Nivoji podrobnosti

Level of Detail (LOD)

Problem s podrobnostmi

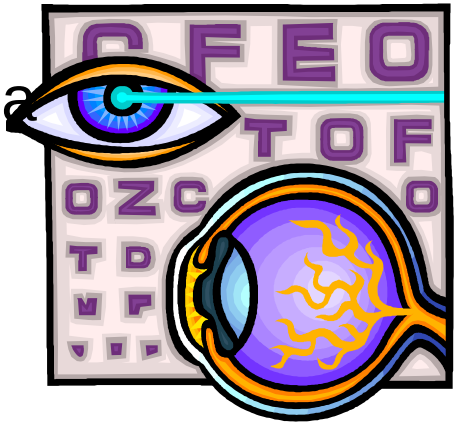
- Grafični sistemi so preplavljeni s podatki o modelih
  - Zelo podrobne podatkovne baze CAD
  - Zelo natančna skeniranja ploskev
- Viri, ki so na voljo, so omejeni
  - CPE, prostor, hitrost grafike, Pasovna širina omrežja
- Potrebujemo bolj ekonomične modele
  - Želimo najmanjši nivo podrobnosti (level of detail, LOD), ki še zadošča

LOD in interaktivnost

- Nivo podrobnosti (LOD) je pomembna tehnika za zagotavljanje interaktivnosti
  - Kompromis med vernostjo in učinkovitostjo
  - Ni edina tehnikal! Je komplement:
    - Vzporednem upodabljanju
    - Izločanju zakritih stvari
    - Upodabljanju na nivoju slike (image-based rendering)
    - itd...

Omejitve vida

- **Visual acuity**
  - Retina can resolve detail of around 0.5 min of arc
  - 130 million photoreceptors / 1 million ganglion cells
- **Peripheral Vision**
  - Highest sensitivity to spatial detail at fovea (the central 4 to 5 degrees of vision)
  - 35-fold reduction from fovea → periphery
- **Motion Sensitivity**
  - Eye less sensitive to detail moving across retina
  - Fast moving objects become “blurred”



Visual Perception Software

Vermeer

"Officer and Laughing Girl", 1658-60

120 x 135 degrees FOV

No eccentricity blurring

No velocity blurring

Visual Perception Software

Vermeer

“Officer and Laughing Girl”, 1658-60

120 x 135 degrees FOV

Eccentricity blurring

No velocity blurring

Visual Perception Software

Vermeer

“Officer and Laughing Girl”, 1658-60

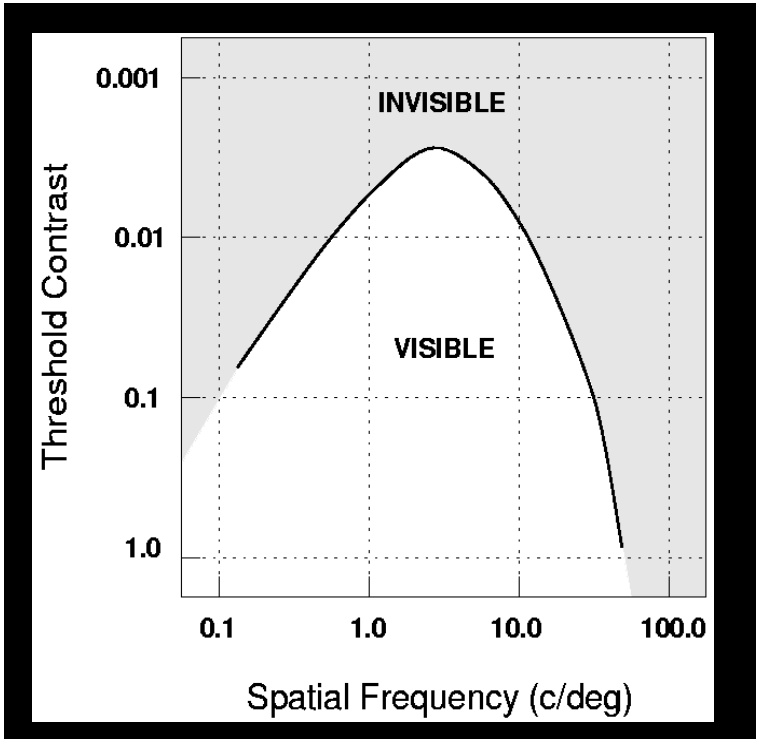
120 x 135 degrees FOV

Eccentricity blurring

Velocity = 60 deg/s

Modeliranje omejitev vida

- Results of Contrast Grating tests can be modeled with a Contrast Sensitivity Function
- CSF defines the bandwidth of vision



Faktorji CSF

- Background illumination
  - Contrast sensitivity degrades in dim conditions
- Display Device Settings
  - Brightness, contrast, color, and gamma
- Viewer's level of light adaption
  - Photoreceptor range and pupil dilation controlled by a feedback loop
- Viewer's visual system efficiency
  - e.g., myopia causes light to converge in front of retina
- Viewer's age
  - Contrast sensitivity less developed in infants & declines with old age

Faktorji CSF (nadaljevanje)

- Viewer's emotional state
  - Affects dilation of pupils: smaller pupil = less light = drop in visual acuity
- Auditory Stimuli?
  - Recent Nature paper shows visual perception affected by a adding an audible beep during task

Therefore, perceptual data are normally based upon a "Standard Observer", assuming ideal environmental and viewer conditions.

LOD: osnovna zamisel

- The problem:
  - Geometric datasets can be too complex to render at interactive rates
- One solution:
  - Simplify the polygonal geometry of small or distant objects
  - Known as *Level of Detail* or *LOD*
    - polygonal simplification, geometric simplification, mesh reduction, decimation, multiresolution modeling, …

Tvorba LOD za predmete

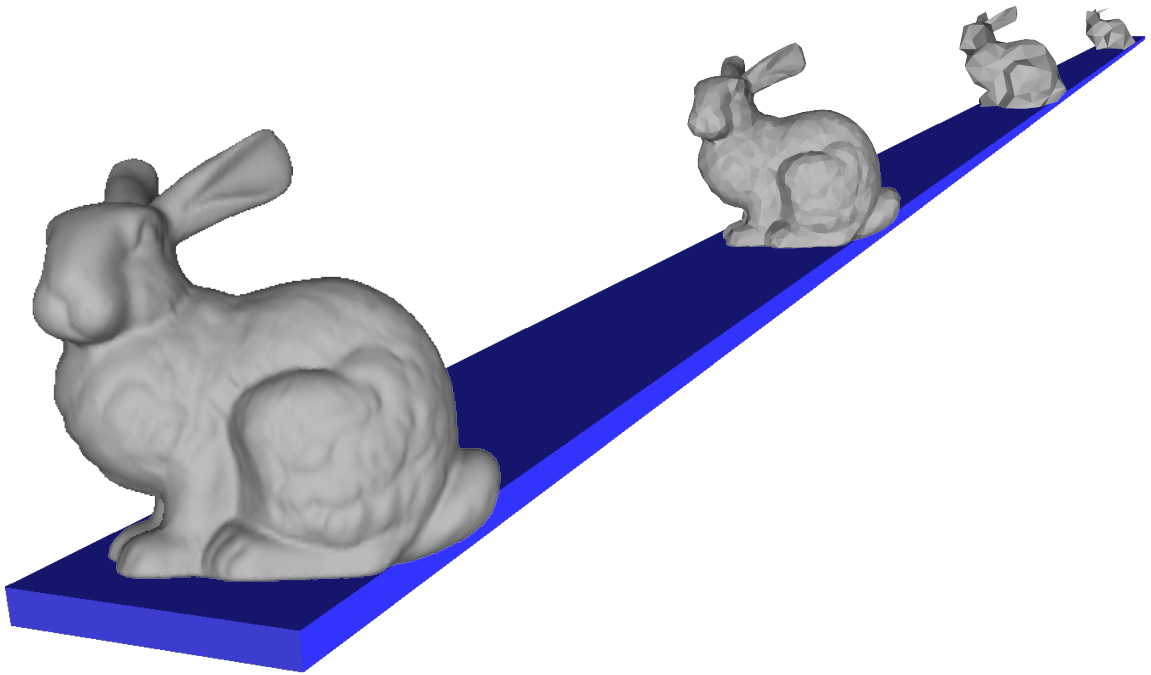
69,451 poligonov 2,502 poligonov 251 poligonov 76 poligonov

LOD: Vprašanja

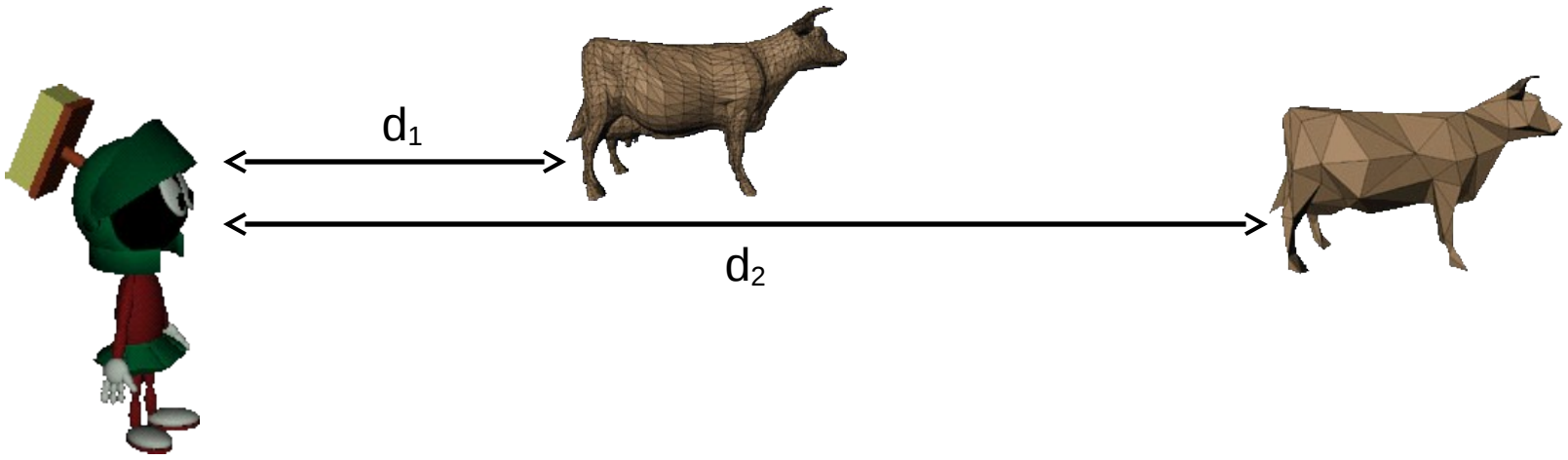
- *How to represent and generate simpler versions of a complex model?*
- *How to evaluate the fidelity of the simplified models?*
- *When to use which LOD of an object?*

69,451 poligonov 2,502 poligonov 251 poligonov 76 poligonov

Oddaljeni predmeti - bolj grob nivo podrobnosti



LOD in oddaljenost



Select resolution based upon the distance between an element and the viewpoint, i.e. coarser resolution for distant geometry.

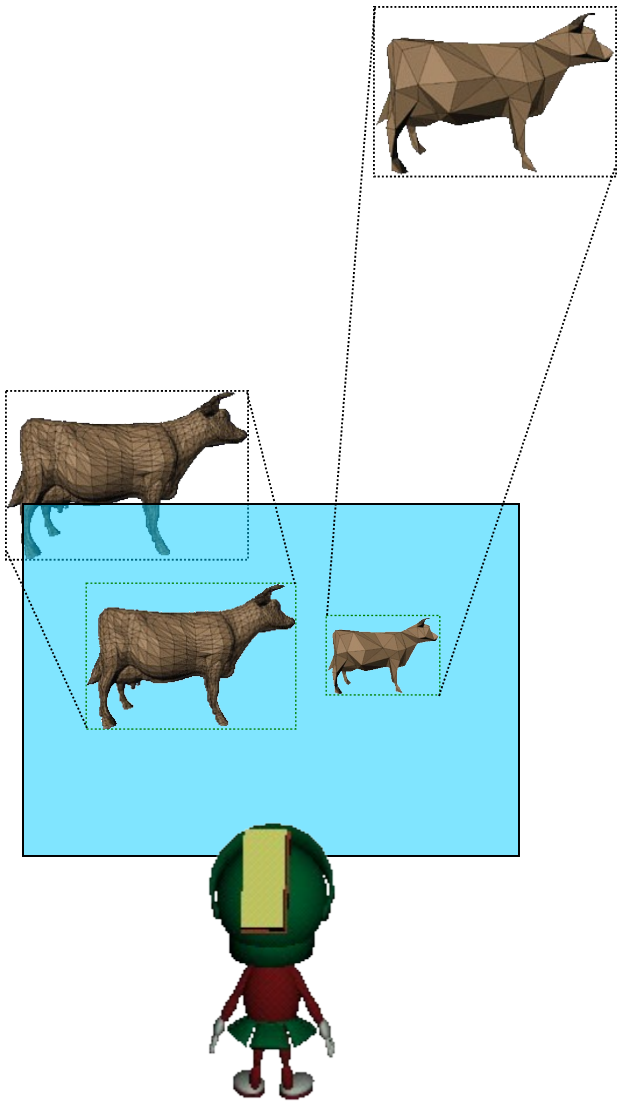
- Simple to calculate (3-D Euclidean distance)
- Scale dependent
- Resolution dependent
- Field of View dependent

Velikost in LOD

Select resolution based upon the projected screen size (or area) of an element. Objects appear smaller as they move further away.

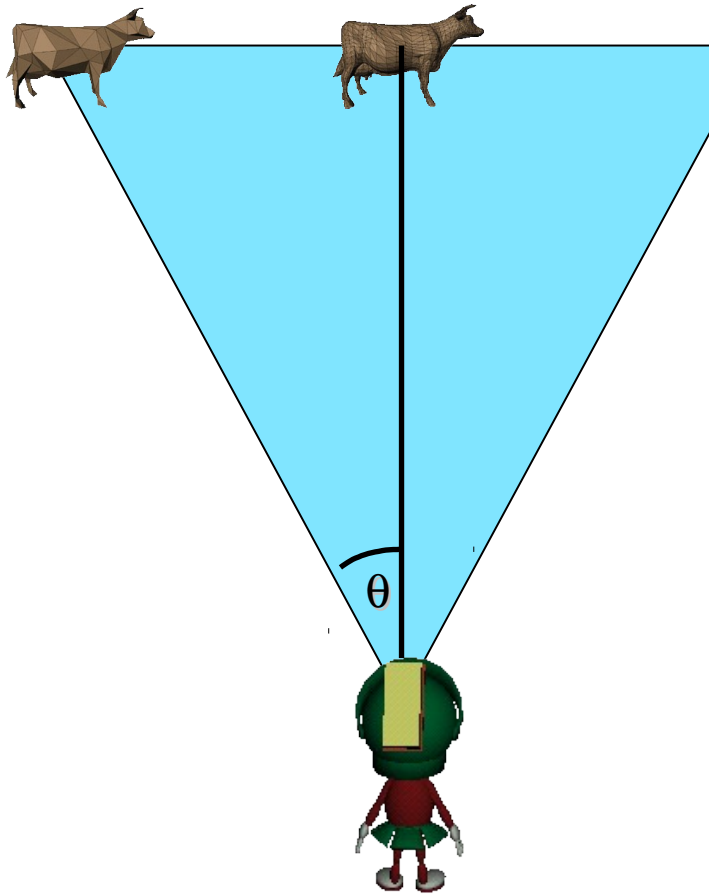
- Requires 3-D $\rightarrow$ 2-D projection
- Scale invariant
- Resolution invariant
- Field of View invariant

- Bounding spheres or ellipsoids normally used instead of boxes as more efficient to calculate projected extent



