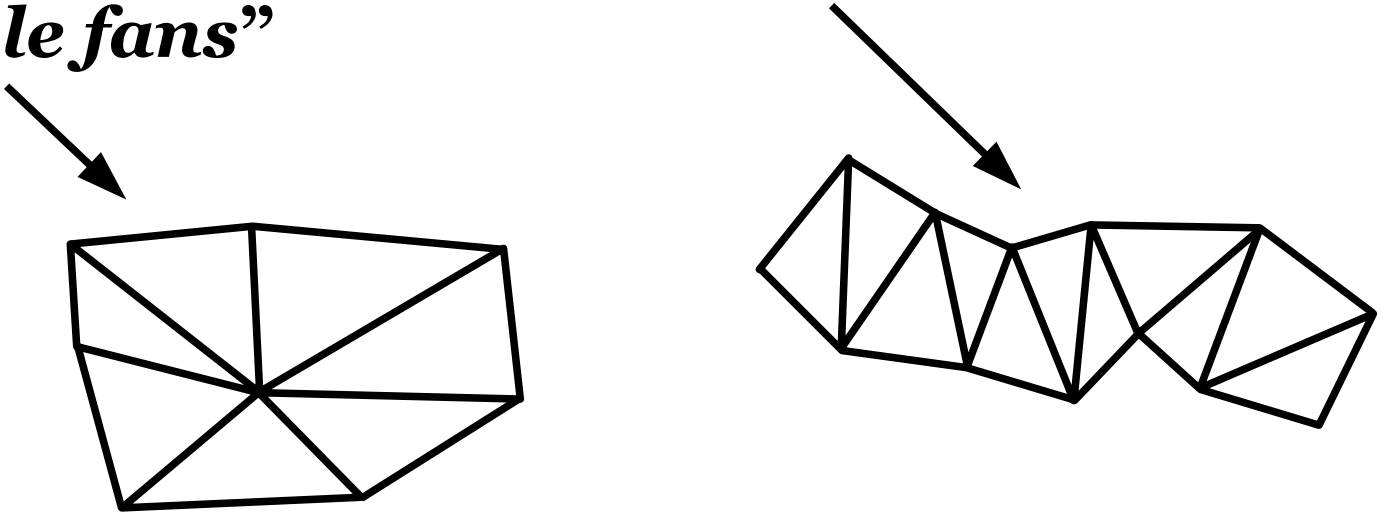


V_1	x,y,z,w
V_2	x,y,z,w
V_3	x,y,z,w
V_4	x,y,z,w
V_5	x,y,z,w
...	

- similar to a **relational database**
  - **vertex table** (V, „*vertex*"), [edge table (E, „*edge*")], **face/triangle table** (F, „*face*") are interconnected
  - each entity can have additional **attributes** (vertex: normal vector, color, .., face: texture, material, ..)
  - references = **indices** (`int` or `uint`)

Triangle mesh II

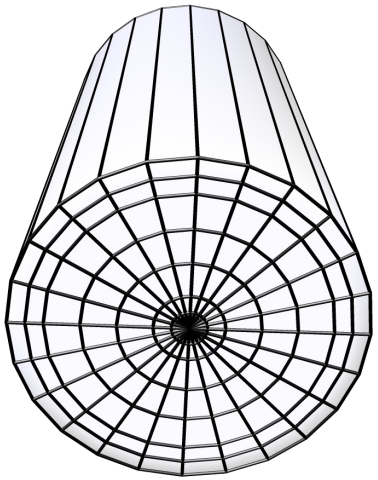
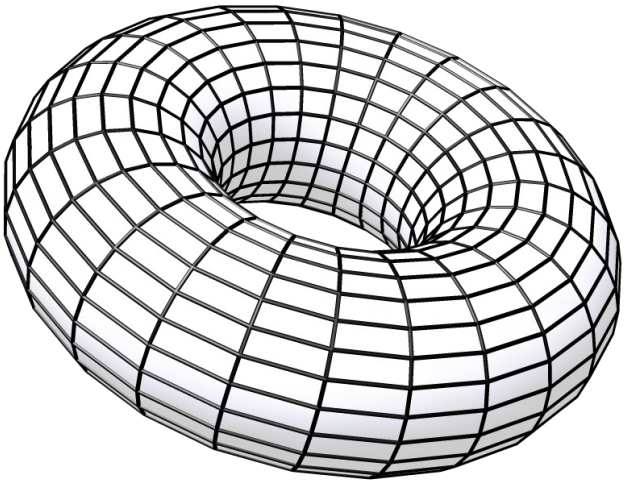
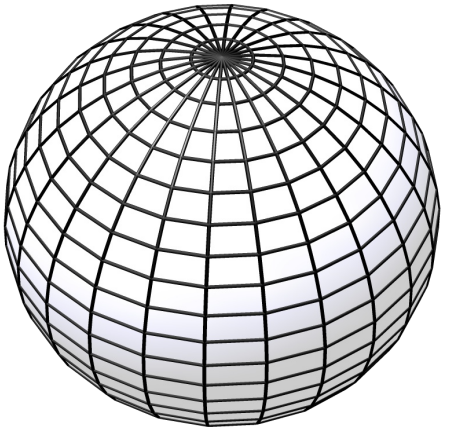
- edges are rarely used (not in HW)
- GPU-efficient data access:
  - pre-loaded **vertex array** (server-side "VBO", "VAO")
  - vertices are referenced by integer **indices** ("index buffer")
  - HW-friendly primitives: "***triangle strips***", "***triangle fans***"



Smooth shape approximation

- **large number of triangles** is often needed for a good appearance
  - hundreds faces for a sphere, tens of thousands for a human body
- **shading techniques** can help a little
  - color interpolation, normal vector interpolation
- **distant objects need not be in full detail !**
  - … "***Level of Detail***" (LoD)

Smooth shape approximation

- even this accuracy might not be sufficient
  - see contours of these “smooth” solids

Hierarchies

- scene consists of objects
  - objects consist of components
    - components consist of parts
      - » parts consist of ...

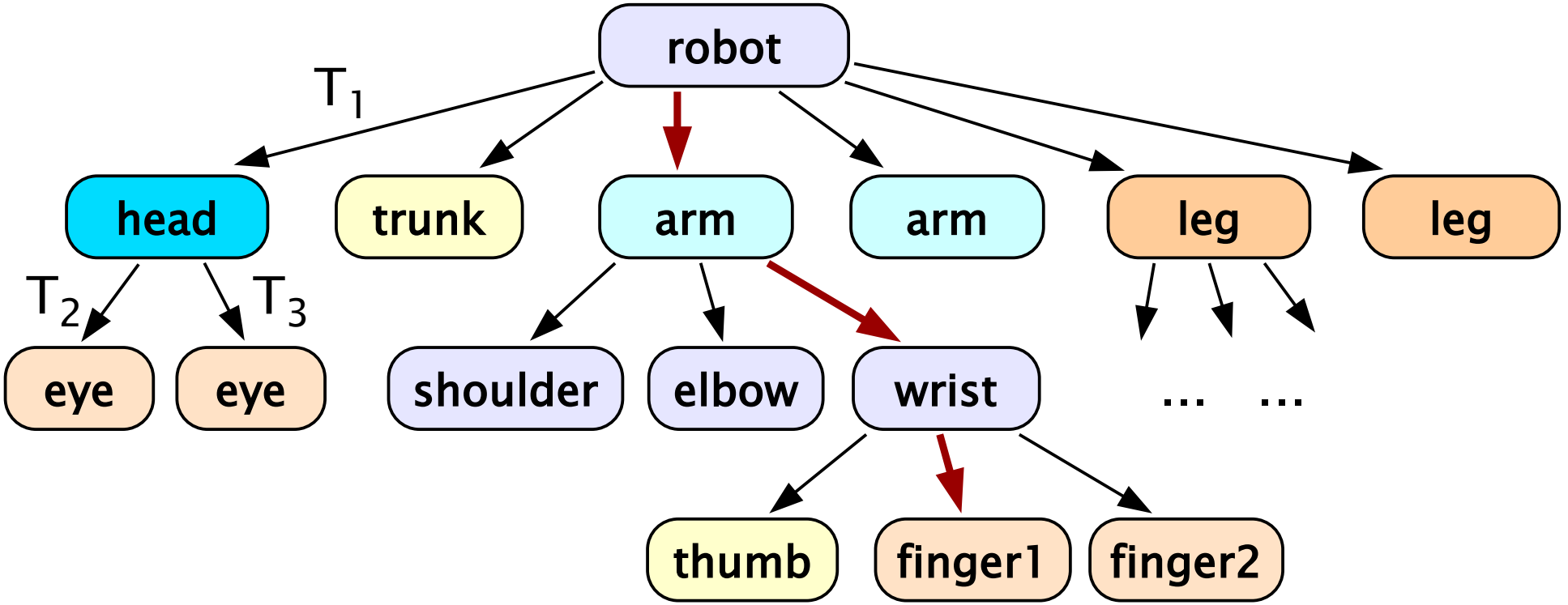
Hierarchical modeling is natural and efficient