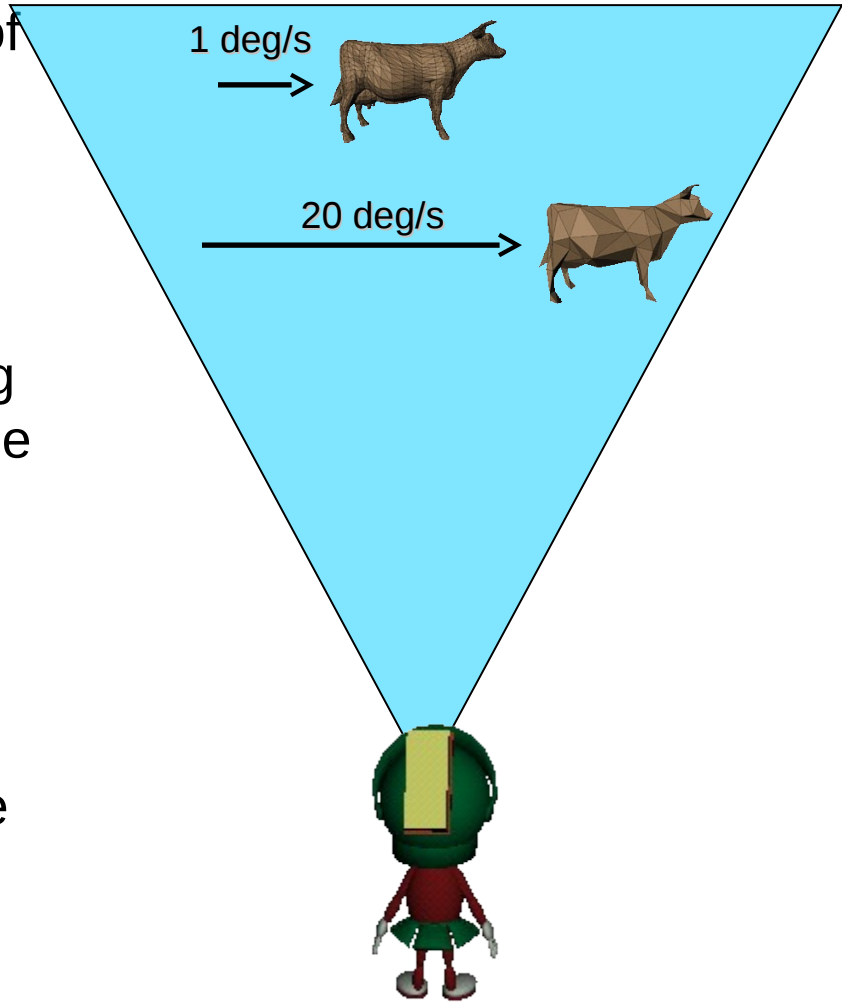
Ekscentričnost in LOD

- Resolution is selected based upon the degree to which an element exists in the visual periphery, i.e. display elements that the user is looking at in high resolution.
- Humans can resolve less detail in their peripheral field due to:
  - more retinal photoreceptors (rods/cones) towards fovea
  - retinal and cortical cell receptive field sizes increases linearly with eccentricity
  - 80% of cortical cells devoted to central 10 degrees of vision
- Use eye tracking system to track user's gaze or assume user looking towards center of display

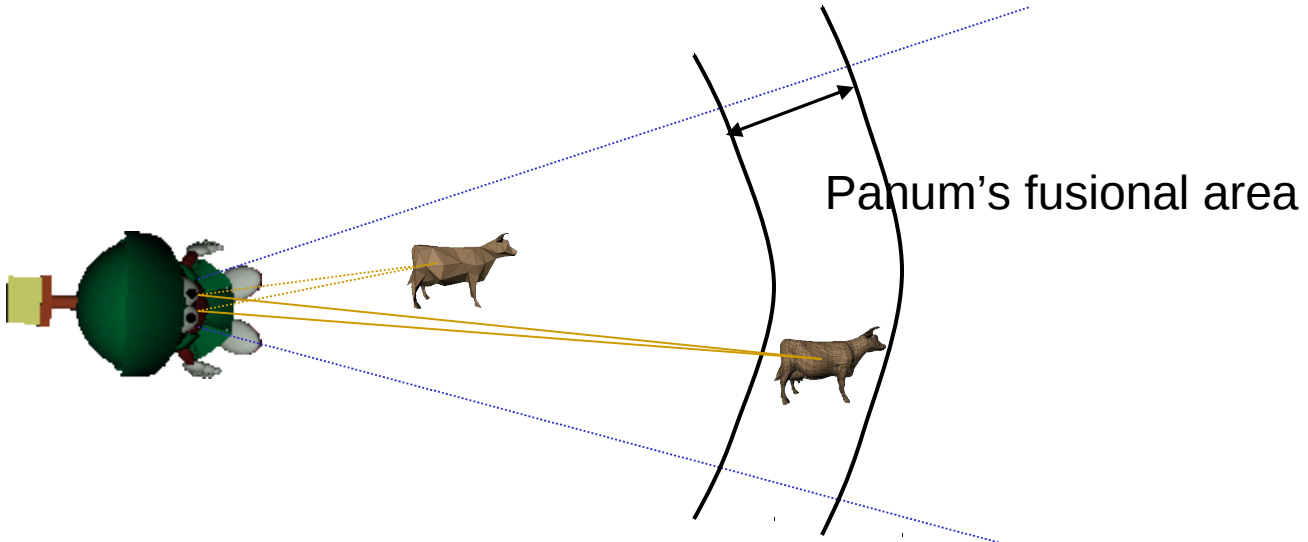


Hitrost in LOD

- olution based upon the angular velocity of an element across the visual field, i.e. faster moving objects appear in lower resolution
- Humans can resolve less spatial detail in objects moving across the retina, causing objects to blur as they move/ rotate, or the user's gaze moves
- It is believed visual information for small features are destroyed by the process of integrating stimulus energy over time
- Without eye tracking technology, assume angular velocity across display device



Globina polja in LOD

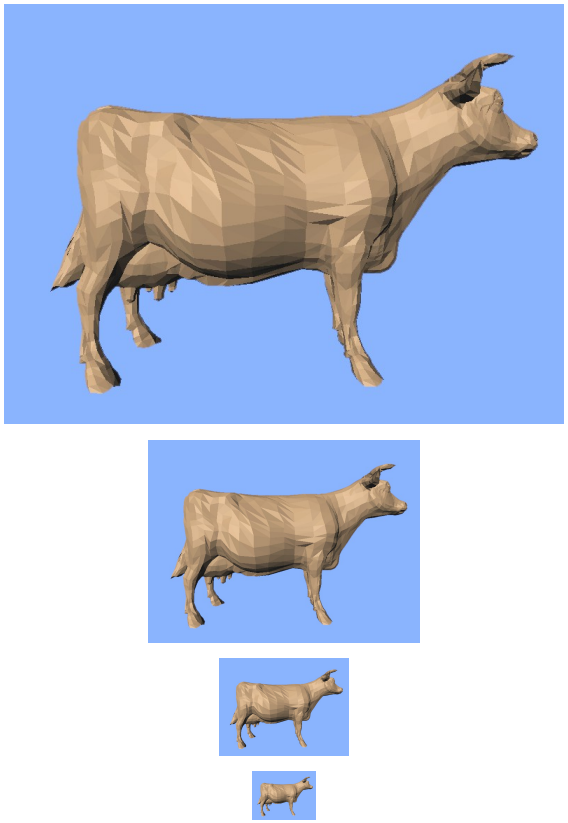


- Resolution of element dependent upon the depth of field focus of the user's eyes, i.e. objects out with the fusional area appear in lower detail
- Under binocular vision, both eyes converge on object at certain distance in order to focus retinal image
- Objects in front or behind this fusional area are unfocused, suffering from double images
- Must track both eyes accurately to evaluate convergence distance

Povzetek

- **Primary LOD selection criteria**
  - Distance or Size
  - Velocity
  - Eccentricity
  - Depth of Field
- **Additional LOD constraints**
  - Fixed-frame rate schedulers (reactive or predictive)
  - Hysteresis (switching lag)
  - Priority schemes
  - Alpha-blended transitions (fading regions)
  - Geomorph transitions (morph geometry)

Statična resolucija ne zadošča



- Model used in variety of contexts
  - many machines; variable capacity
  - projected screen size will vary
- Context dictates required detail
  - LOD should vary with context
  - context varies over time
  - with what level of coherence?
    - generally high coherence in view
    - possibly poor coherence in load

Potrebujemo večresolucijske modele (Multiresolution Models)

- Encode wide range of levels of detail
  - extract appropriate approximations at run time
  - must have low overhead
    - space consumed by representation
    - cost of changing level of detail while rendering
  - can be generated via simplification process
- Image pyramids (mip-maps) a good example
  - very successful technique for raster images

Zgradba LOD

- Discrete LOD
  - Generate a handful of LODs for each object
- Continuous LOD (CLOD)
  - Generate data structure for each object from which a spectrum of detail can be extracted
- View-dependent LOD
  - Generate data structure from which an LOD specialized to the current view parameters can be generated on the fly.
  - One object may span multiple levels of detail
- Hierarchical LOD
  - Aggregate objects into assemblies with their own LODs

Discrete Multiresolution Models

- Given a model, build a set of approximations
  - can be produced by any simplification system
  - at run time, simply select which to render
- Inter-frame switching causes “popping”
  - can smooth transition with image blending
  - or use geometry blending: geomorphing [Hoppe]
- Supported by several software packages

Tradicionalni pristop: Diskretni nivo podrobnosti

- 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, we call this *discrete LOD*

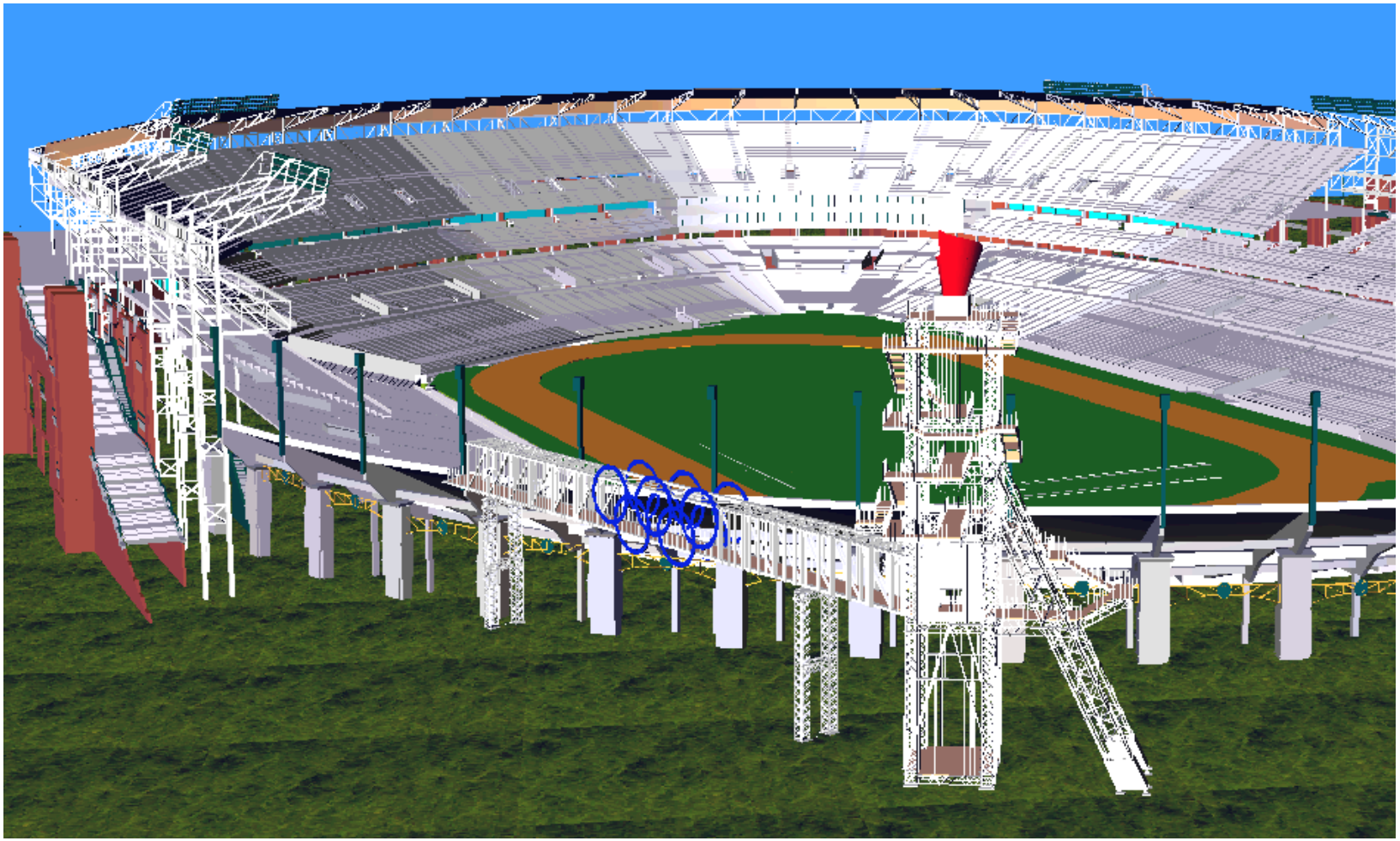
Diskretni LOD:Prednosti

- Simplest programming model; decouples simplification and rendering
  - LOD creation need not address real-time rendering constraints
  - Run-time rendering need only pick LODs
- Fits modern graphics hardware well
  - Easy to compile each LOD into triangle strips, display lists, vertex arrays, …
  - These render *much* faster than unorganized triangles on today's hardware (3-5 x)

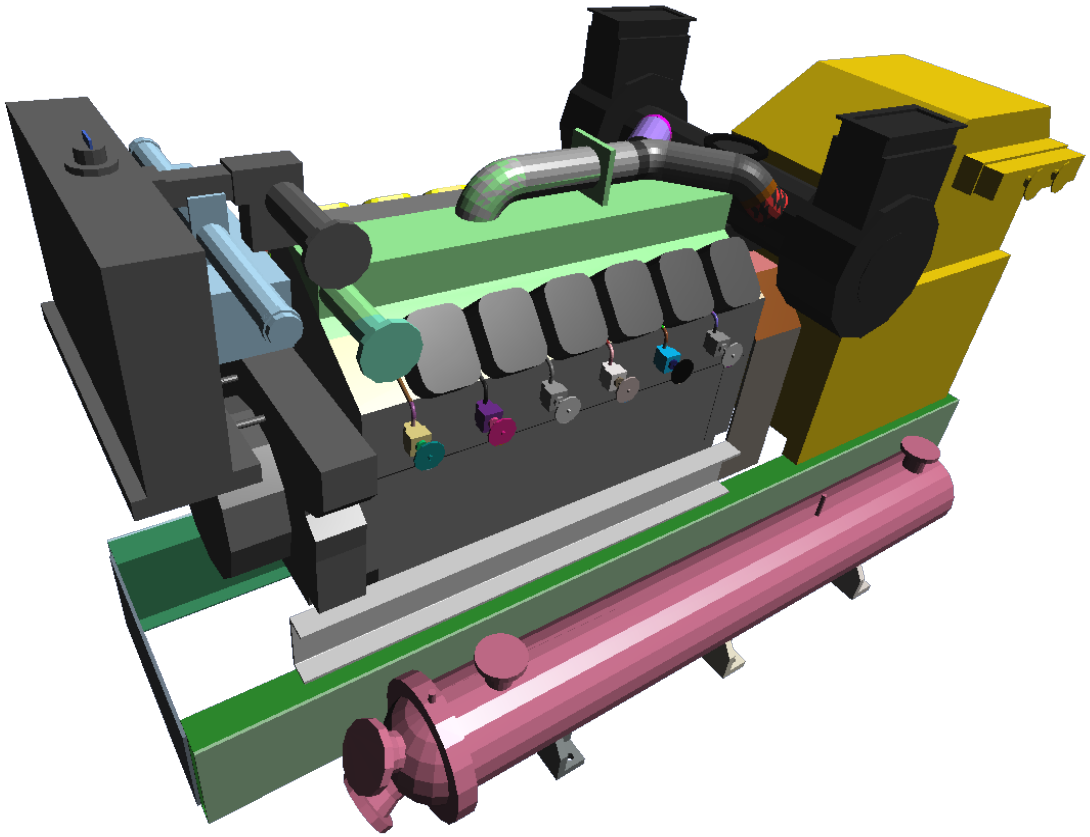
Diskretni LOD:Slabosti

- 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

Poenostavljanje: problem velikih objektov



Poenostavljanje: problem majhnih objektov



Limits of Discrete Models

- We may need varying LOD over surface
  - large surface, oblique view (eg. on terrain)
    - need high detail near the viewer
    - need less detail far away
  - single LOD will be inappropriate
    - either excessively detailed in the distance (wasteful)
    - or insufficiently detailed near viewer (visual artifacts)
- Doesn’t really exploit available coherence
  - small view change may cause large model change

Poenostavljanje

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

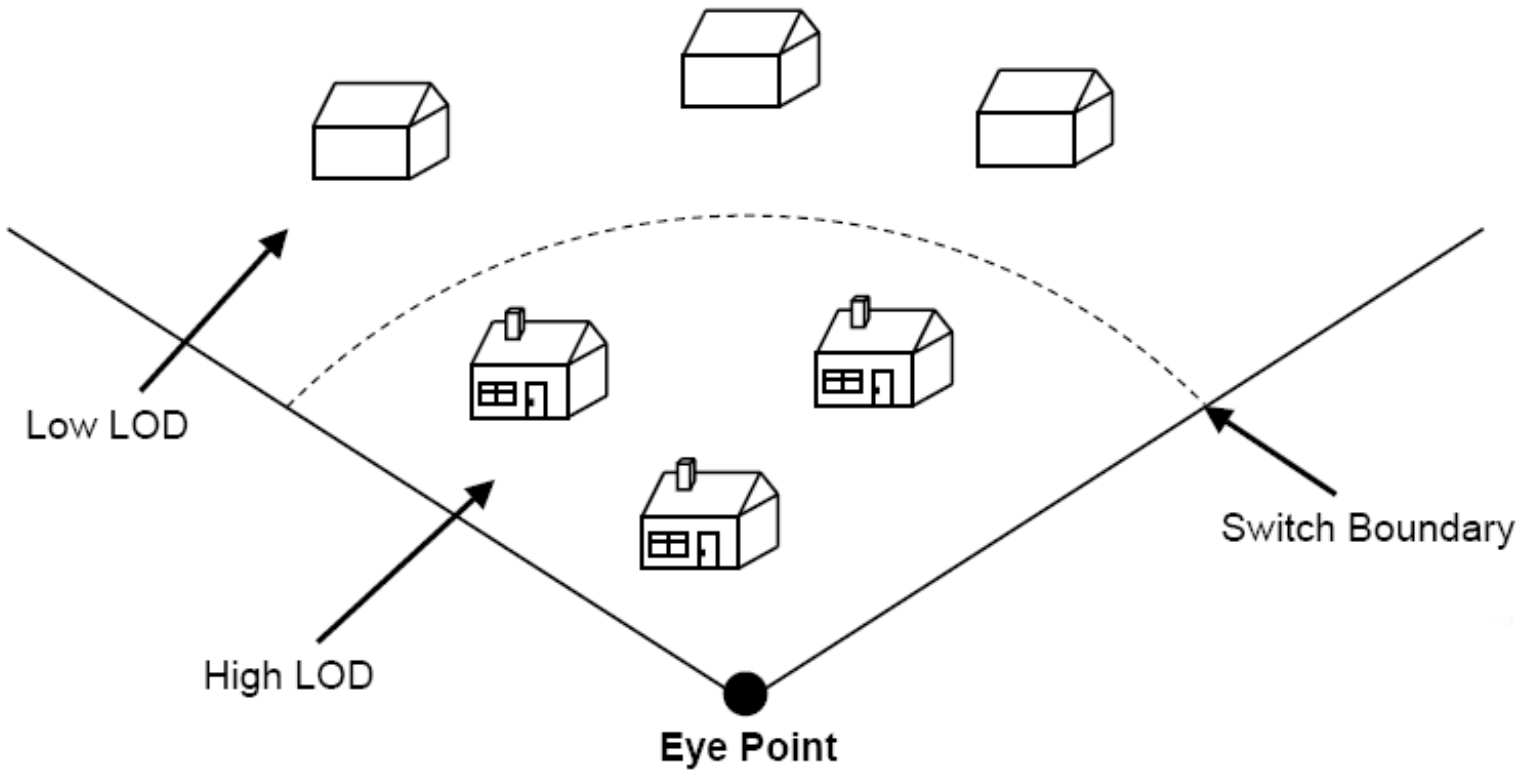
Choosing LODs: LOD Run-Time Management

- Fundamental LOD issue: where in the scene to allocate detail?
  - For discrete LOD this equates to choosing which LOD will represent each object
  - Run every frame on every object; keep it fast

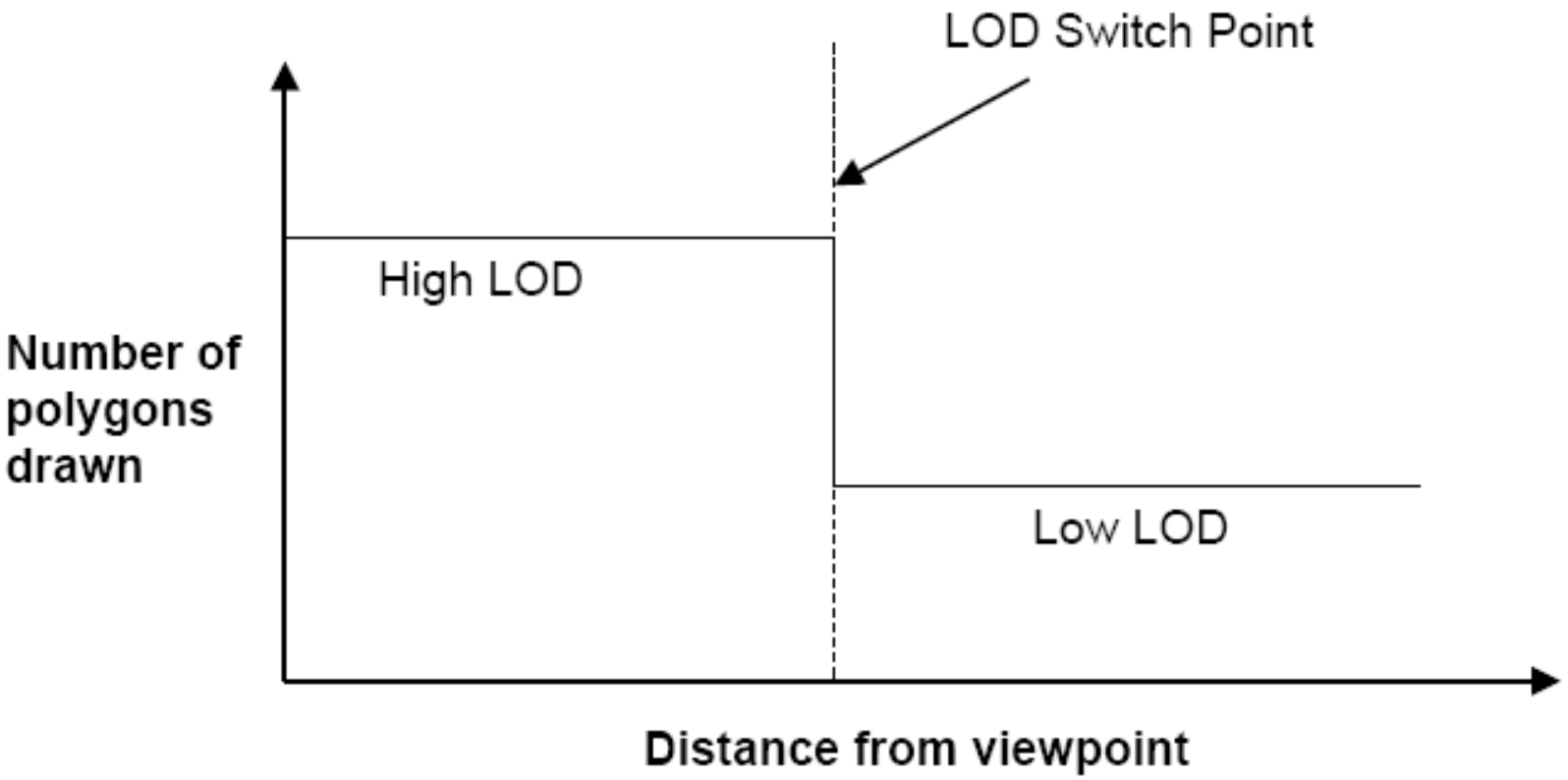
Choosing LODs

- *Describe a simple method for the system to choose LODs*
  - Assign each LOD a range of distances
  - Calculate distance from viewer to object
  - Use corresponding LOD
- *How might we implement this in a scene-graph based system?*

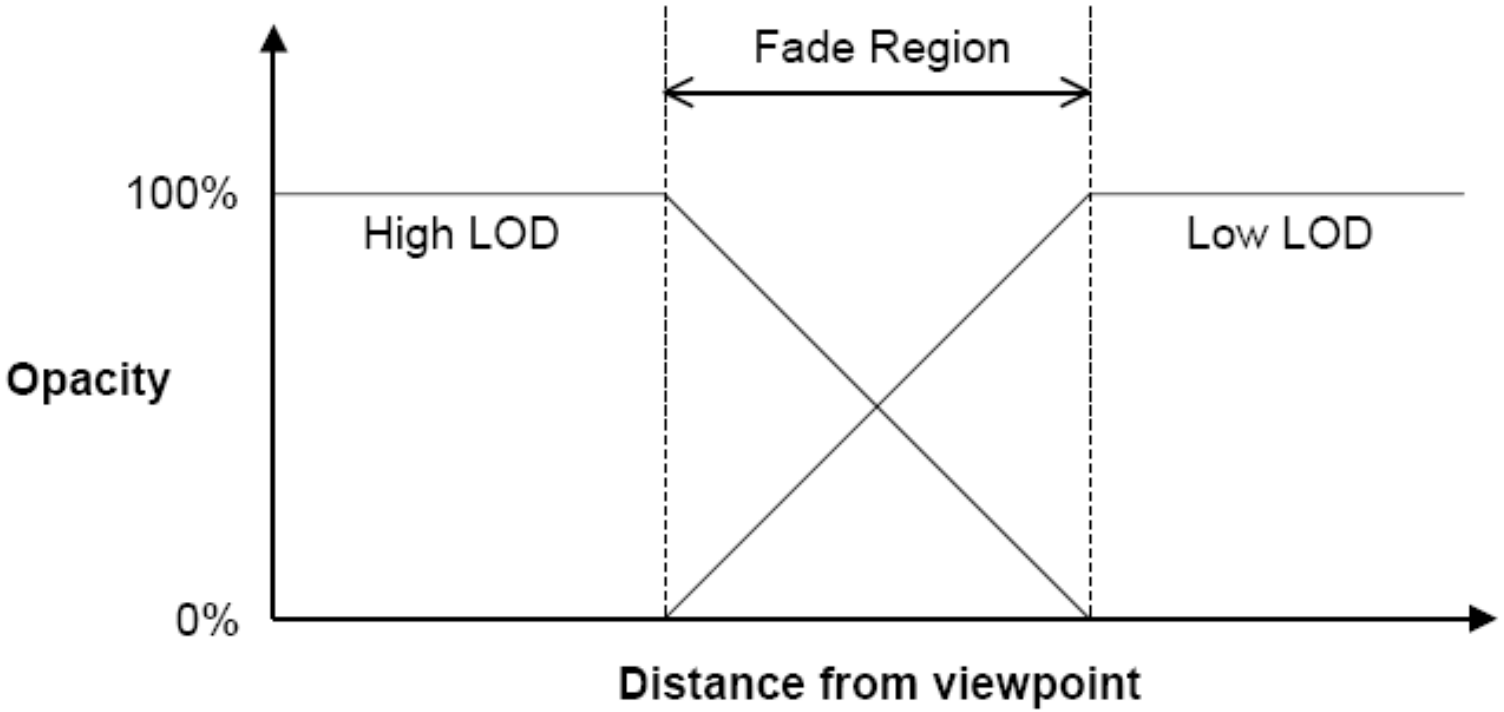
Implementacija preklapljanja LOD



Implementacija preklapljanja LOD

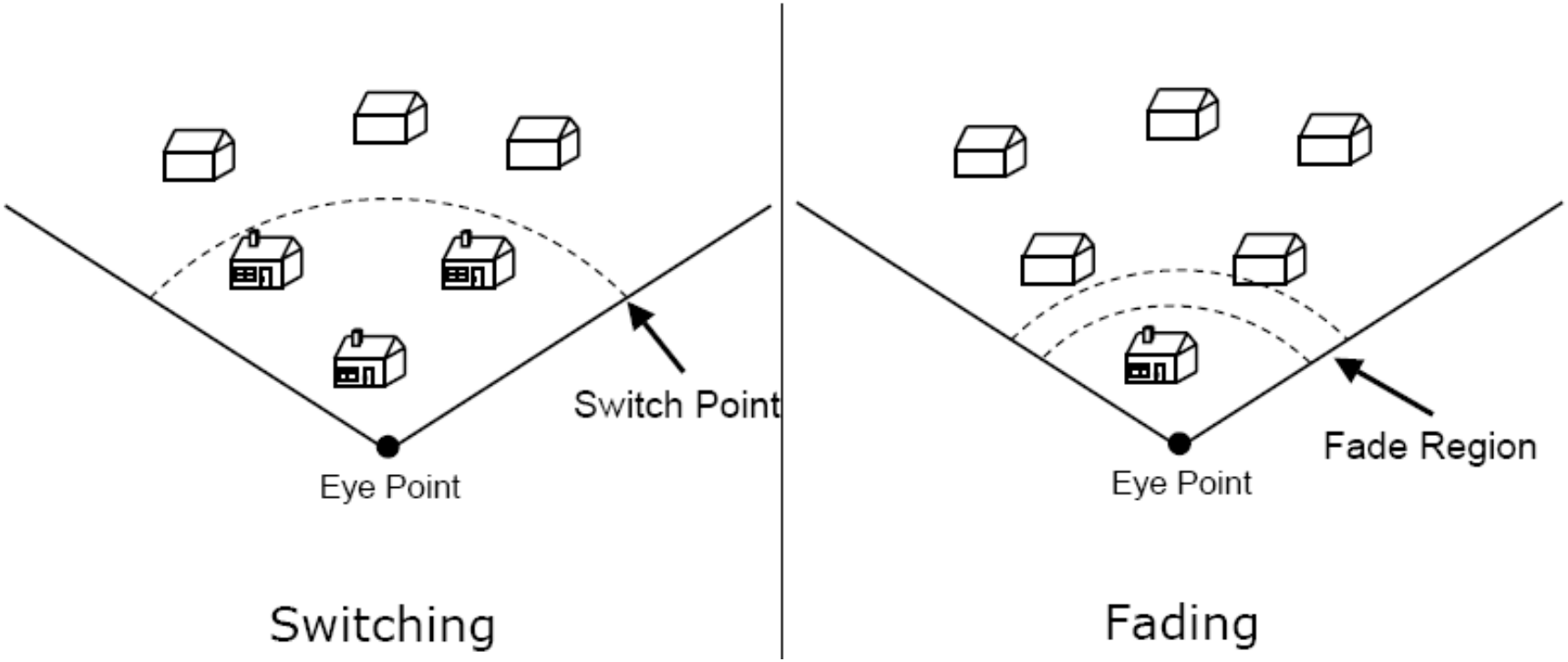


Implementacija mehkega prehoda LOD

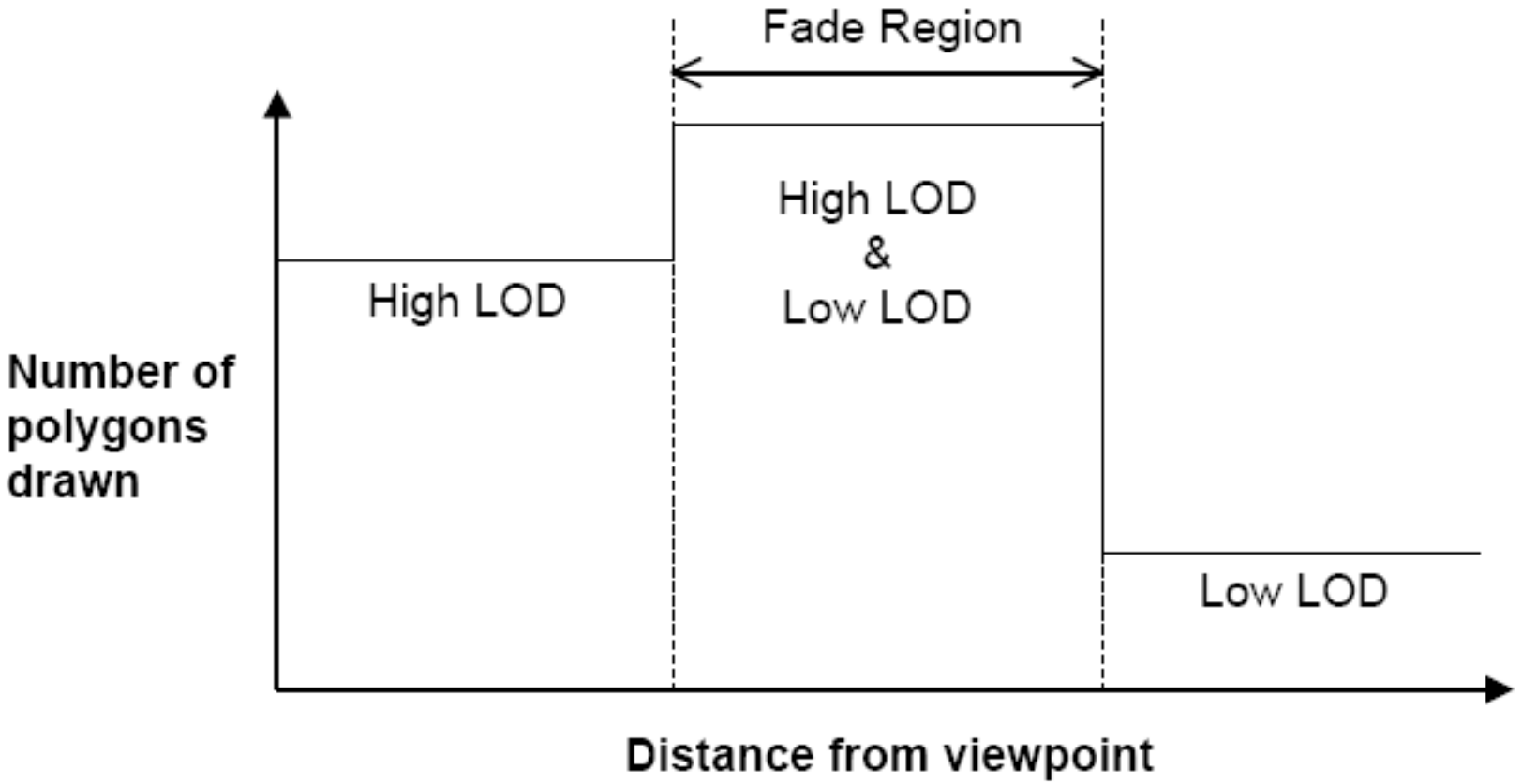


Primerjava preklapljanja in prehoda

- Transition Distances



Mehki prehod



Choosing LODs

- *What's wrong with this simple approach?*
  - Visual "pop" when switching LODs can be disconcerting
  - Doesn't maintain constant frame rate; lots of objects still means slow frame times
  - Requires someone to assign switching distances by hand
  - Correct switching distance may vary with field of view, resolution, etc.
- *What can we do about each of these?*