- object/component databases to choose from
- additional features:
  - optional **attributes** associated with individual scene nodes (inheritance, parameterization)
  - relative **transformation matrices** (ancestor ↔ descendant connection)

Scene tree

Suitable for **storage**, net **transfers** (memory efficiency)

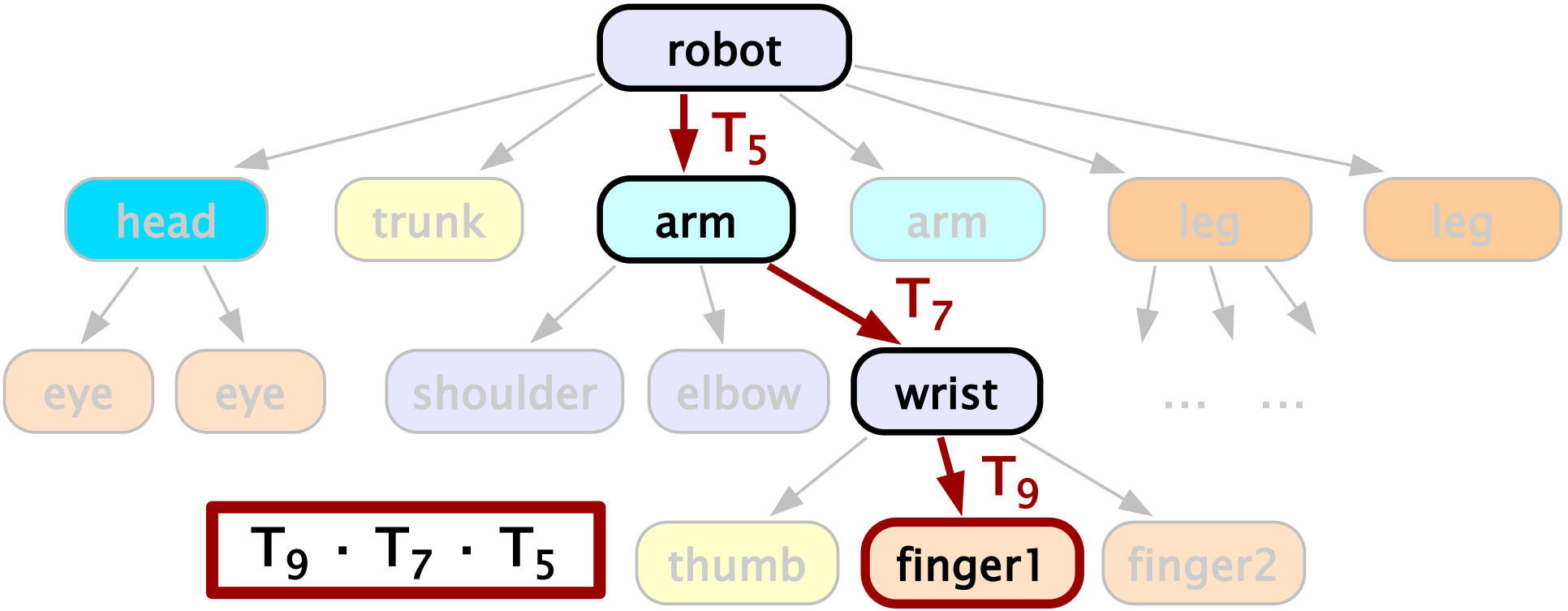
- multiple references have to be unfolded in memory



Relative transformations

Scene **leaf node** (triangle mesh) is transformed into world coordinates by a sequence of relative transformations

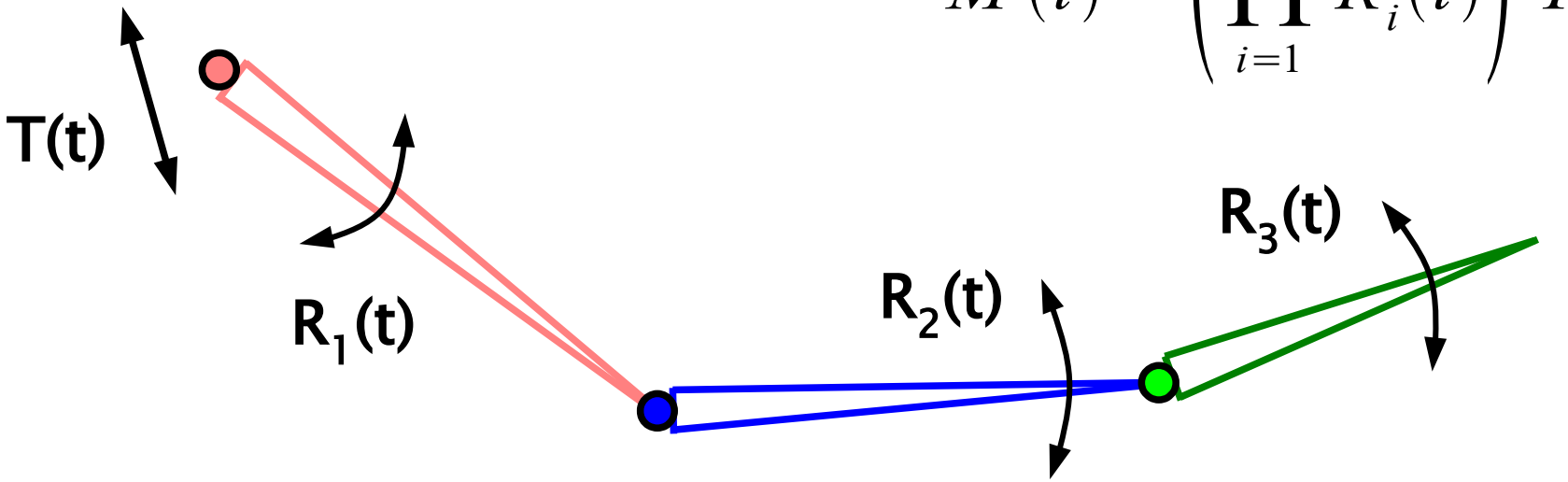
- matrix transformations are computed on a **GPU**



Skeletal animation

- **"bone hierarchy"** (perceived as coord-system hier.)
  - **relative** transformations – rotation relative to an ancestor + translation (matrix, quaternion): $\mathbf{R_i(t)}$
  - **global translation** (the whole object): $\mathbf{T(t)}$
  - **total** transformation:

$$M(t) = \left(\prod_{i=1}^{n} R_i(t) \right) \cdot T(t)$$



Level of detail (LoD)

Optimal **rendering efficiency**:

- **details of distant objects** (~pixel size) need not be drawn
- **the closest objects** (and/or user focus) need the best available visual quality

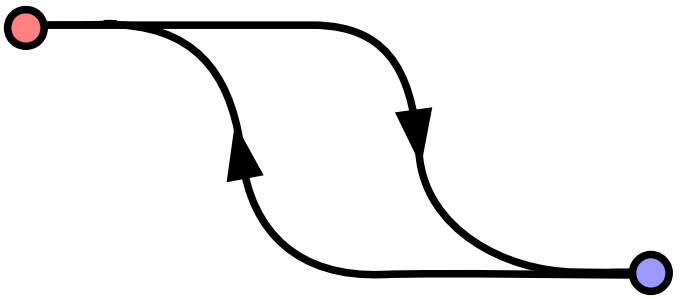
- **dynamic level of detail**
  - rendering system adjusts individual rendering accuracy
  - global parameter definition (e.g. total approximate number of drawn triangles)
  - advance data preparation: discrete LoD levels / continuous LoD pre-processing..