Choosing LODs Maintaining constant frame rate

- One solution: scale LOD switching distances by a “bias”
  - Implement a feedback mechanism:
    - If last frame took too long, decrease bias
    - If last frame took too little time, increase bias
  - Dangers:
    - Oscillation caused by overly aggressive feedback
    - Sudden change in rendering load can still cause overly long frame times

Choosing LODs: Maintaining constant frame rate

- A better (but harder) solution: predictive LOD selection
- For each LOD estimate:
  - *Cost* (rendering time)
  - *Benefit* (importance to the image)

Choosing LODs: Maintaining constant frame rate

- A better (but harder) solution: predictive LOD selection
- For each LOD estimate:
  - *Cost* (rendering time)
    - # of polygons
    - How large on screen
    - Vertex processing load (e.g., lighting) OR
    - Fragment processing load (e.g., texturing)
  - *Benefit* (importance to the image)

Choosing LODs: Maintaining constant frame rate

- A better (but harder) solution: predictive LOD selection
- For each LOD estimate:
  - *Cost* (rendering time)
  - *Benefit* (importance to the image)
    - Size: larger objects contribute more to image
    - Accuracy: no of verts/polys, shading model, etc.
    - Priority: account for inherent importance
    - Eccentricity: peripheral objects harder to see
    - Velocity: fast-moving objects harder to see
    - Hysteresis: avoid flicker; use previous frame state

Zvezni nivo podrobnosti

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

Zvezni LOD:prednosti

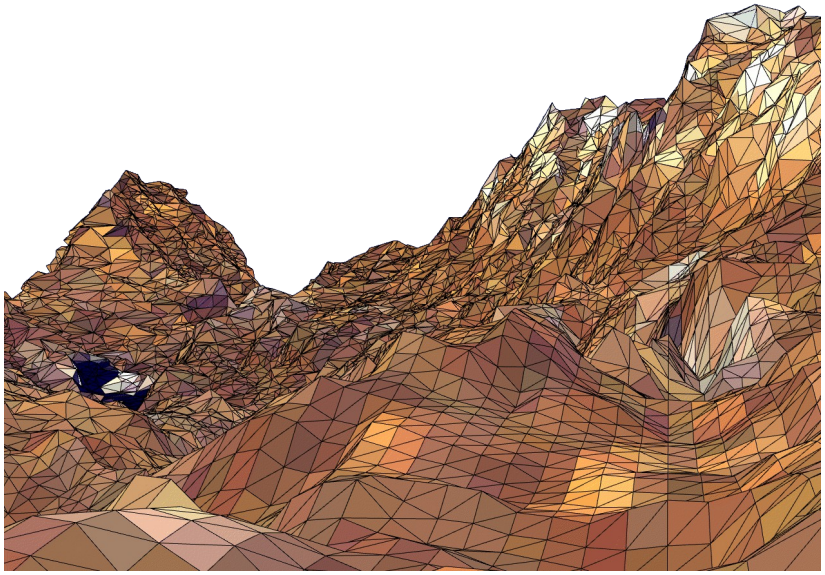
- Better granularity → better fidelity
  - LOD is specified exactly, not chosen from a few pre-created options
  - Thus objects use no more polygons than necessary, which frees up polygons for other objects
  - Net result: better resource utilization, leading to better overall fidelity/polygon
- Better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

Zvezni LOD:prednosti

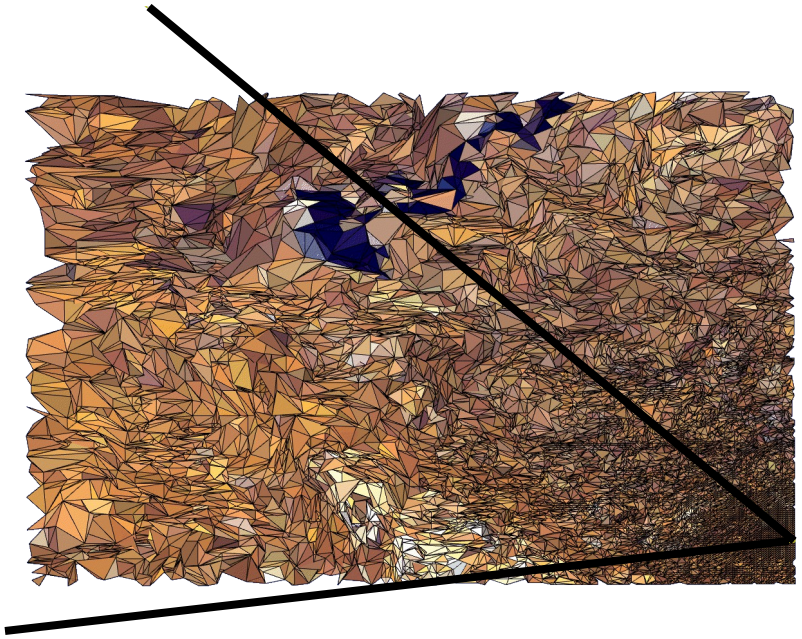
- Supports progressive transmission
  - *Progressive Meshes [Hoppe 97]*
  - *Progressive Forest Split Compression [Taubin 98]*
- Leads to *view-dependent LOD*
  - Use current view parameters to select best representation *for the current view*
  - Single objects may thus span several levels of detail

View-Dependent LOD: Primeri

- Show nearby portions of object at higher resolution than distant portions



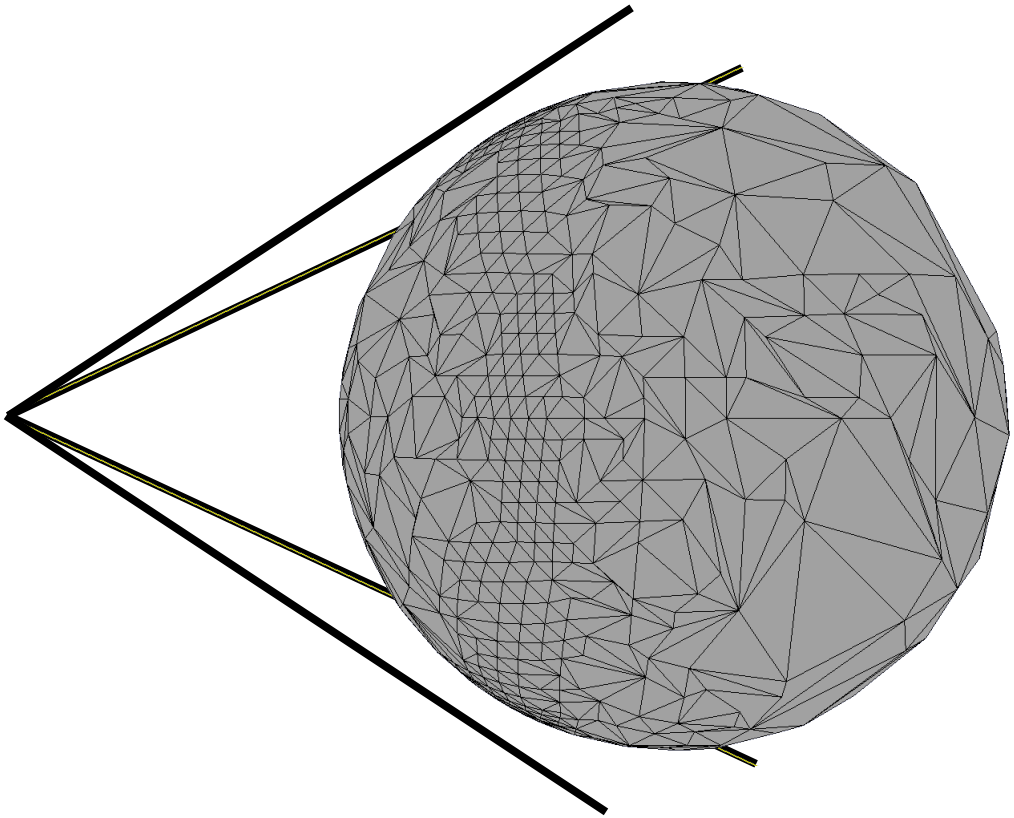
View from eyepoint



Birds-eye view

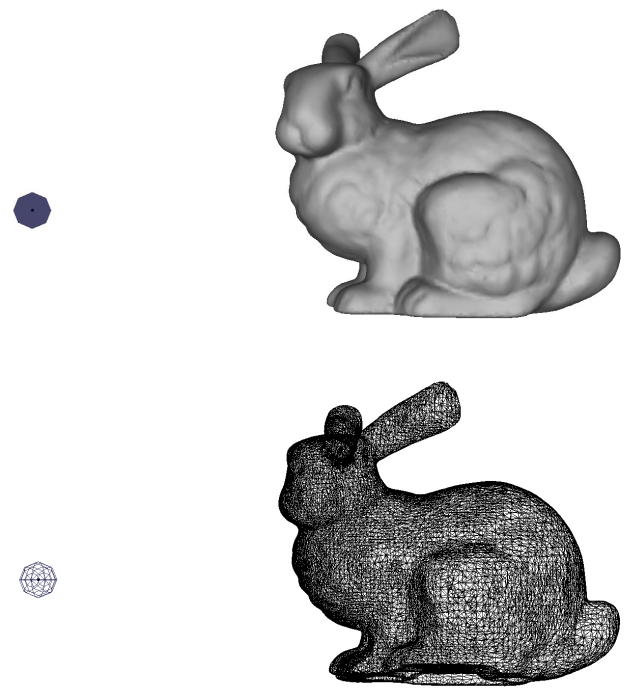
View-Dependent LOD: Primeri

- Show silhouette regions of object at higher resolution than interior regions



View-Dependent LOD:Primeri

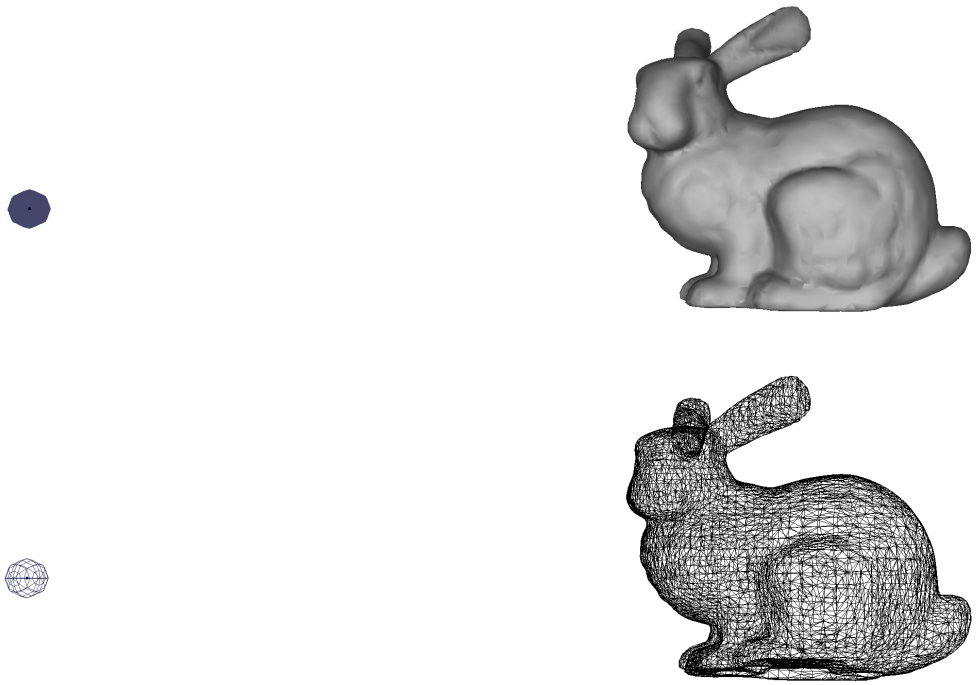
- Show more detail where the user is looking than in their peripheral vision:



34,321 trikotnikov

View-Dependent LOD:primeri

- Show more detail where the user is looking than in their peripheral vision:



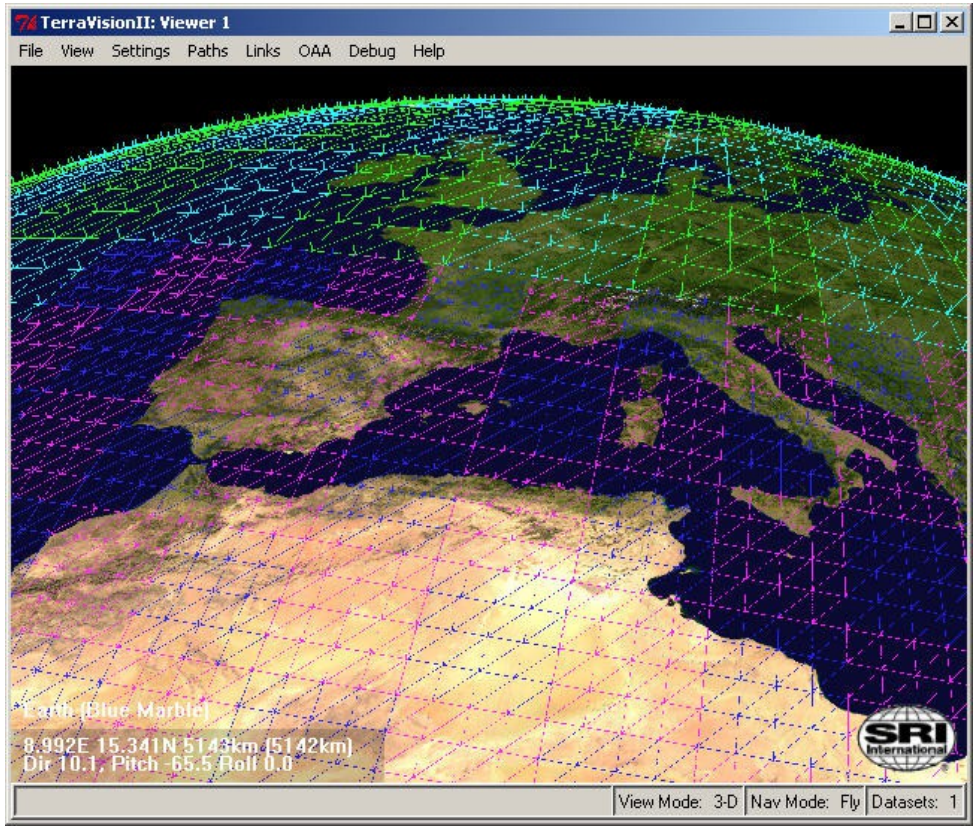
11,726 trikotnikov

View-Dependent LOD:Prednosti

- Even better granularity
  - Allocates polygons where they are most needed, within as well as among objects
  - Enables even better overall fidelity
- Enables drastic simplification of very large objects
  - Example: stadium model
  - Example: terrain flyover

Streaming over the Web

- TerraVision (SRI)



Yosemite Park

San Francisco Bay

Hierarhični LOD

- View-dependent LOD solves the Problem With Large Objects
- *Hierarchical LOD* can solve the Problem With Small Objects
  - Merge objects into assemblies
  - At sufficient distances, simplify assemblies, not individual objects
  - *How to represent this in a scene graph?*

Hierarhični LOD

- Hierarchical LOD dovetails nicely with view-dependent LOD
  - Treat the *entire scene* as a single object to be simplified in view-dependent fashion
- Hierarchical LOD can also sit atop traditional discrete LOD schemes

Izbira LODs: Upravljanje LOD v realnem času

- Fundamental LOD issue: where in the scene to allocate detail?
  - For discrete LOD this equates to choosing which LOD will represent each object
  - Run every frame on every object; keep it fast

Izbira LOD

- *Describe a simple method for the system to choose LODs*
  - Assign each LOD a range of distances
  - Calculate distance from viewer to object
  - Use corresponding LOD
- *How might we implement this in a scene-graph based system?*