Discrete LoD

- fixed **LoD levels** are prepared in advance
  - shape approximations with different accuracy
  - can come from the finest model – **generalization** (can be time consuming)
- **rendering – choosing** appropriate LoD level:
  - according to object-camera distance
  - according to bounding object projection size
    - or even exact object projection – errors perpendicular to viewing direction are most noticeable
  - object importance, "player focus", ..
  - global balancing (declared number of triangles to draw)

LoD transitions

- **simple switching**
  - hysteresis must be introduced ! (against unwanted "popping")



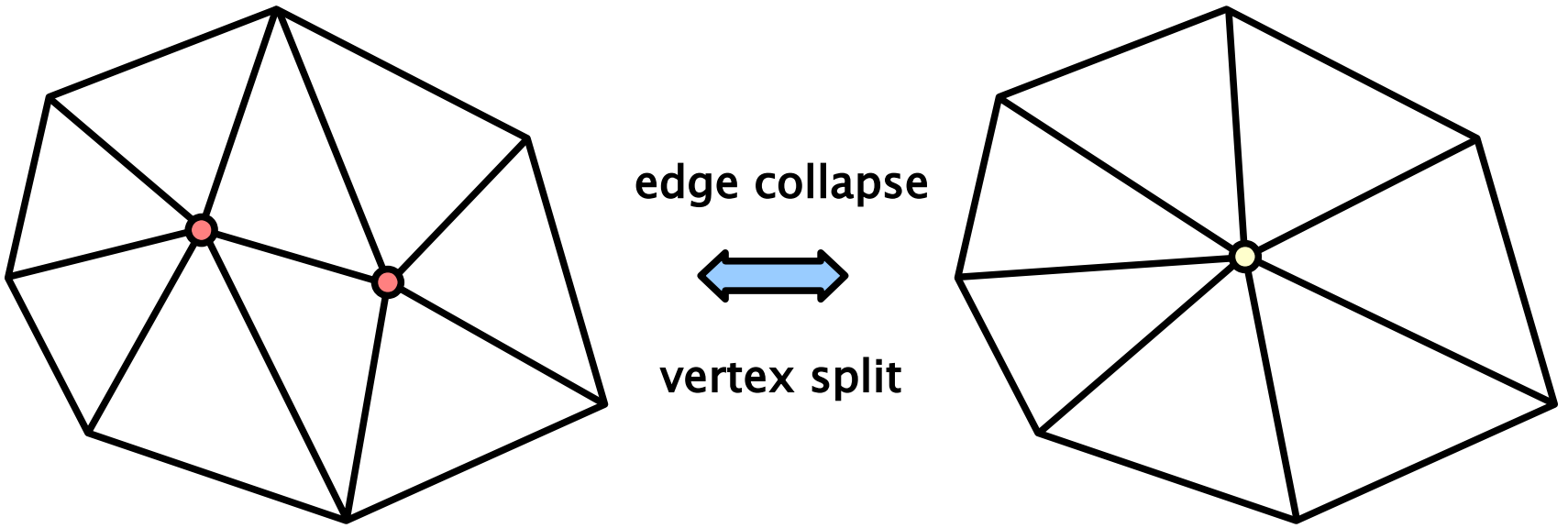
- **blending** neighboring LoD levels
  - drawing both neighboring levels using **transparency**
    - linear combination (blending, "transition")
  - another option
    - current LoD level is opaque
    - additional semitransparent new LoD ("over" **α** operation)
    - "z-writing" enabled only for the current LoD level

Incremental LoD

- reduce popping
  - ~ continuous transition: number of elementary transit.
  - time-consuming preparation
- **elementary operations** on triangle meshes:
  - **edge collapse** (collapses two incident triangles as well)
    - 1 vertex is eliminated together with 3 edges, 2 triangles
  - **vertex split** (inverse operation)
    - new vertex + 3 edges, 2 triangles
  - transition animation: remaining vertex ↔ edge center

Elementary operations I

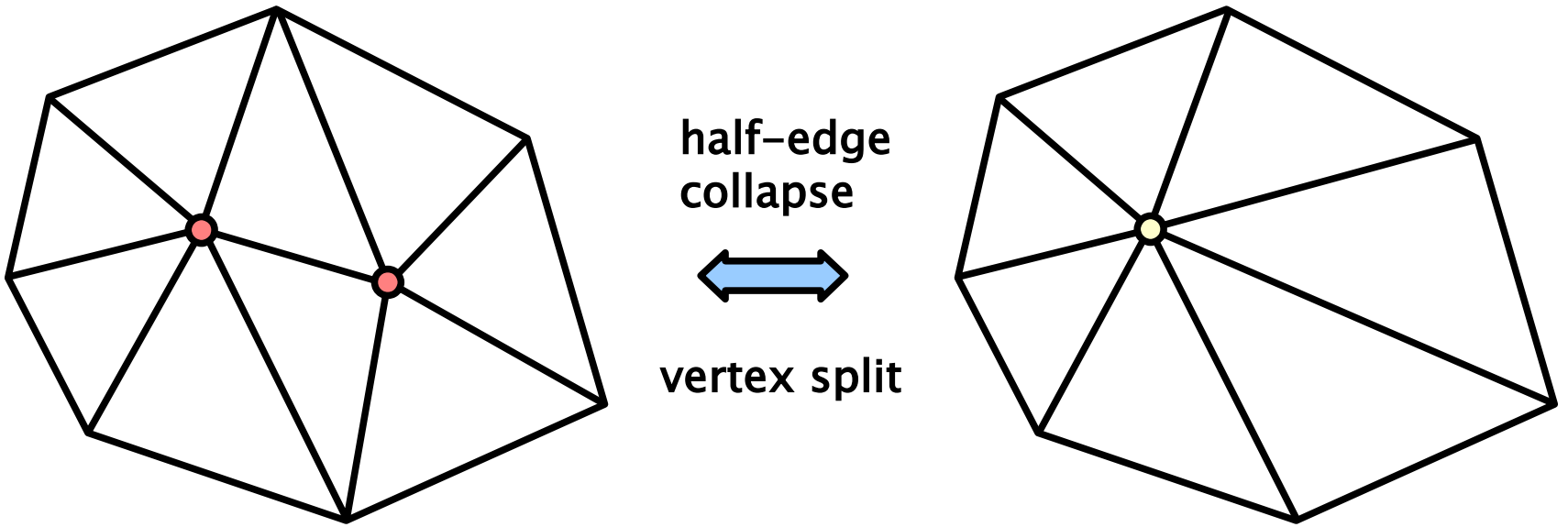
- **edge collapse** vs. **vertex split**:



- possibility of transition animation

Elementary operations II

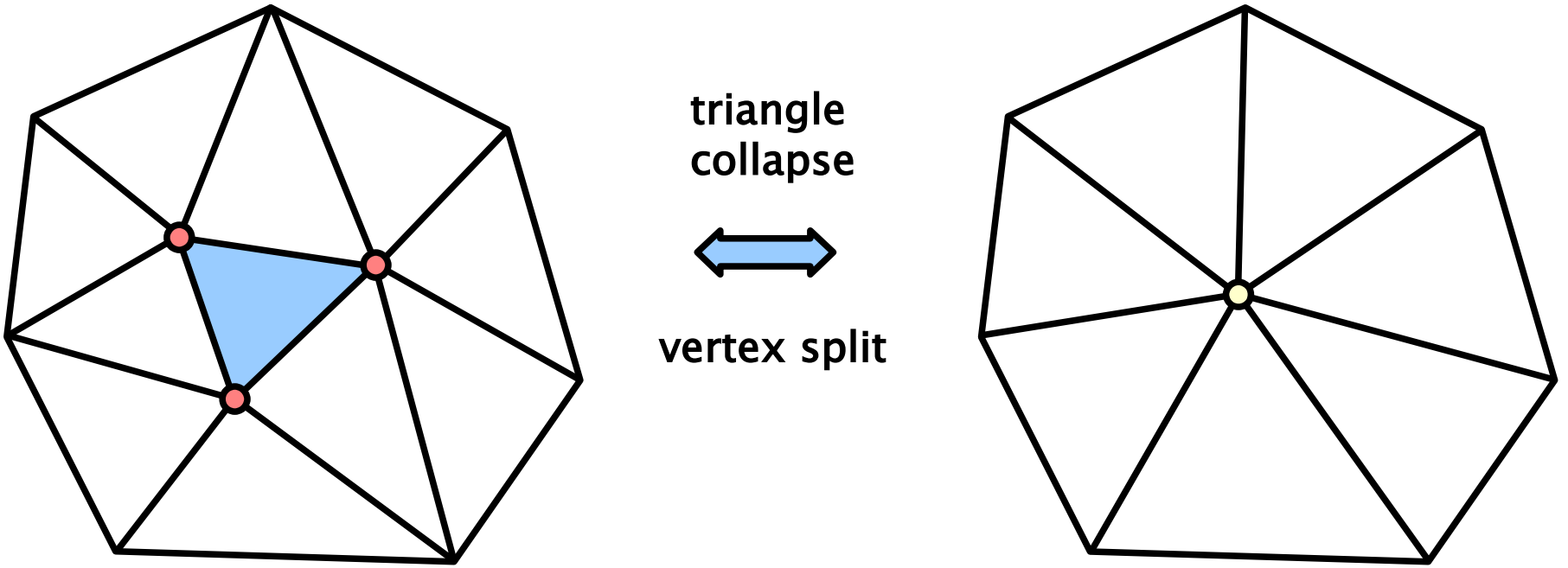
- **one vertex** can remain in its position:



- form of vertex removal (see later)