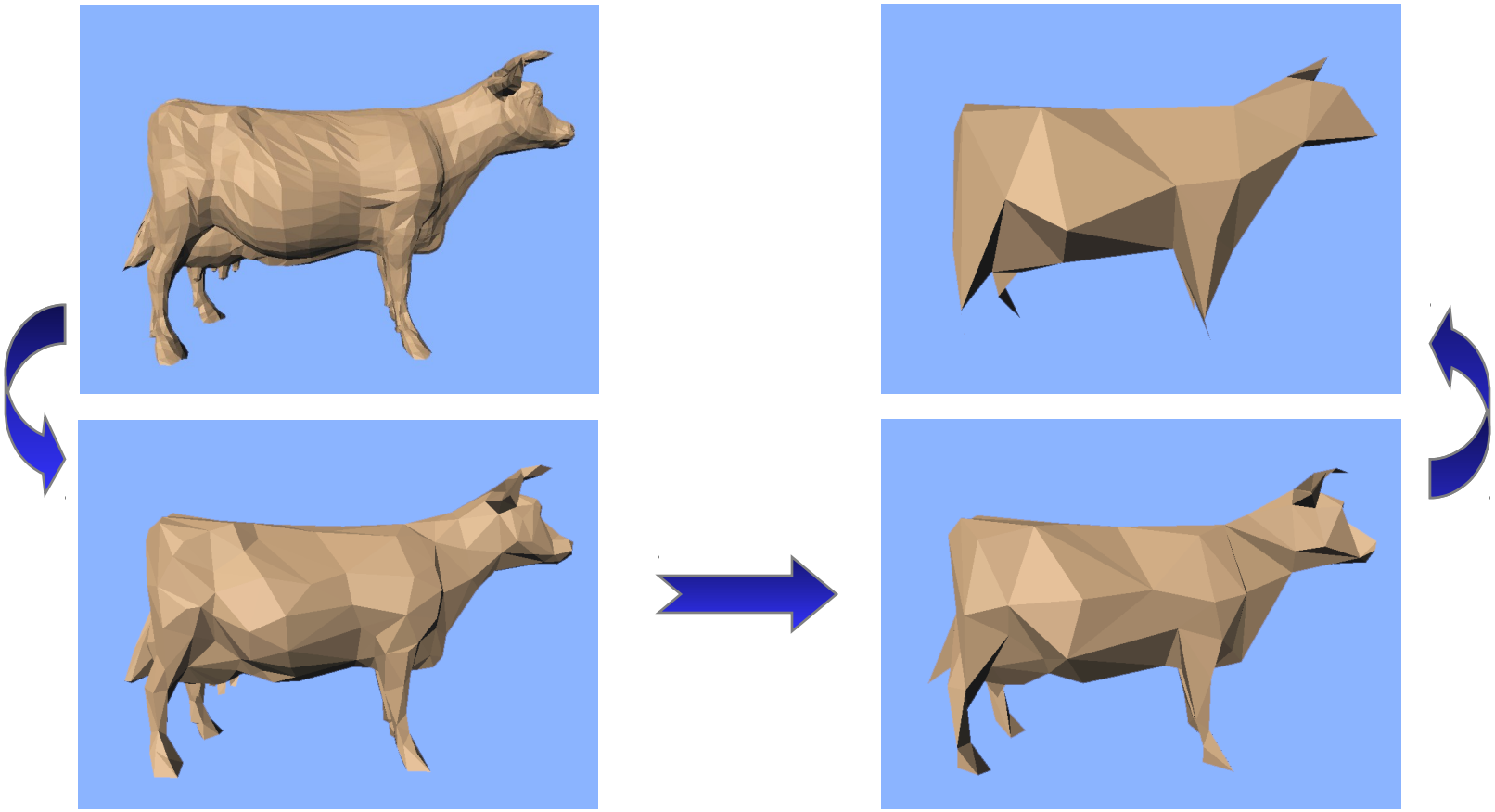
Izbira LOD

- *What's wrong with this simple approach?*
  - Visual "pop" when switching LODs can be disconcerting
  - Doesn't maintain constant frame rate; lots of objects still means slow frame times
  - Requires someone to assign switching distances by hand
  - Correct switching distance may vary with field of view, resolution, etc.

Izbira LOD: vzdrževanje konstantnega števila slik (frame rate)

- One solution: scale LOD switching distances by a “bias”
  - Implement a feedback mechanism:
    - If last frame took too long, decrease bias
    - If last frame took too little time, increase bias
  - Dangers:
    - Oscillation caused by overly aggressive feedback
    - Sudden change in rendering load can still cause overly long frame times

Avtomatska poenostavitev ploskev



Avtomatska poenostavitev ploskev

- Produce approximations with fewer triangles
  - should be as similar as possible to original
  - want computationally efficient process
- Need criteria for assessing model similarity
  - for display, visual similarity is the ultimate goal
  - similarity of shape is often used instead
    - generally easier to compute
    - lends itself more to applications other than display

Fokus na poligonskih modelih

- Polygonal surfaces are ubiquitous
  - only primitive widely supported in hardware
  - near-universal support in software packages
  - output of most scanning systems
- Switching representations is no solution
  - indeed, some suffer from the same problem
  - many applications *want* polygons
- Will always assume models are triangulated

Druga področja

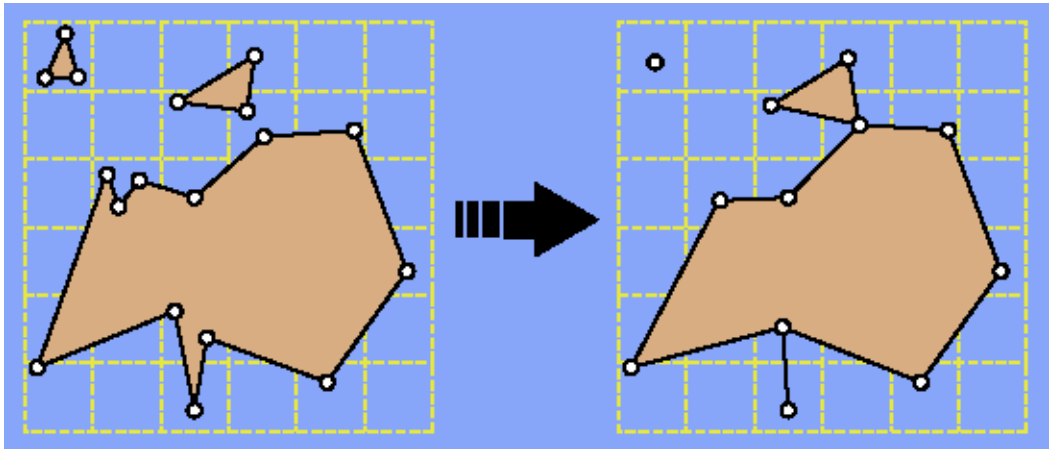
- Geometry compression
  - simplification is a kind of lossy compression
- Surface smoothing
  - reduces geometric complexity of shape
- Mesh generation
  - finite element analysis (e.g., solving PDE’s)
  - need appropriate mesh for good solution
  - overly complex mesh makes solution slow

Pregled metod poenostavljanja

- Manual preparation has been widely used
  - skilled humans produce excellent results
  - very labor intensive, and thus costly
- Most common kinds of automatic methods
  - vertex clustering
  - vertex decimation
  - iterative contraction

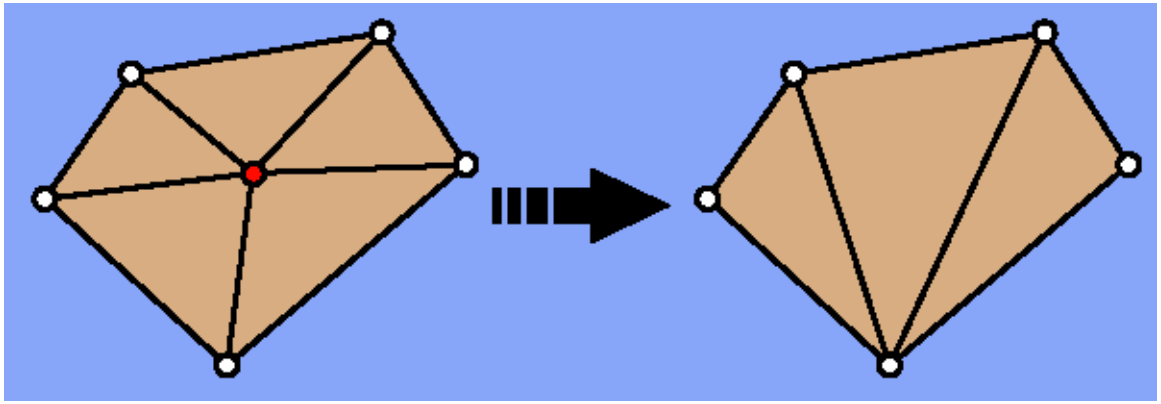
Združevanje verteksov

- Partition space into cells
  - grids [Rossignac-Borrel], spheres [Low-Tan], octrees, ...
- Merge all vertices within the same cell
  - triangles with multiple corners in one cell will degenerate



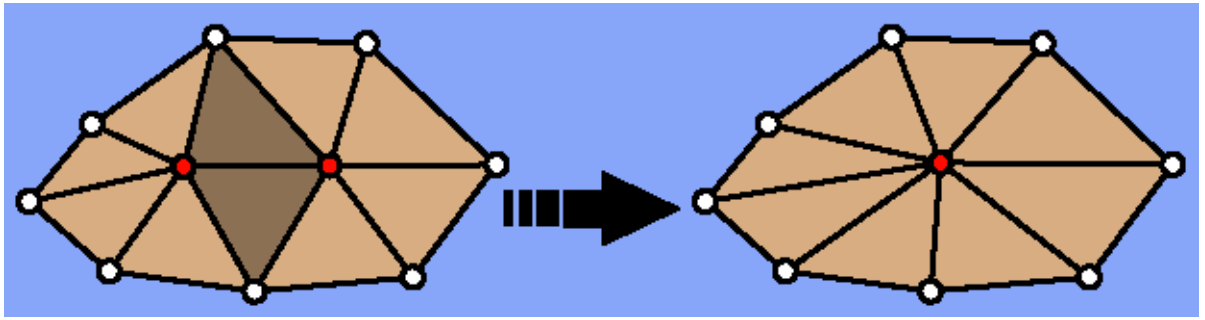
Zmanjševanje števila verteksov

- Starting with original model, iteratively
  - rank vertices according to their importance
  - select unimportant vertex, remove it, retriangulate hole
- A fairly common technique



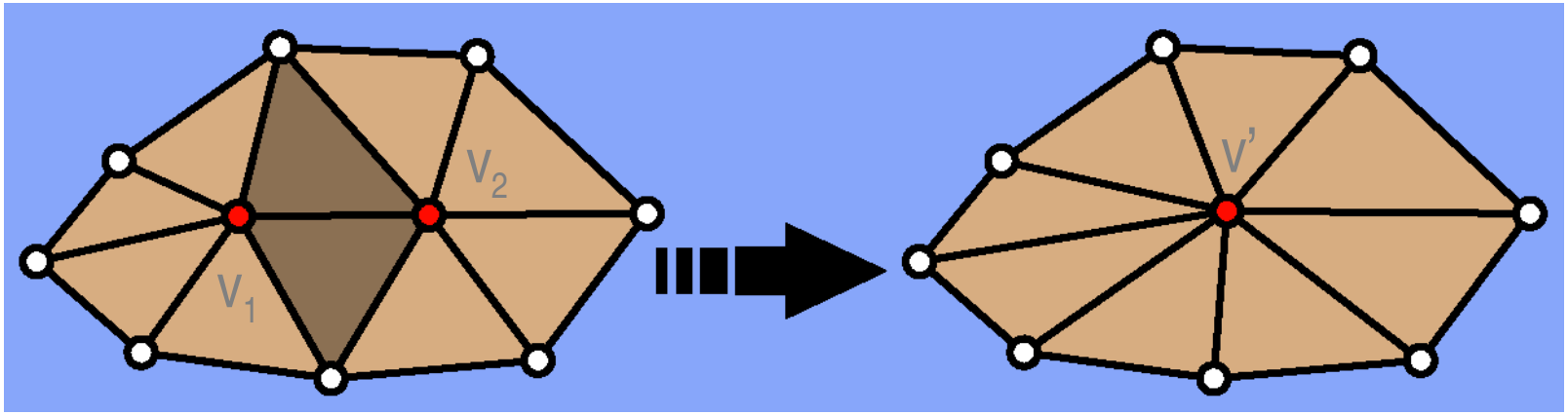
Iterativno krčenje robov

- Contraction can operate on any set of vertices
  - edges (or vertex pairs) are most common, faces also used
- Starting with the original model, iteratively
  - rank all edges with some cost metric
  - contract minimum cost edge
  - update edge costs

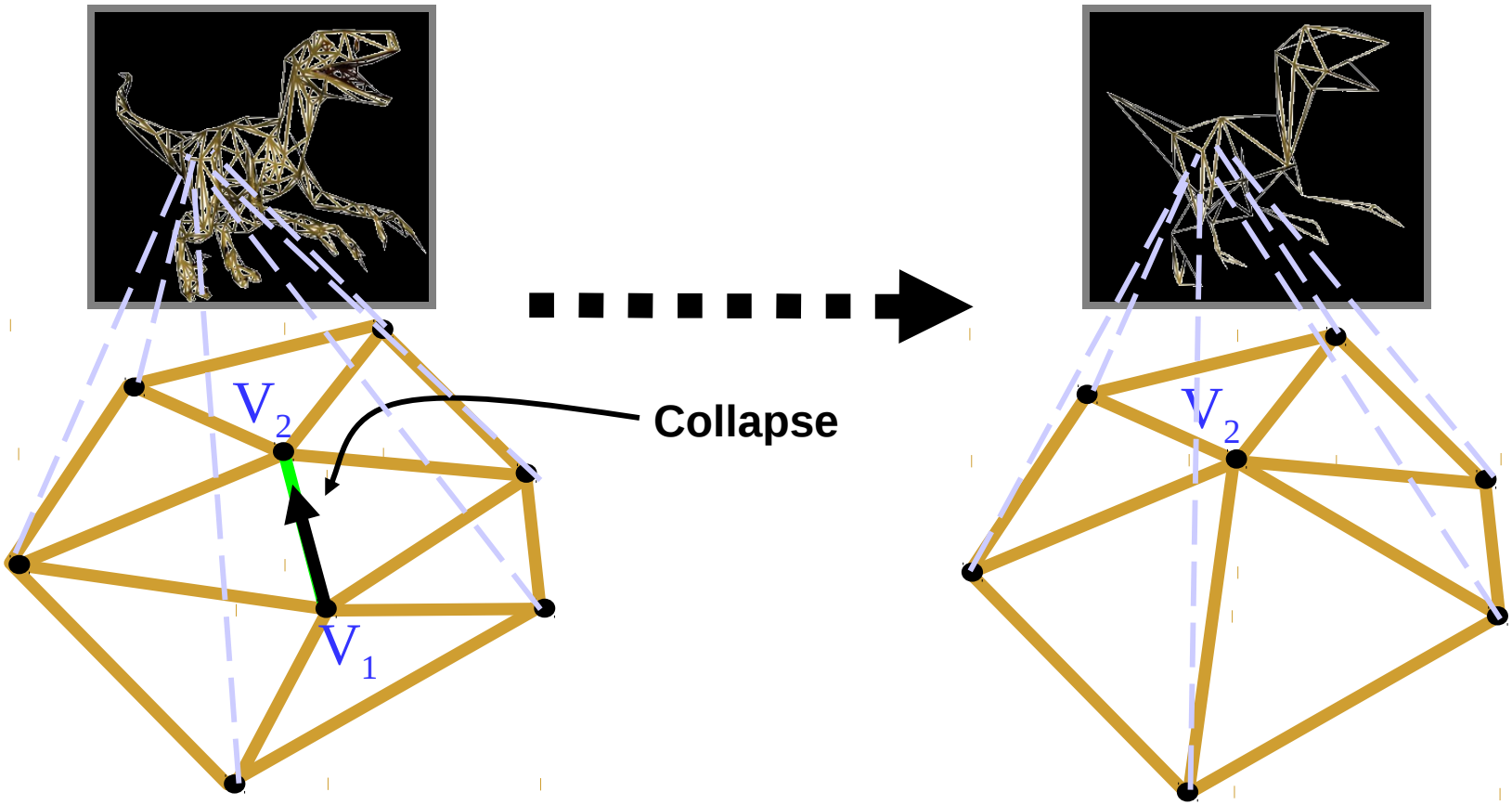


Krčenje robov

- Single edge contraction $(v_1, v_2) \rightarrow v'$ is performed by
  - moving v_1 and v_2 to position v'
  - replacing all occurrences of v_2 with v_1
  - removing v_2 and all degenerate triangles



Algoritem rušenja robov



Algoritem rušenja robov

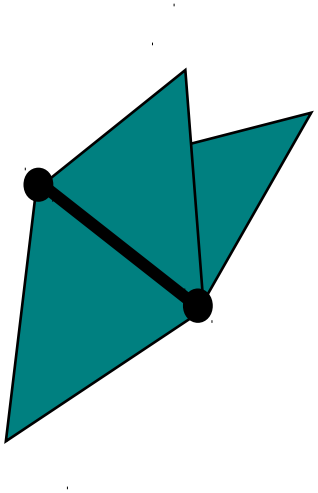
```
Sort all edges (by some metric)
repeat
   Collapse edge
   choose edge vertex (or compute optimal
     vertex)
   Fix-up topology
until (no edges left)
```

Iterativno krčenje robov

- Currently the most popular technique
  - simpler operation than vertex removal
  - well-defined on any simplicial complex
- Also induces hierarchy on the surface
  - a very important by-product
  - enables several multiresolution applications

Prednosti rušenja robov

- Edge collapse operation is simple
- Supports non-manifold topology:

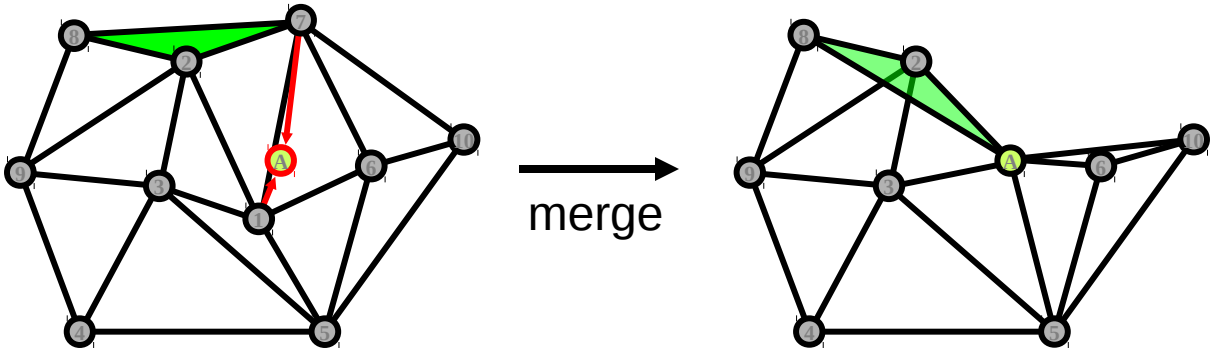


Ohranjevanje mej

- To preserve important boundaries, label edges as normal or *discontinuity*
- For each face with a discontinuity, a plane perpendicular intersecting the discontinuous edge is formed.
- These planes are then converted into quadrics, and can be weighted more heavily with respect to error value.