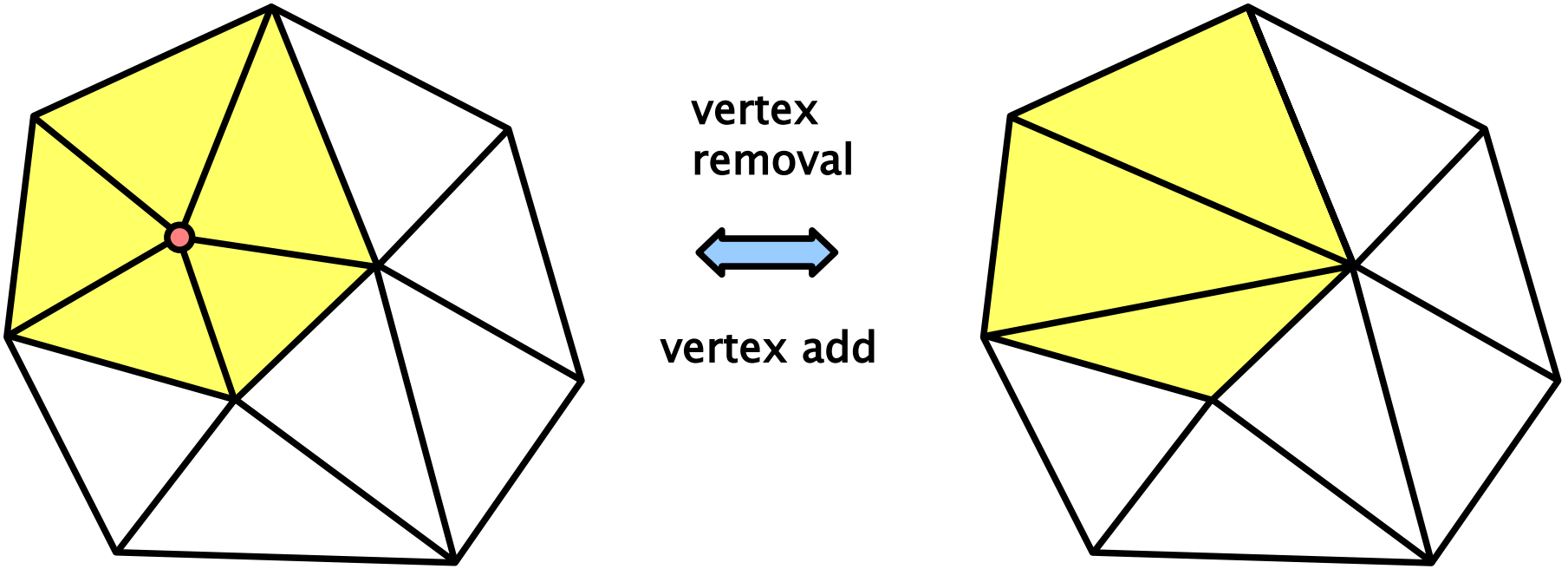
Elementary operations III

- **triangle collapse** vs. **vertex split**:



Elementary operations IV

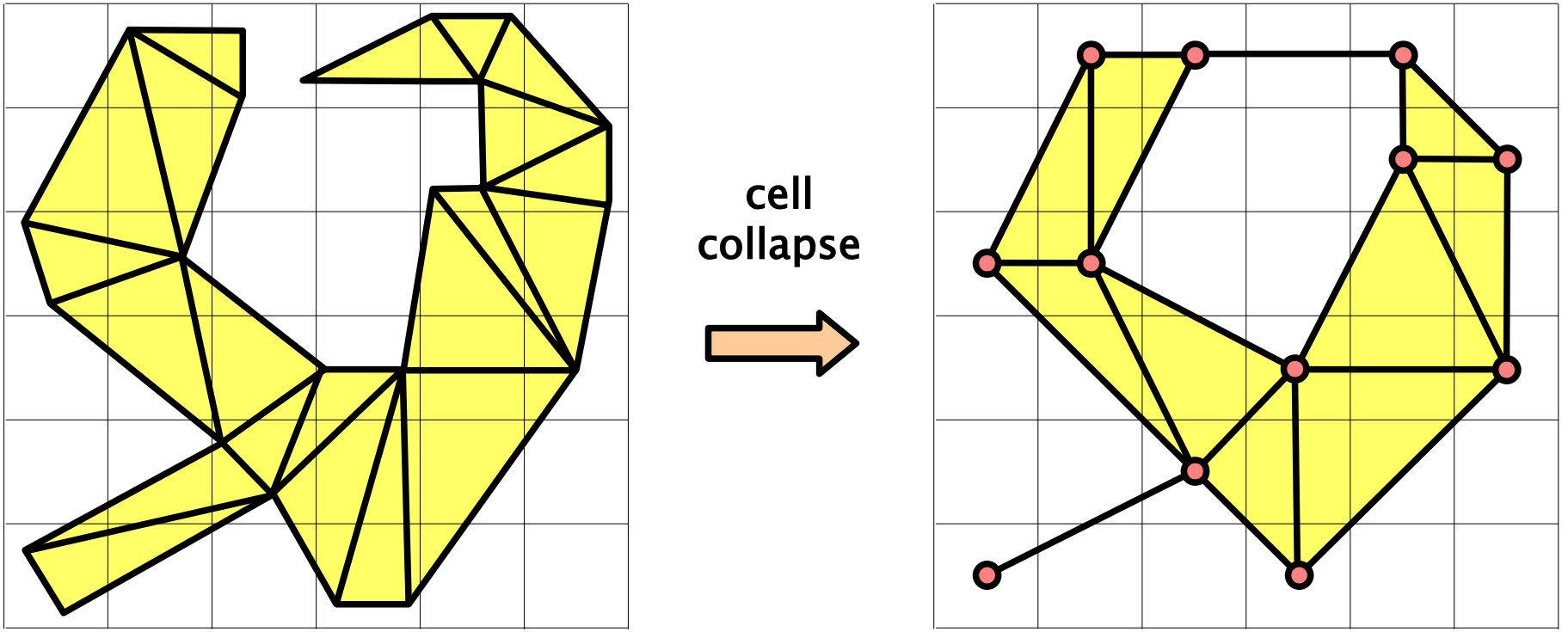
- **vertex removal** vs. **vertex add**:
  - re-triangulation of the polygon



Cell collapse

- **grid of cells**: all **vertices sharing the same cell** will collapse into one (in the cell center)

  + induced topological changes (hierarchy .. octree)



Generating LoDs

- **top-down approach**
  - based on the simplest shape representation (low-poly), additional details are introduced
  - less frequent (subdivision surfaces..)
- **bottom-up**
  - starts with detailed (most accurate) model
  - gradual simplification / generalization (data reduction)
- various **optimizations**
  - "finest" representation using fixed number of faces
  - global optimization is NP-complete

Optimization for creating LoDs

- **w/o optimization** (e.g. cell collapse)
- **"greedy"** algorithms
  - criterion function – error metric
  - all simplification possibilities are considered (many evaluations of a metrics)
- **"lazy evaluation"**
  - greedy approach with reduced computation
  - after a local change neighborhood is marked as **"dirty"** but metrics is not recomputed yet ..
  - less nice results, but more efficient, ..

Optimization for creating LoDs

- **“metric estimation”**
  - faster metric estimation
  - can be combined with lazy evaluation, three states:
    - **dirty** (not re-computed)
    - **estimate** (approximate value)
    - **exact** value
- **independent simplification**
  - set of **independent elementary operations** (do not interfere, can be processed in parallel)
  - **hierarchy** – logarithmic number of levels (previous methods were all linear)

Error metrics

- strictly **3D geometry** based
  - not accounted: view direction, projection parameters
- based on **projection** (on result – "target driven")
  - more efficient – directly connected to visual error
  - contours are more important than shape interior
  - considering changes in contrast texture mapping
- **attribute error** metrics
  - color (frequently improperly interpolated)
  - normal vector (affects shading)
  - texture mapping

Concrete metrics I

- **vertex to vertex** distance
  - for cell collapse
  - for approximation of more advances metrics
- **vertex to plane** distance (SP – Ronfard, 1996)
  - $\mathbf{d = p \cdot v = n_x v_x + n_y v_y + n_z v_z + D}$
  - **"supporting planes" (SP)**: in original mesh – planes of all faces incident with the given vertex
  - **edge collapse**: SP of the new vertex = union of SPs of the original vertices
  - maximum of all SP: $\mathbf{Err = max\ (p \cdot v)^2}$

Concrete metrics II

- **quadric error** (Garland & Heckbert, 1997)

  $\mathbf{Err = \Sigma\, (p \cdot v)^2 = \Sigma\ v^T Q_p v = v^T Q_{all} v}$

  - $\mathbf{Q_p}$ symmetrical 4×4 matrix (10 different values) representing plane **p**
  - quadrics (matrices) can be **summed together** ($\mathbf{Q_{all}}$)

- **vertex to surface** distance
  - complicated, can be simplified by surface sampling

- **surface to surface** distance
  - approaches: geometry, texture, bounding volumes, ..

Constant frame rate I

- elementary **reducing** operations (e.g. edge collapse)
- elementary **refining** operations (e.g. vertex split)
- dynamic **reducing (merge) and refining (split) lists** (based on chosen error metrics)
- **reducing list**: the detail level is no more needed
  - higher priority (reduces fill-rate)
- **refining list**: more detail is needed (appearance)
  - "temporal disjunction" with the reducing list
  - total data volume should be checked while processing this list

Constant frame rate II

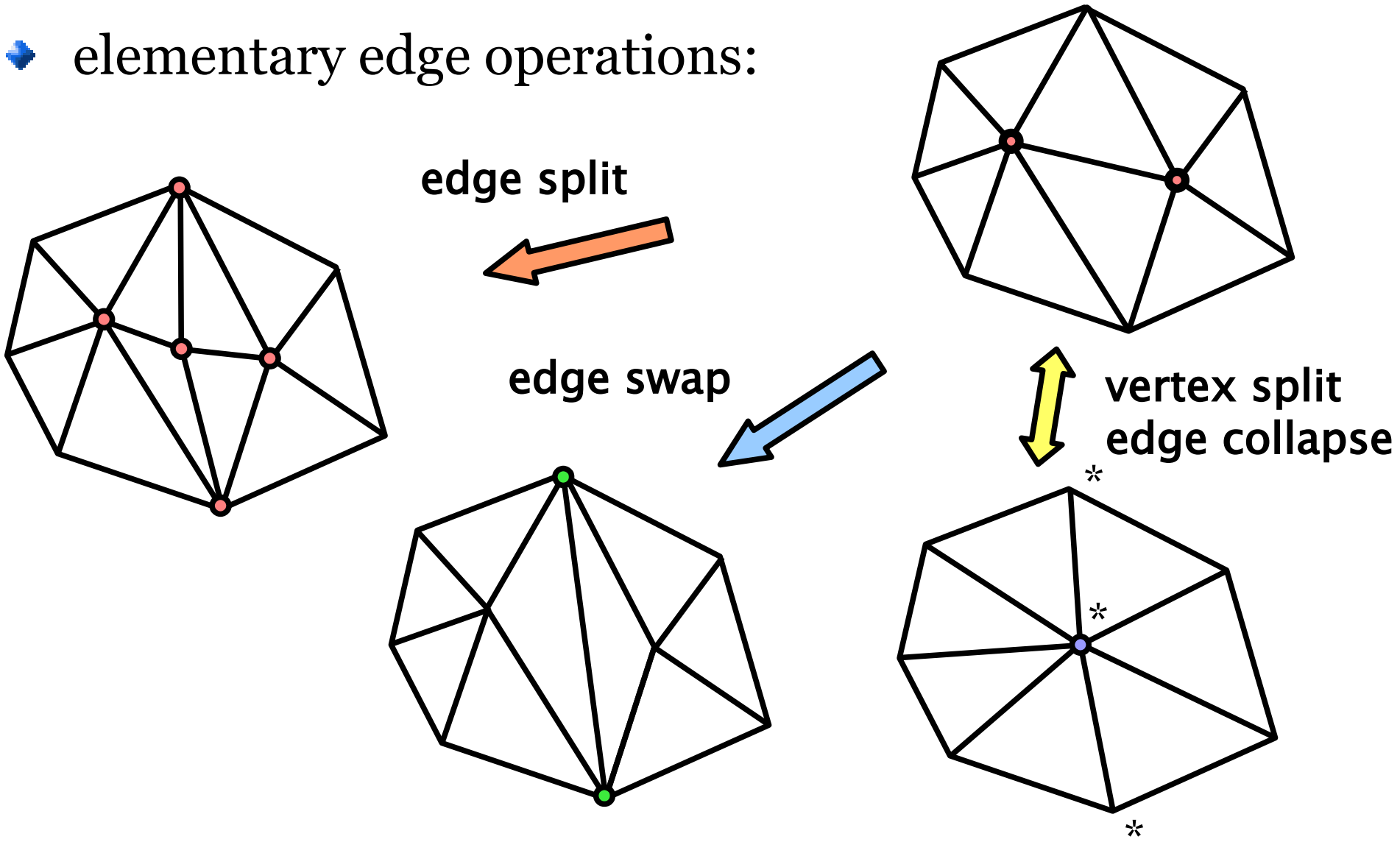
- possible **multi-thread implementation**
  - thread #1: **scene rendering** and perhaps executing actual **LoD operations** (reducing, refining)
  - thread #2: asynchronous **list maintenance** (metrics recomputations, reducing/refining suggestions)
  - thread #3 [optional]: rendering support, executing **LoD list operations**

  - **feedback** from rendering engine:
    - fine-tuning metrics parameters
    - total (maximal) number of triangles

Continuous LoD

- **Progressive meshes** (Hugues Hoppe, 1996)
  - initial triangle mesh + sequence of elementary operations
  - uses previously published optimization techniques
  - suitable for progressive data transfer (Internet) or geometry compression
- **View-dependent PM** (Hugues Hoppe, 1997)
  - possibility of independent adaptive mesh refinement
  - not based on pre-computed linear refinement sequence

Progressive Meshes I



Progressive Meshes II

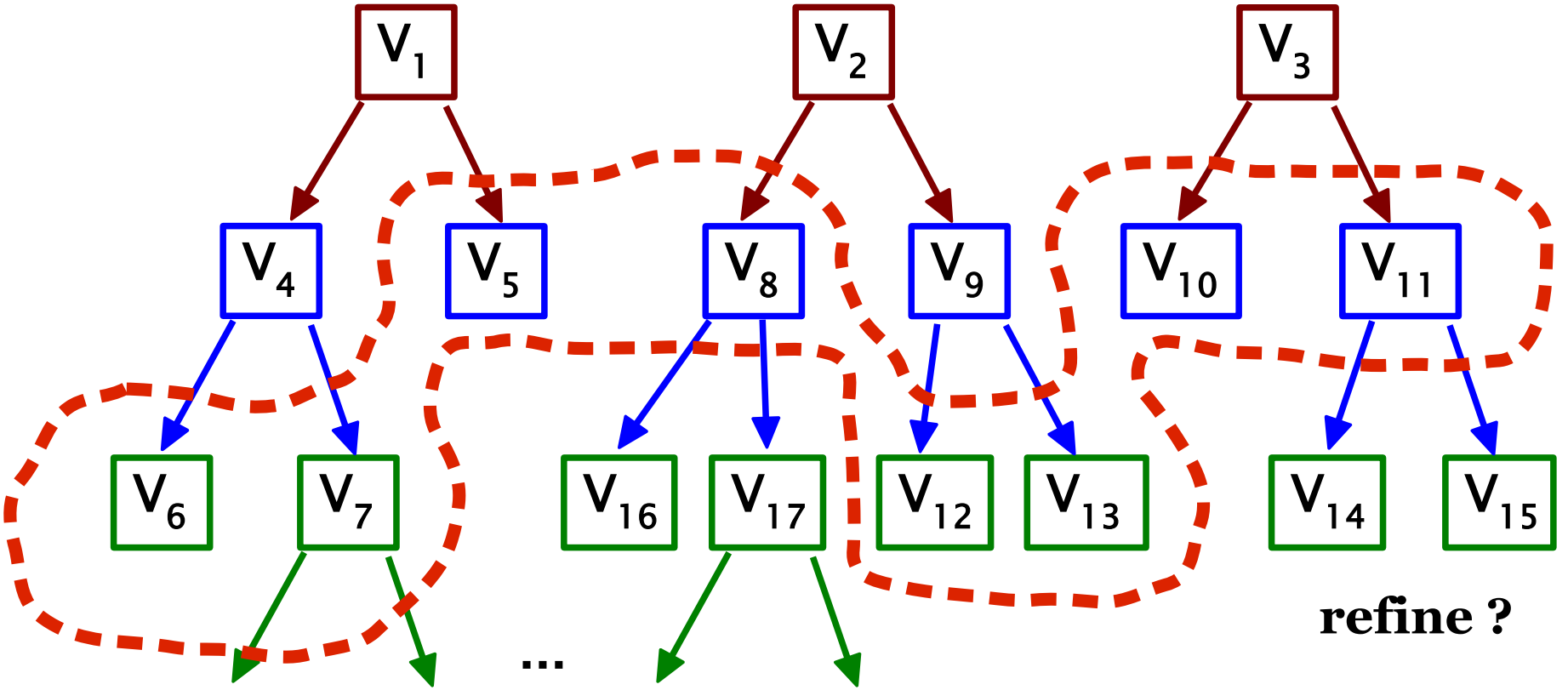
- **triangle mesh** representation
  - initial mesh $\mathbf{M_0}$
  - sequence of "vertex split" operations $\mathbf{vs_0}$, $\mathbf{vs_1}$, ..
  - each operation must store: [$\mathbf{V_s}$, $\mathbf{V_l}$, $\mathbf{V_r}$, $\mathbf{A}$]
    (reference to three old vertices* and attribute: positions of two new vertices)
- **"geomorphs"**
  - possibility of transitional animation (one **vs**)
  - two new vertices are gradually "detaching" from the original vertex