Preprečevanje inverzije mreže

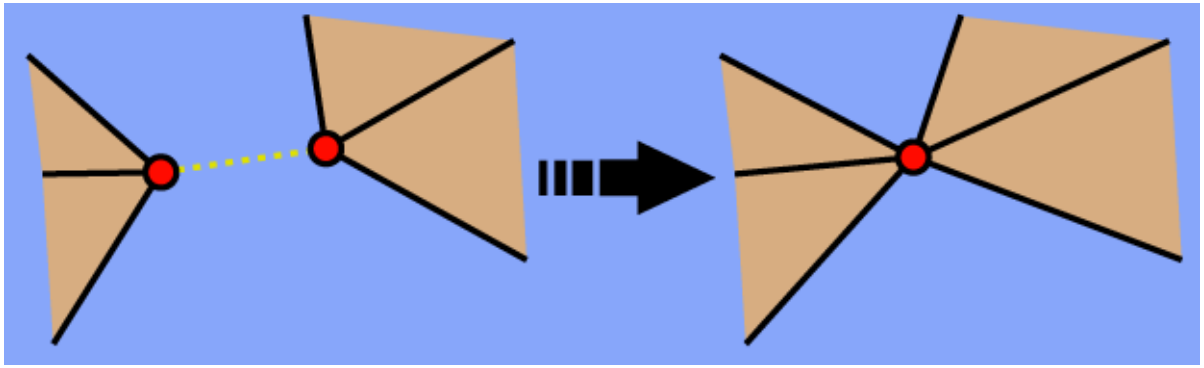
- Preventing foldovers:



- Calculate the adjacent face normals, then test if they would flip after simplification
- If so, that simplification can be weighted heavier or disallowed.

Krčenje parov verteksov

- Can also easily contract any pair of vertices
  - fundamental operation is exactly the same
  - joins previously unconnected areas
  - can be used to achieve topological simplification



Rušenje robov : združevanje parov verteksov

- Even better: *vertex-pair merging* merges two vertices that:
  - Share an edge, or
  - Are within some threshold distance *t*

View-Dependent LOD: Algoritmi

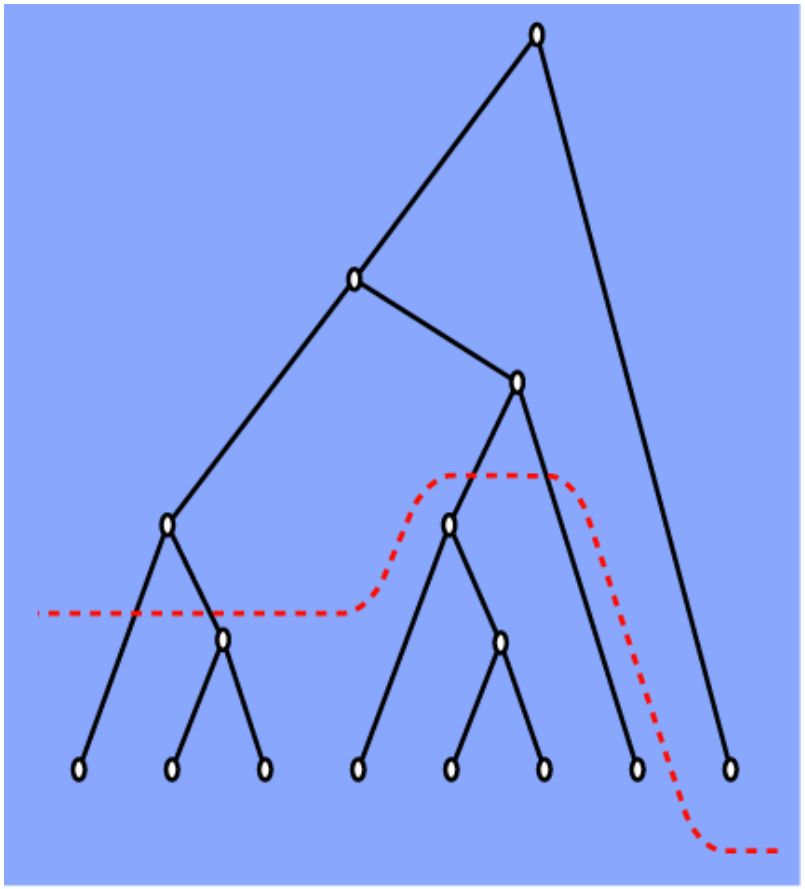
- Many good published algorithms:
  - *Progressive Meshes* by Hoppe
    Merge Trees by Xia & Varshney [Visualization 96]
  - *Hierarchical Dynamic Simplification* by Luebke & Erikson
    Multitriangulation by DeFloriani et al
  - Others…

Pregled: Algoritem VDS

- Overview of the VDS algorithm:
  - A preprocess builds the *vertex hierarchy*, a hierarchical clustering of vertices
  - At run time, clusters appear to grow and shrink as the viewpoint moves
  - Clusters that become too small are collapsed, filtering out some triangles

Vertex Hierarchies

- A cut through the tree
  - contract all below cut
  - leaves are “active”
  - determines partition
  - and an approximation
- Encodes dependencies
  - PM’s assume total order
  - disjoint subtrees indep.
  - get novel approximations
  - but must avoid fold-over



Podatkovne strukture

- The *vertex hierarchy*
  - Represents the entire model
  - Hierarchy of *all* vertices in model
  - Queried each frame for updated scene
- The *active triangle list*
  - Represents the current simplification
  - List of triangles to be displayed
  - Triangles added and deleted by operations on vertex tree

Vertex Hierarchies for View-Dependent Refinement

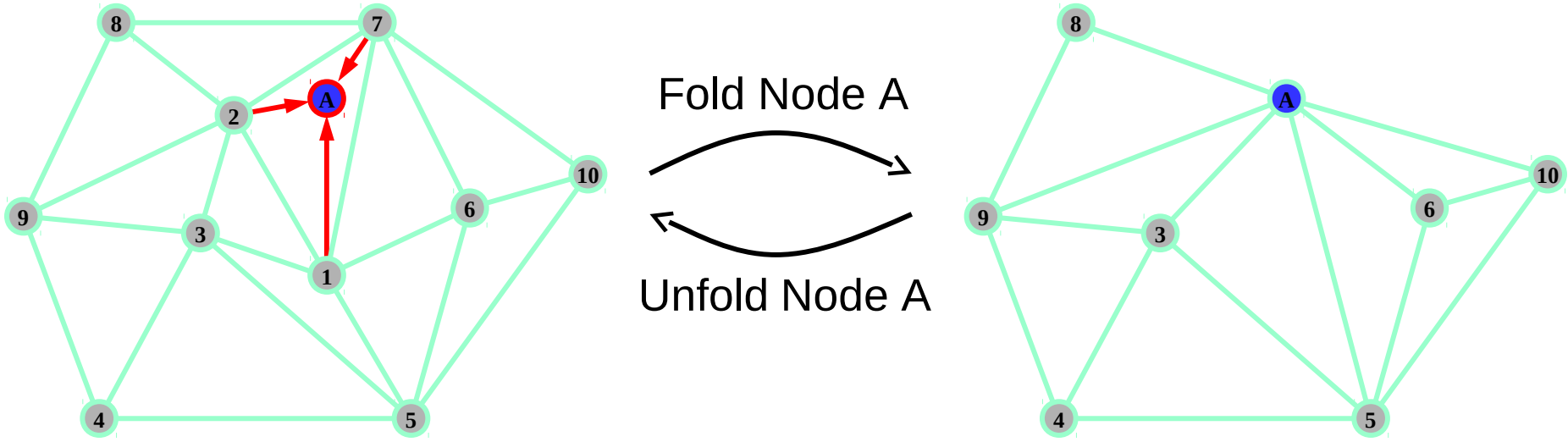
- Multiresolution representation for display
  - incrementally move cut between frames
    [Xia-Varshney, Hoppe, Luebke-Erickson]
  - move up/down where less/more detail needed
  - relies on frame-to-frame coherence
  - can accommodate geomorphing
- Common application of vertex hierarchy
  - hierarchy only guides active front evolution
  - more flexibility & overhead vs. discrete multires

Hierarhija verteksov

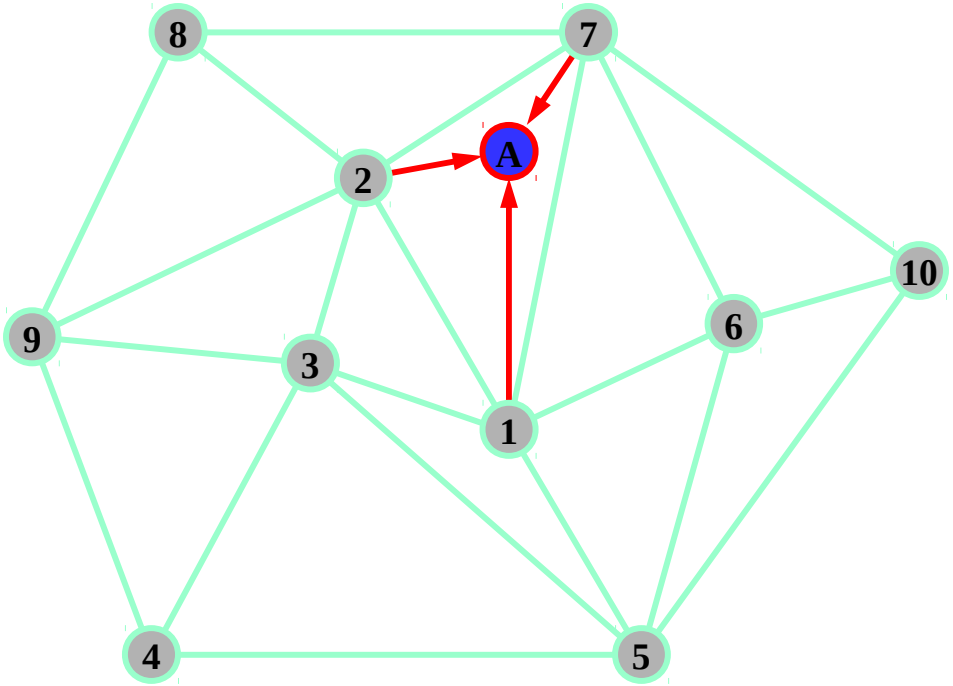
- Each node in vertex hierarchy *supports* a subset of the model vertices
  - Leaf nodes support a single vertex from the original full-resolution model
  - The root node supports all vertices
- For each node we also assign a representative vertex or *proxy*

Drevo verteksov: Zapiranje in odpiranje

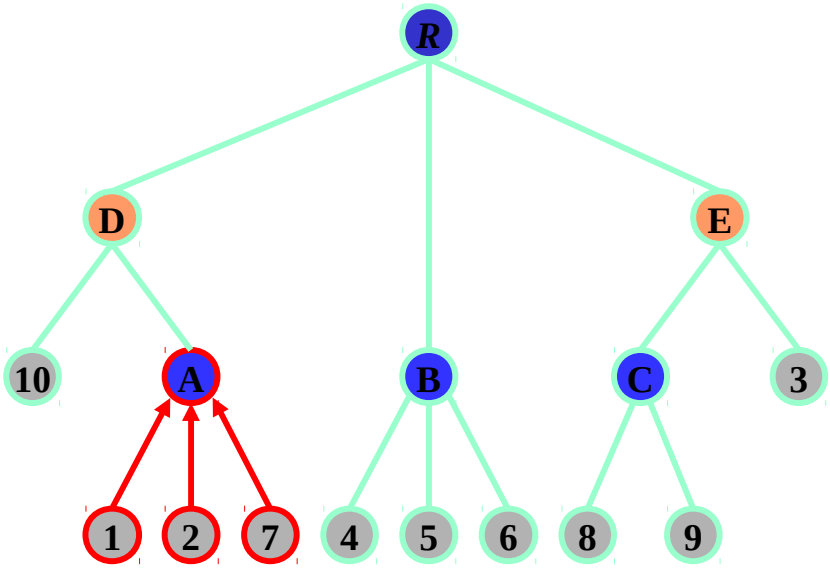
- *Folding* a node collapses its vertices to the proxy
- *Unfolding* the node splits the proxy back into vertices