Progressive Meshes III

- **selective refinement**
  - initial (current) mesh can be refined by using different sequence $\mathbf{vs_{io}}$, $\mathbf{vs_{i1}}$, ..
  - **"out-of-order"** operation can be executed if the three relevant vertices ($\mathbf{V_s}$, $\mathbf{V_l}$, $\mathbf{V_r}$) are already present
  - **order** of the refinement is controlled by a metrics
    - e.g. visibility of a vertex or its presence in a object outline polygon
  - if a critical place does not meet the "dependency" requirement ⇒ consider induced refinements
    - dependency graph can be too large..

View-dependent Refinement of PM I

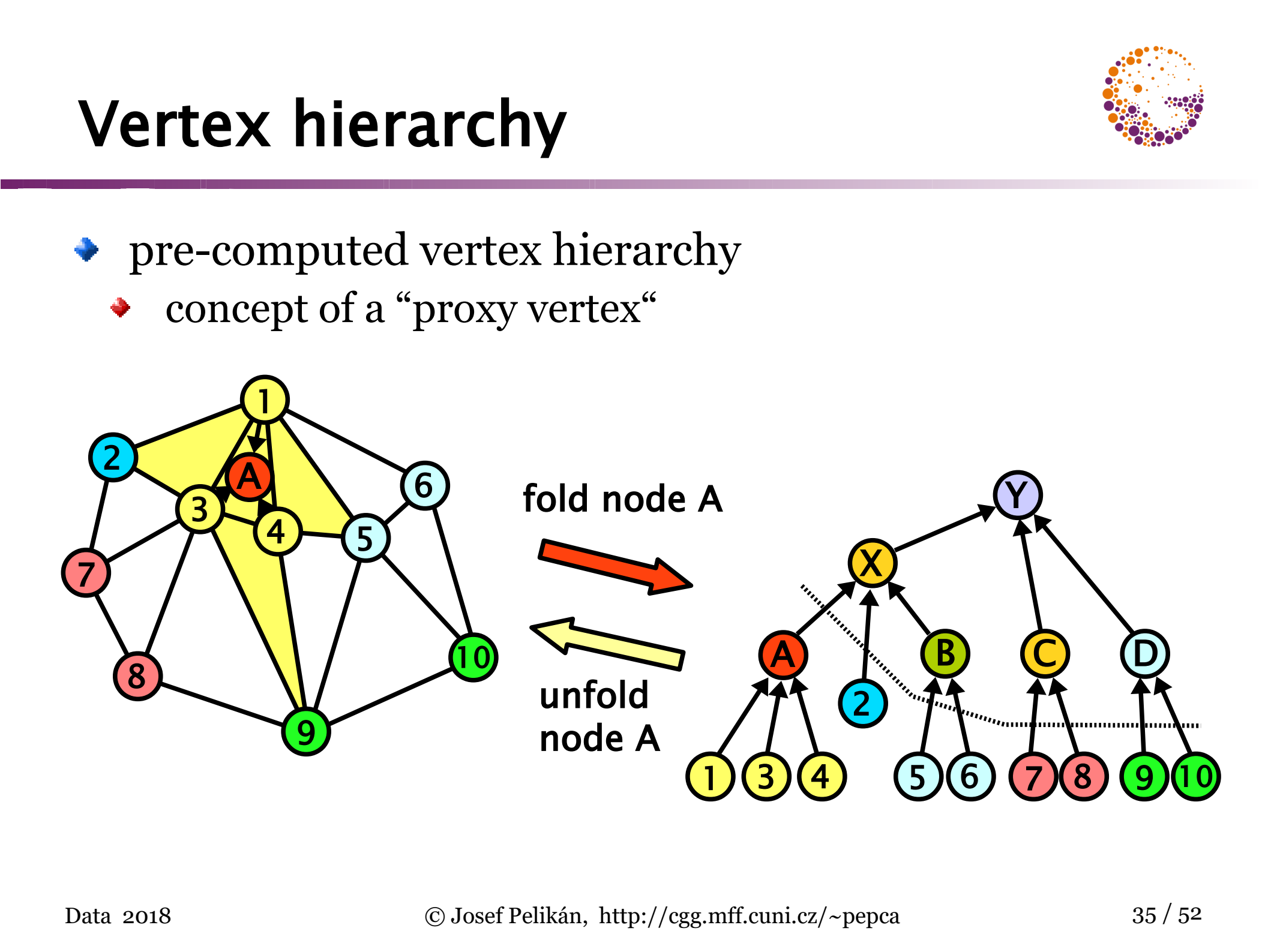
- selective systematic refinement based on PM



View–dependent Refinement of PM II

- **currently displayed mesh** form a graph-cut-like set (“active vertices“)
  - graph node = mesh vertex
  - edge = “vertex split“ (opposite .. “edge collapse“)
- local refinement of a drawing … “cut” moves down
  local reduction/generalization … “cut” moves up
- **real-time rendering**
  - cyclic passing through active vertices
  - evaluation: where a refinement/reduction is needed
  - amortization: mesh update can be asynchronous (dedicated thread)

Vertex hierarchy



- pre-computed vertex hierarchy
  - concept of a “proxy vertex“

Terrain – Earth's surface

- very **extensive data**
  - **USGS** (U.S. Geological Survey): **1km** net – **2 billion** triangles for the whole Earth's surface
  - other resolutions: up to **3m** (1/9 arc second) !
- **regular topology**
  - most frequently used (implementation simplicity)
  - square or rectangular grid
  - rarely – hexagonal grid (equilateral triangles)
- **triangular irregular network** ("TIN")

DEM example (from USGS)

National Elevation Dataset (NED)

Coordinate conformance:
North American Datum 1983 (NAD83)

Horizontal resolution:
1-arc-second (~30m)
1/3-arc-second (~10m)
1/9-arc-second (~3m)

Vertical resolution:
1m

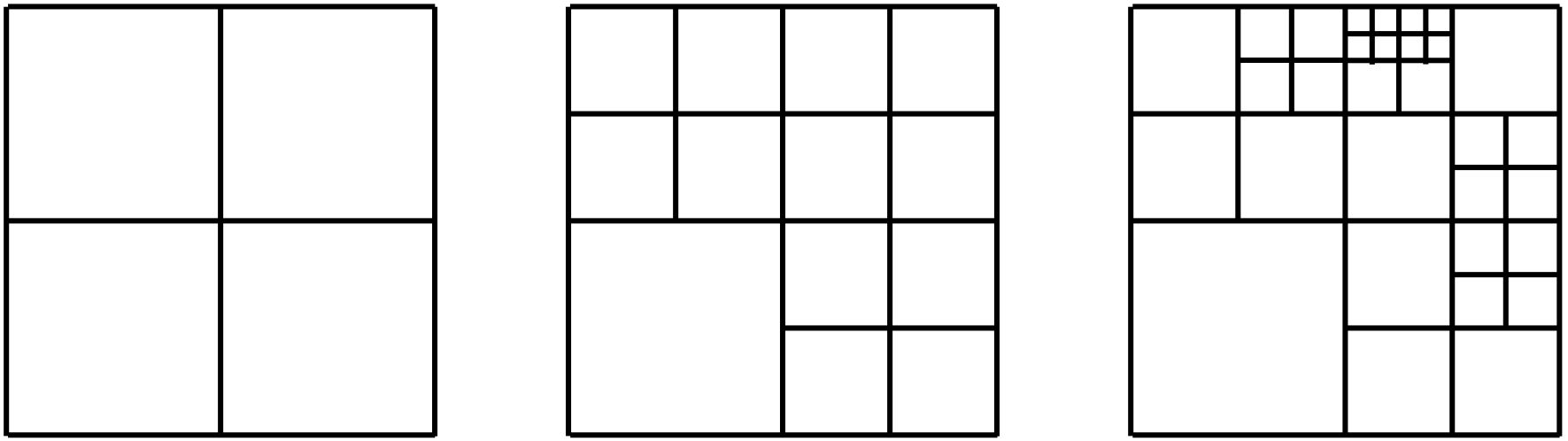
Binary data format:
single file covering 15x15-arc-min area

Pixel resolution (example):
105770 x 65098

`http://nationalmap.gov/elevation.html`

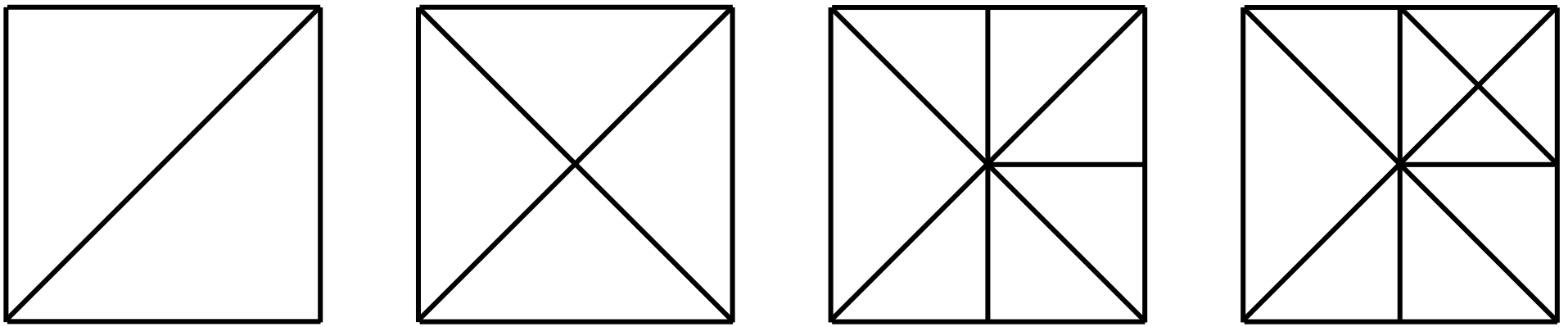
Quad–tree

- **recursive subdivision** of regular rectangular mesh
  - 4-way (quad-tree) or 2-way (bintree)
- **Quad-tree**:



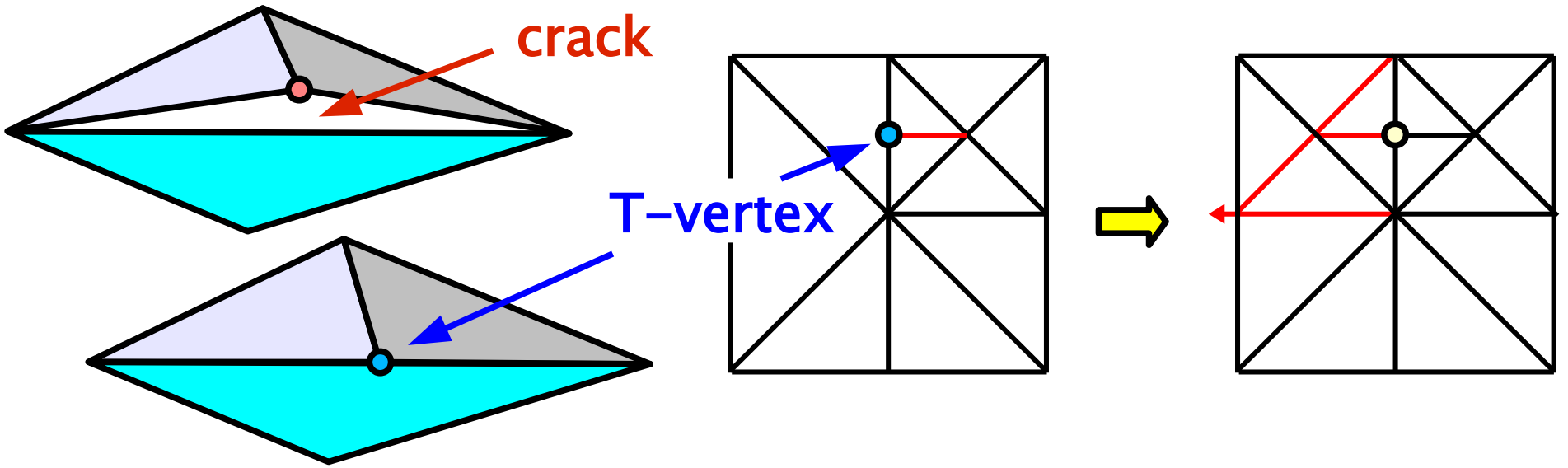
Triangle bintree

- **isosceles right-angled triangles**
  - contains T-vertices (just as quad-tree)



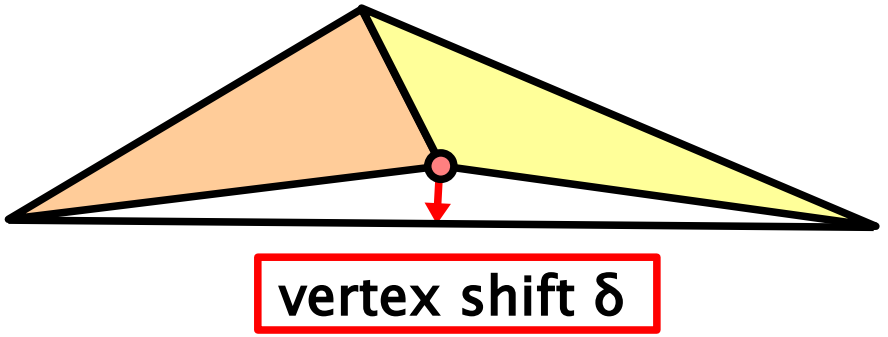
Cracks and T-vertices

- caused by **different levels of subdivision**
  - **crack** can be eliminated by moving the vertex to the center of the incident edge
  - **T-vertex** can still be problematic (color interpolation) … recursive induced subdivision of neighbors



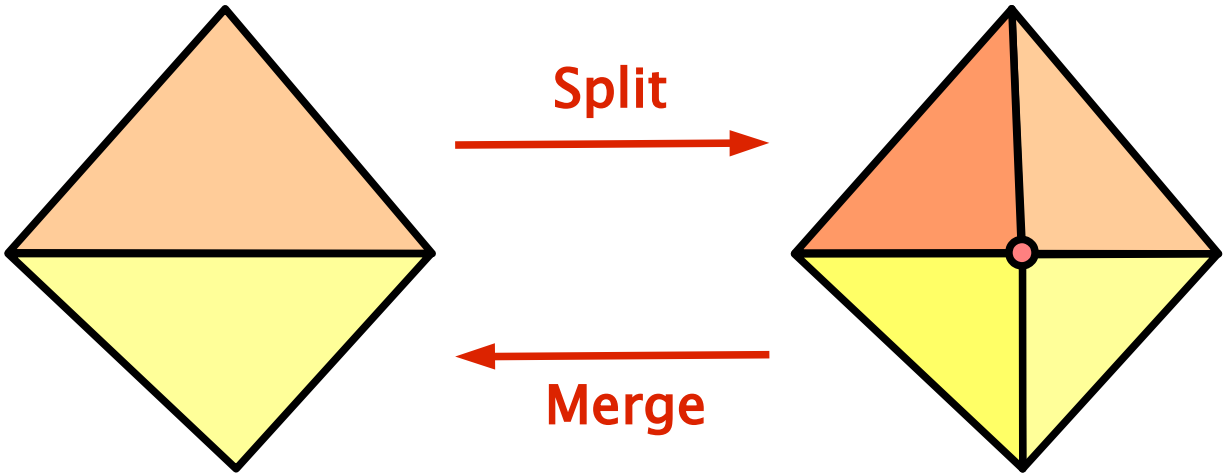
Lindstrom 1996

- algorithm based on **triangle bintree** (DEM)
- initial most fine-grain terrain mesh
  - **metrics**: projection of vertex shift to screen space (1px)
- **T-vertices** eliminated by induced subdivision
- **compact representation**: 32 bits per vertex (**h**, **δ**, flags)