Primer drevesa verteksov

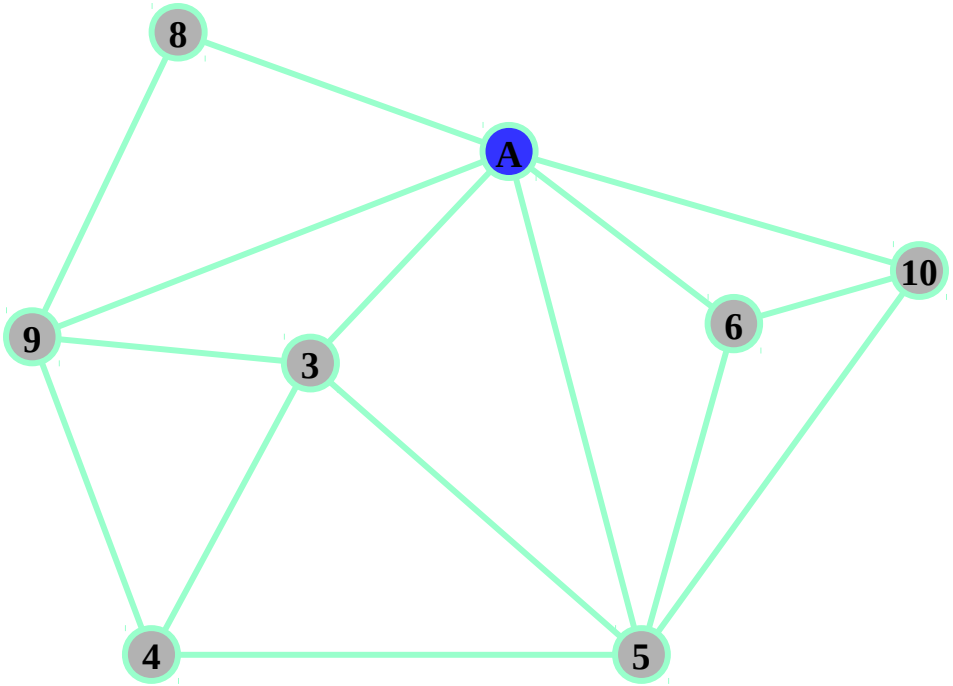


Triangles in active list

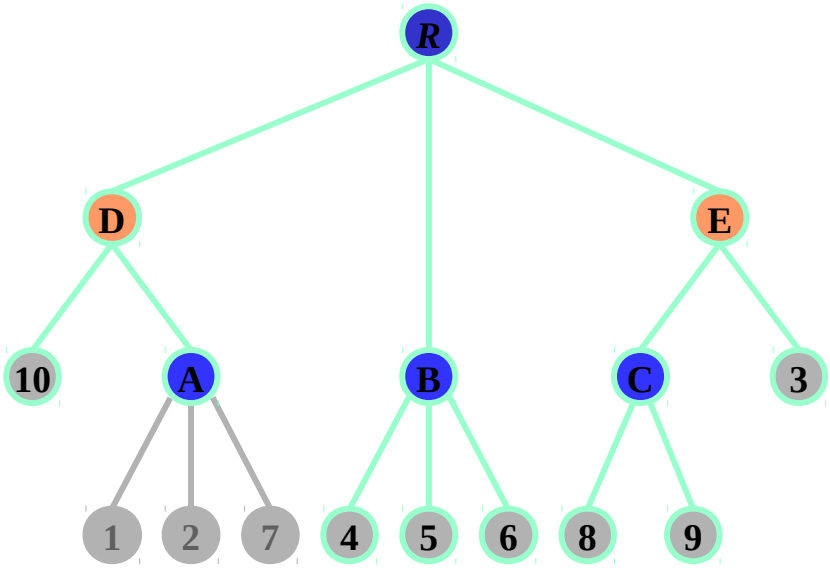


Vertex hierarchy

Primer drevesa verteksov

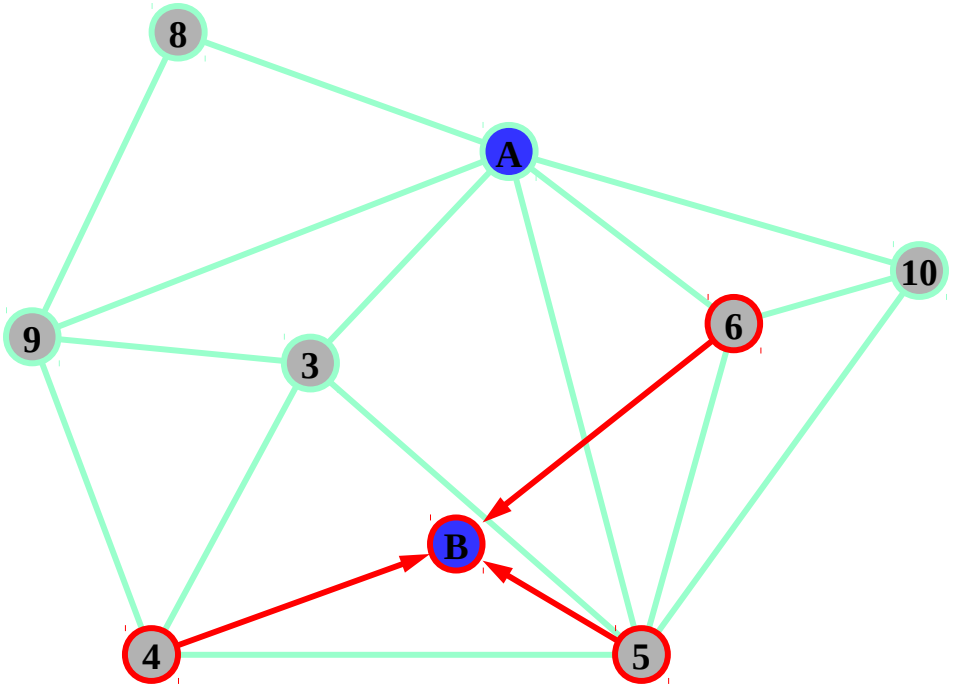


Triangles in active list

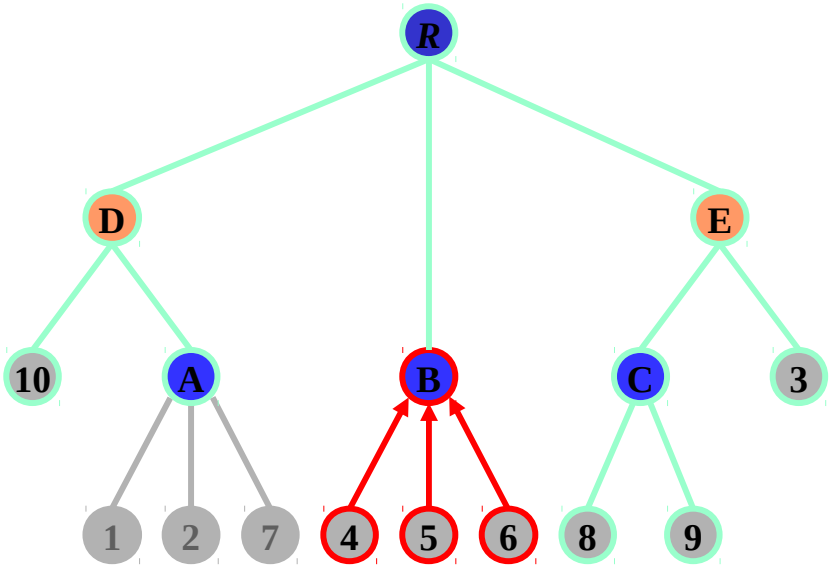


Vertex hierarchy

Primer drevesa verteksov

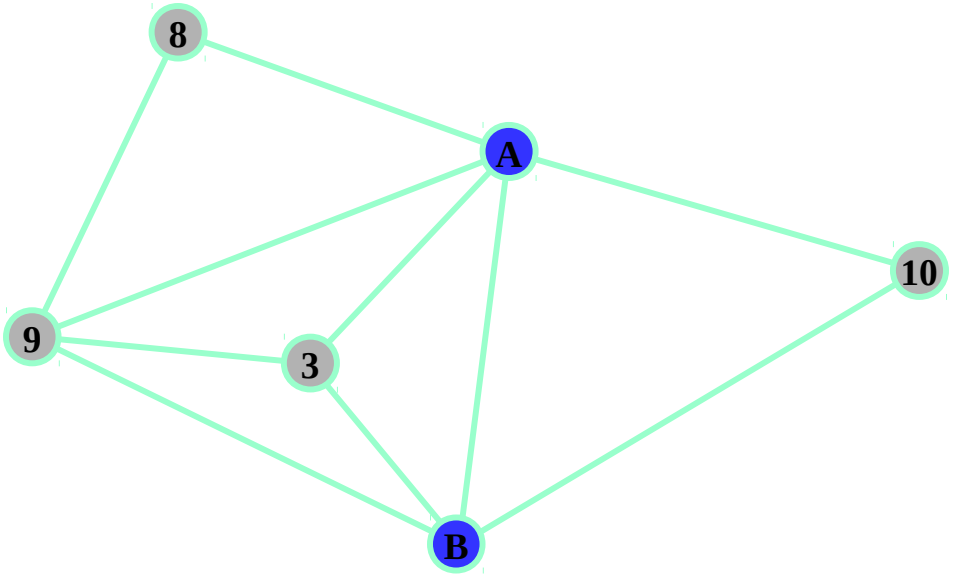


Triangles in active list

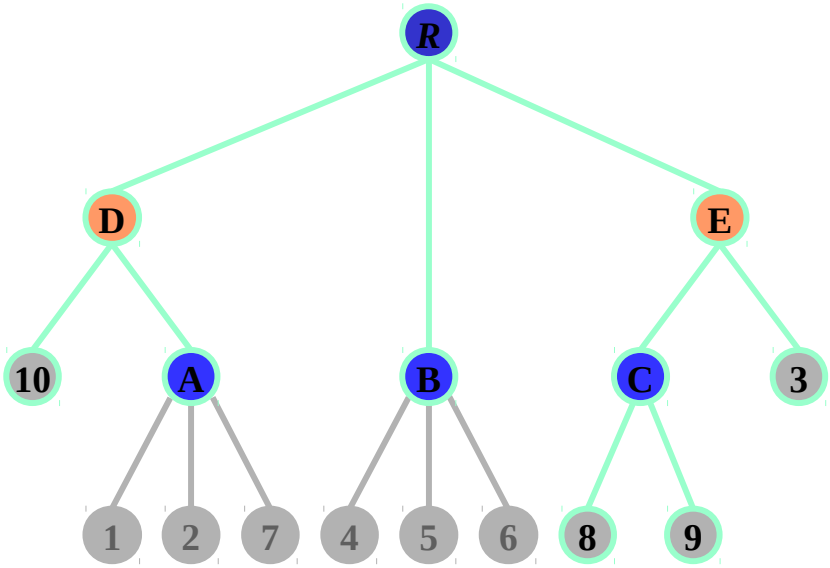


Vertex hierarchy

Primer drevesa verteksov

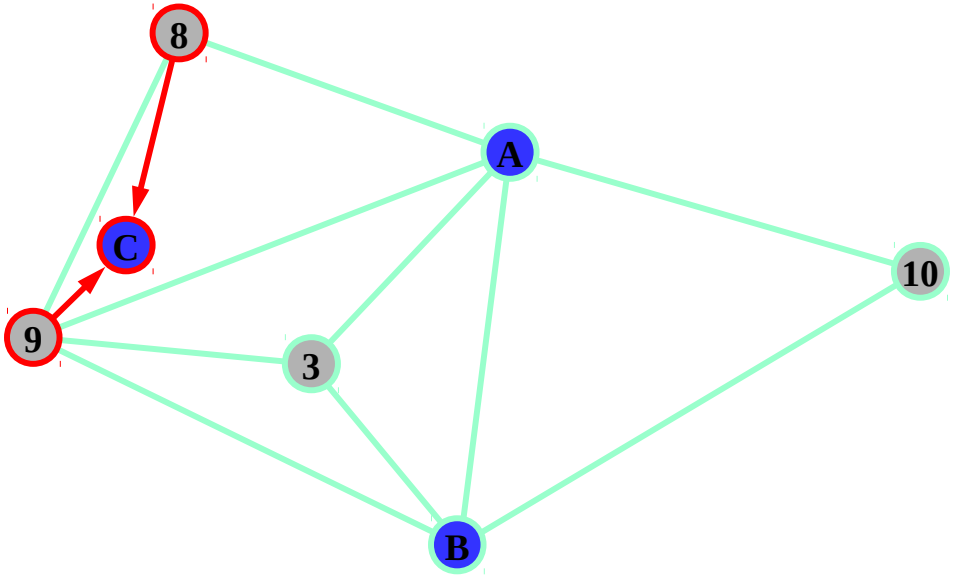


Triangles in active list

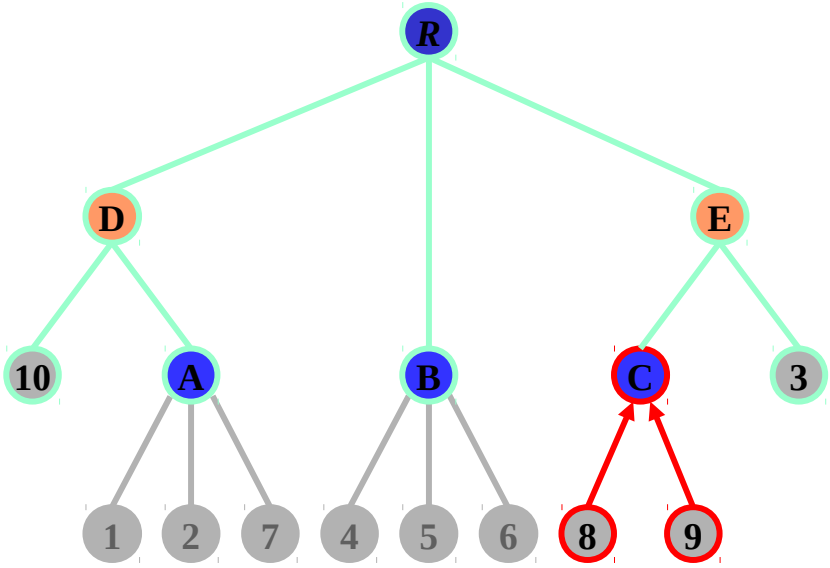


Vertex hierarchy

Primer drevesa verteksov

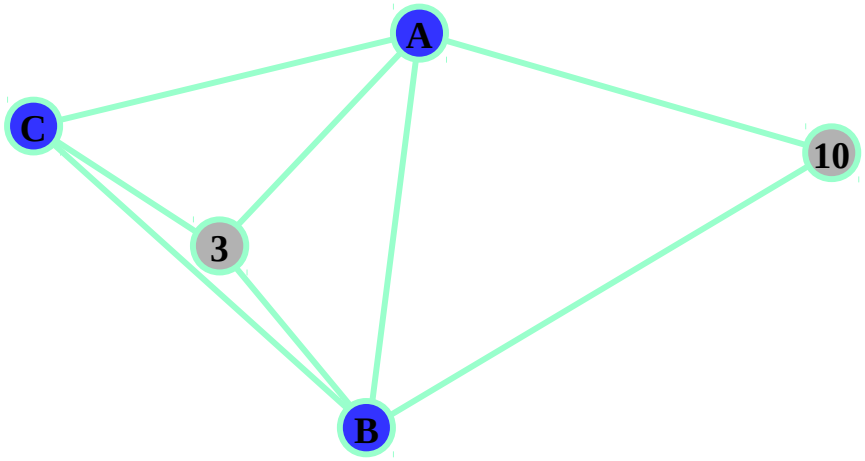


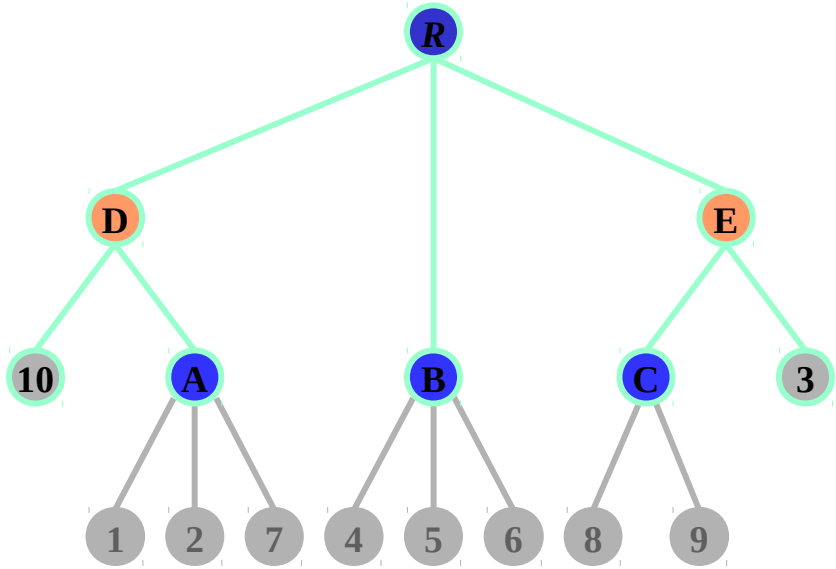
Triangles in active list



Vertex hierarchy

Primer drevesa verteksov

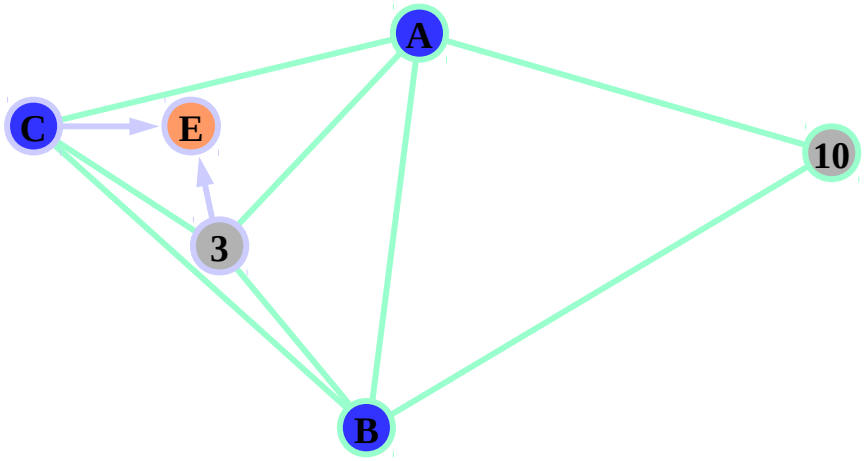


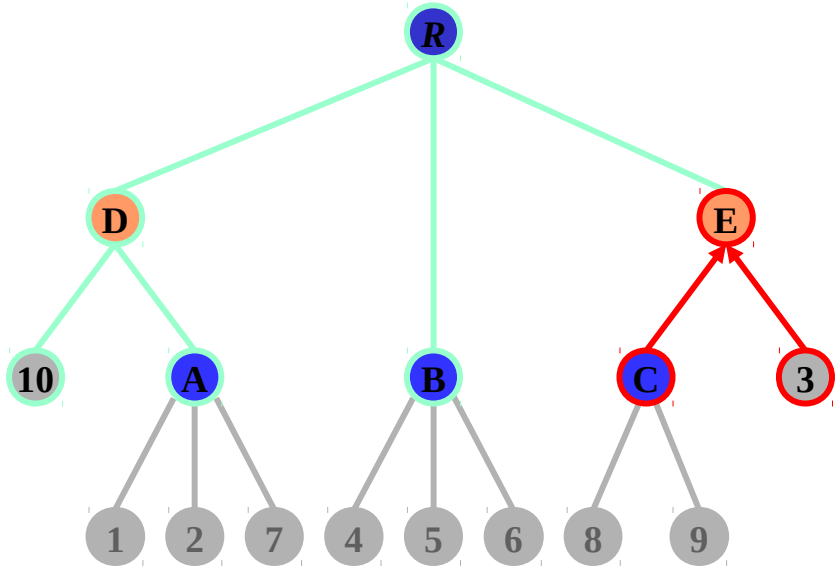


Triangles in active list

Vertex hierarchy

Primer drevesa verteksov