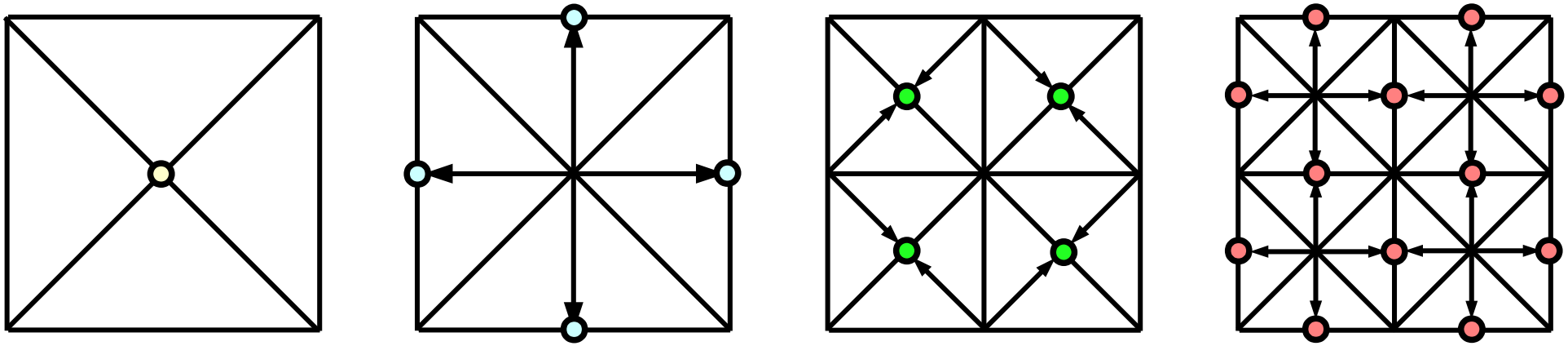
ROAM algorithm (Duchaineau 1997)

- based on **triangle bintree** (needs not be DEM)
- **neighbor triangles** with common hypotenuse are merged/split
- **dynamic** simplification and subdivision (screen error)
  - priority **split queue**
  - priority **merge queue**



Lindstrom & Pascucci 2001

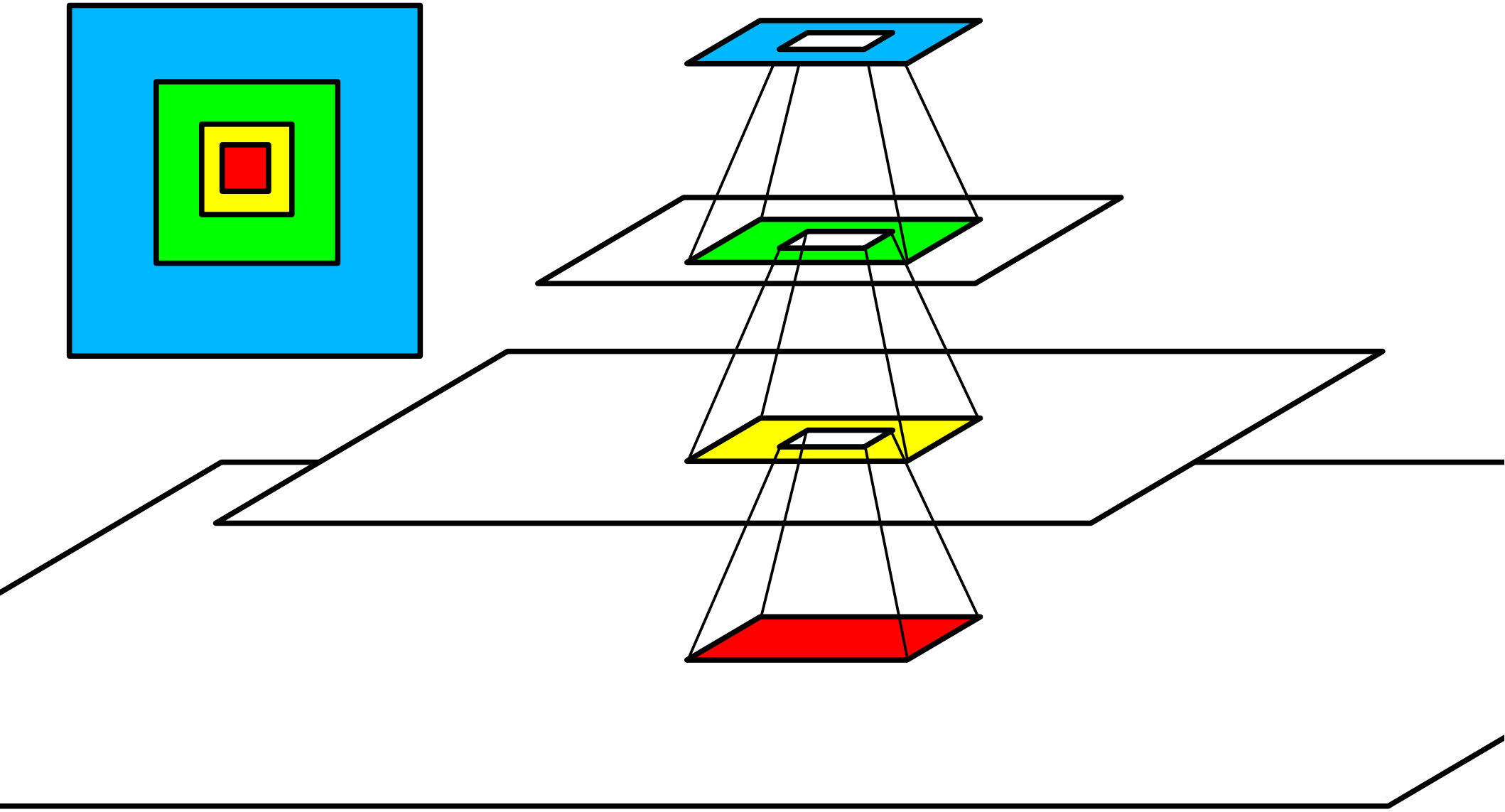
- simple approach **independent on a metrics**
- regular square mesh (Digital Elevation Model)
- **hierarchical system** – parent node is introduced prior to descendant nodes (crack elimination)



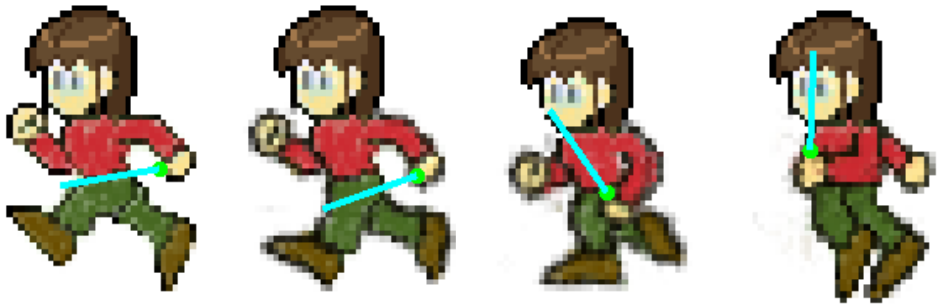
Geometry Clipmaps (2004)

- Losasso, Hoppe (2004)
- built over a **regular square mesh**
- **hierarchical system** à la "MIP-map"
  - pre-computed coarse-grained levels
- efficient **GPU implementation** is possible:
  - base grid remains unchanged (VBO, virtualization … loading)
  - viewer position change ⇒ index-buffer change
  - relatively simple scheme based on viewer-terrain distance (pyramid)

Geometry Clipmaps



Point sprites



- **point-like object** is rendered using a small texture (sprite)
  - transparency
- **applications**: particle systems, point-based surfaces
- texture mapped on a small rectangle **parallel to the projection plane**
  - point size is still used
  - fragment shader receives coordinates of a fragment within a point-sprite (gl_PointCoord)

Billboards

- "**Billboard**" – semitransparent texture showing **more complicated object/scenery**
  - texture is usually mapped on a rectangle
  - often perpendicular to view direction
  - .. following the viewer – special transform matrix
  - rotation around vertical axis only (unsightly from above)
- usage
  - **trees** and **bushes** (even unoriented billboards & multi-billboards)
  - complex **inscriptions**, **2D graphics**, HUD, lens flare..

Imposters

- **"Imposter"** – billboard created dynamically (as necessary) in a rendering engine
  - **cache** of complex scenery (not very dynamic)
  - complex object/scenery (geometric or color complexity)
  - for **distant objects** mostly
  - hierarchy, LoD, multiple instances of the (almost) same object..
- **technique**: HW render-target textures (pbuffer)
- **trees, bushes**
  - imposters might be oriented along main branches..

Examples I

© Linda (Bohemia Interactive)

Examples II

© Silvador Rapid (BIsim)

Sources

- Tomas Akenine-Möller, Eric Haines: ***Real-time rendering, 2nd edition***, A K Peters, 2002, ISBN: 1568811829
- D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, R. Huebner: ***Level of Detail for 3D Graphics***, Morgan Kaufmann, 2002, ISBN: 0321194969
- J. Žára, B. Beneš, J. Sochor, P. Felkel: ***Moderní počítačová grafika***, 2nd edition, Computer Press, 2005, ISBN: 8025104540

Sources

- **http://vterrain.org/LOD/Papers/**
  (Terrain LOD: Runtime Regular-Grid Algorithms)
- **http://research.microsoft.com/~hoppe/**
  (Hugues Hoppe – LoD methods)
- **http://www.geometrictools.com/**
  (Dave Eberly)