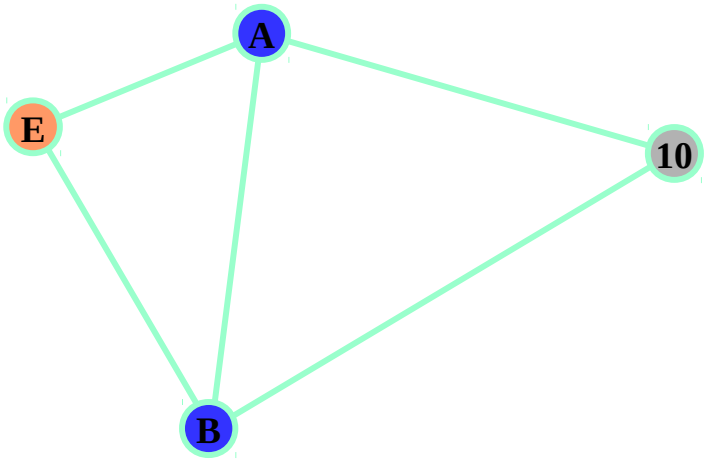


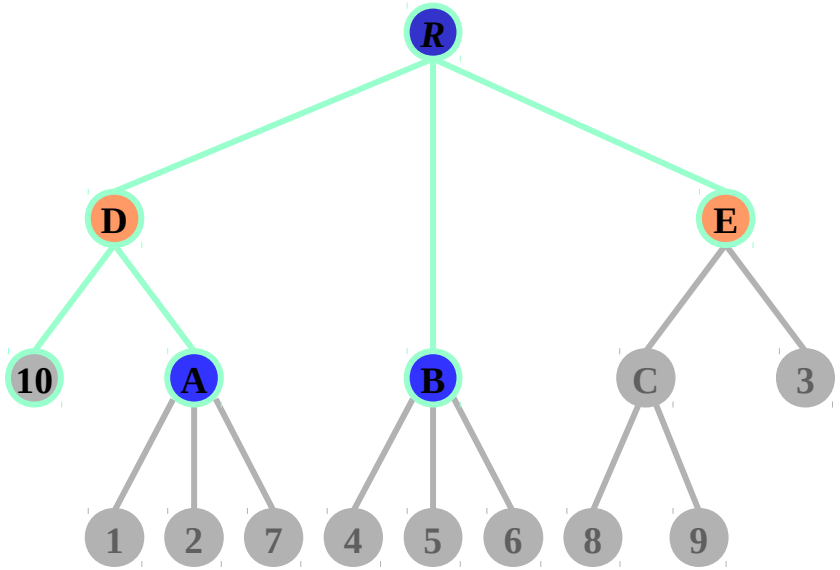


Triangles in active list

Vertex hierarchy

Primer drevesa verteksov

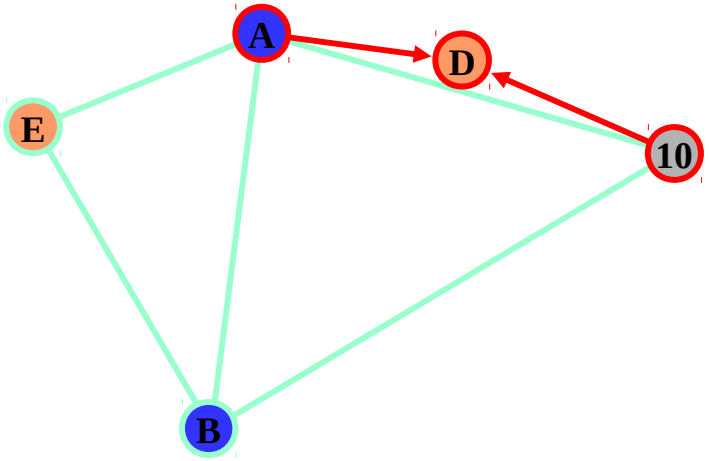


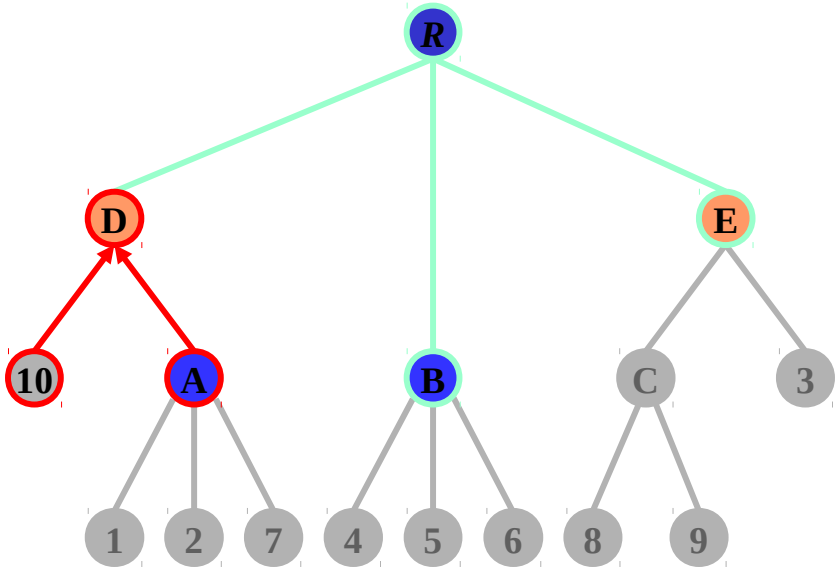


Triangles in active list

Vertex hierarchy

Primer drevesa verteksov

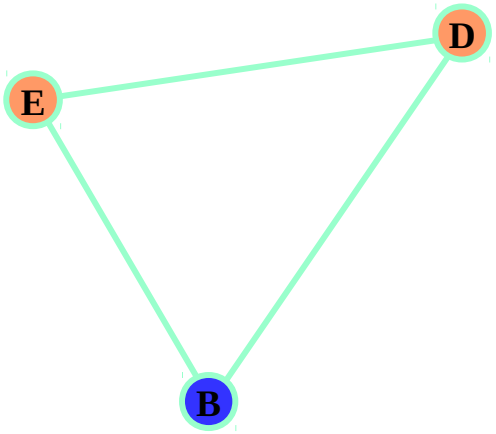




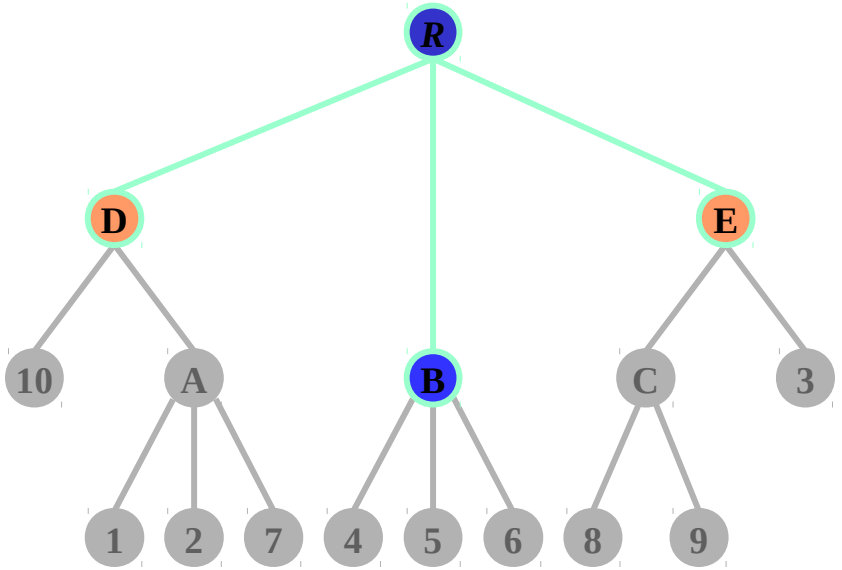
Triangles in active list

Vertex hierarchy

Primer drevesa verteksov

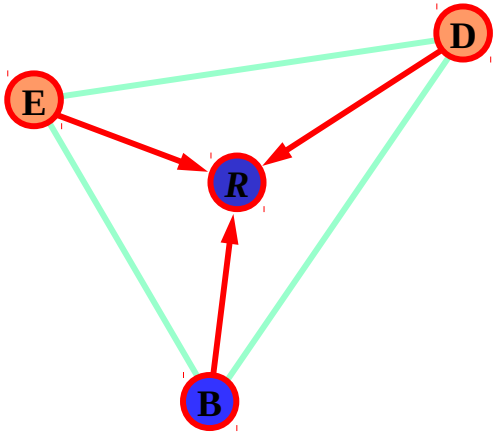


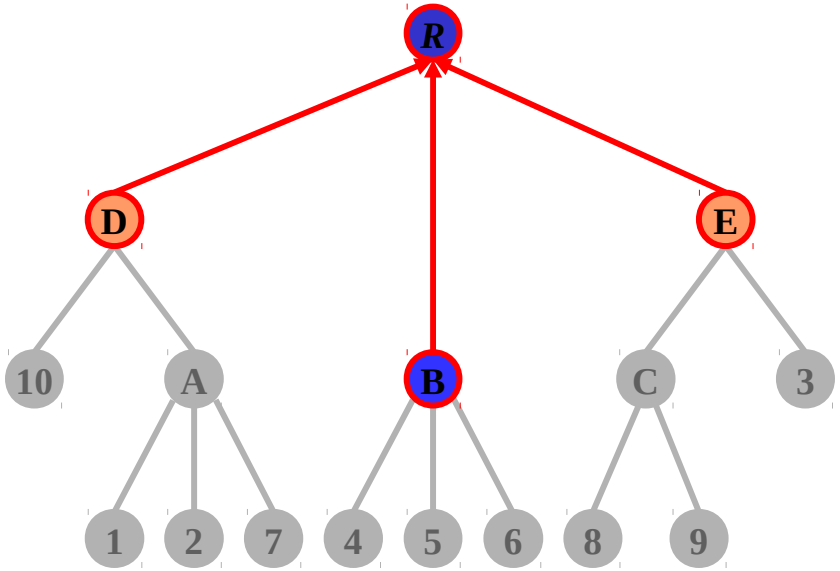
Triangles in active list



Vertex hierarchy

Primer drevesa verteksov

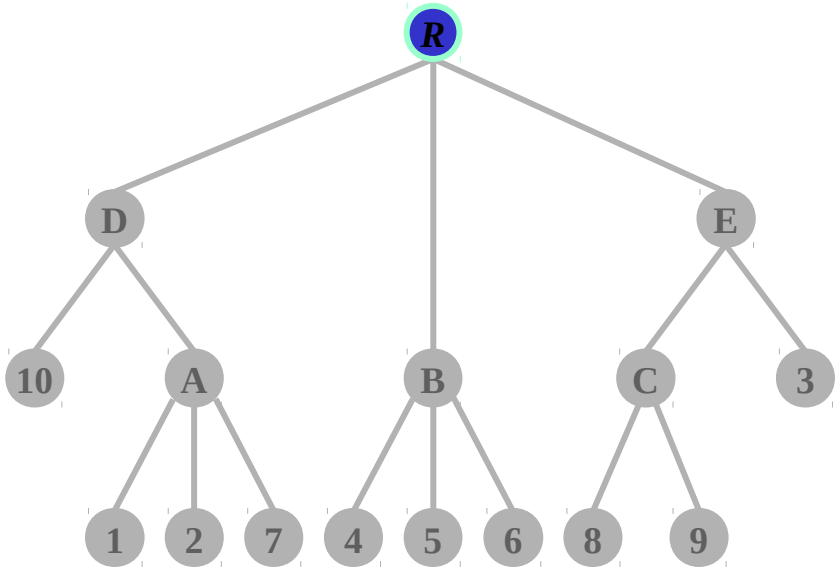




Triangles in active list

Vertex hierarchy

Primer drevesa verteksov

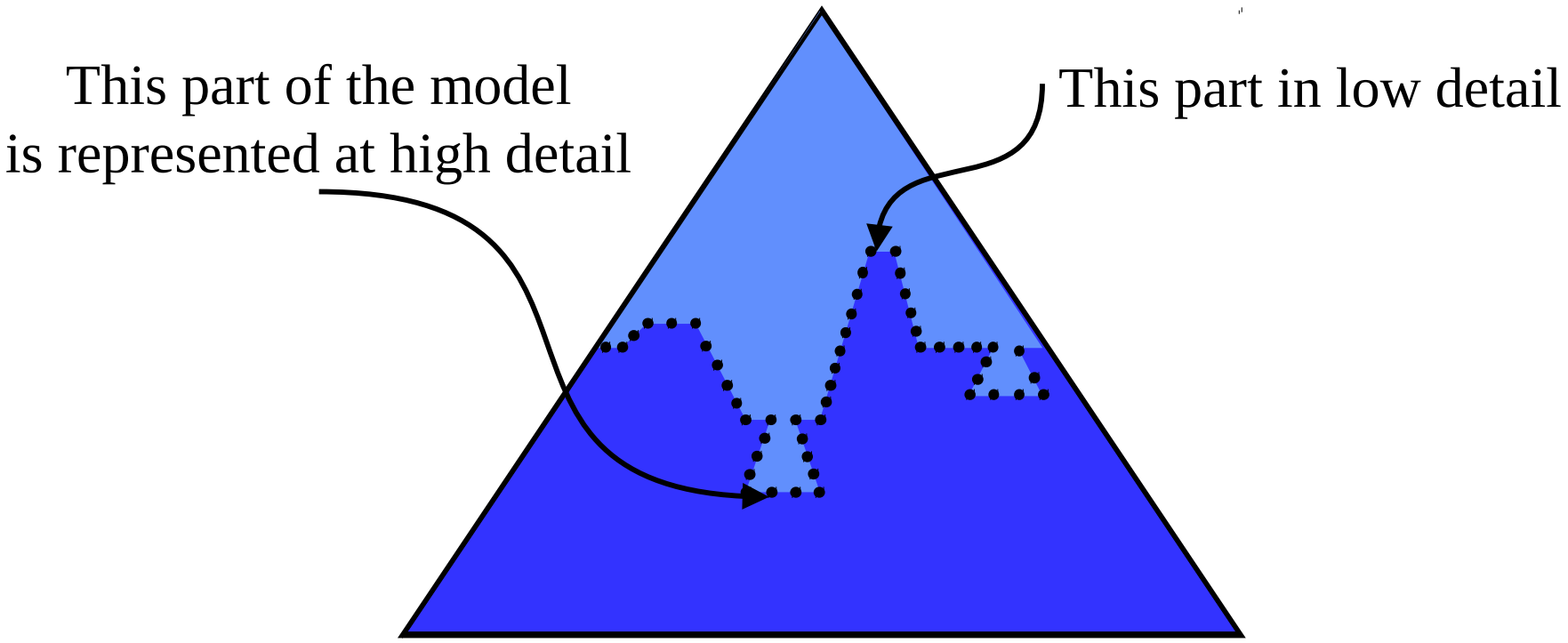


Triangles in active list

Vertex hierarchy

Drevo verteksov

- At runtime, folds and unfolds create a cut or *boundary* across the vertex tree:

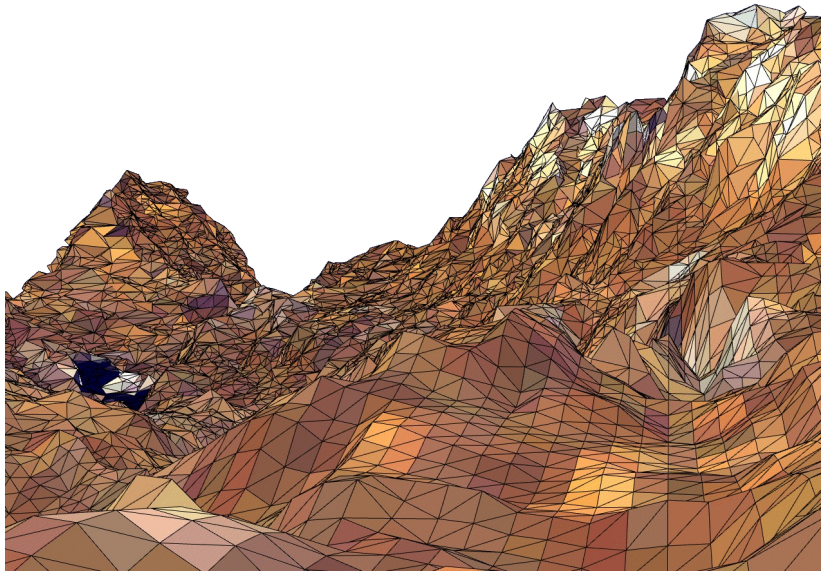


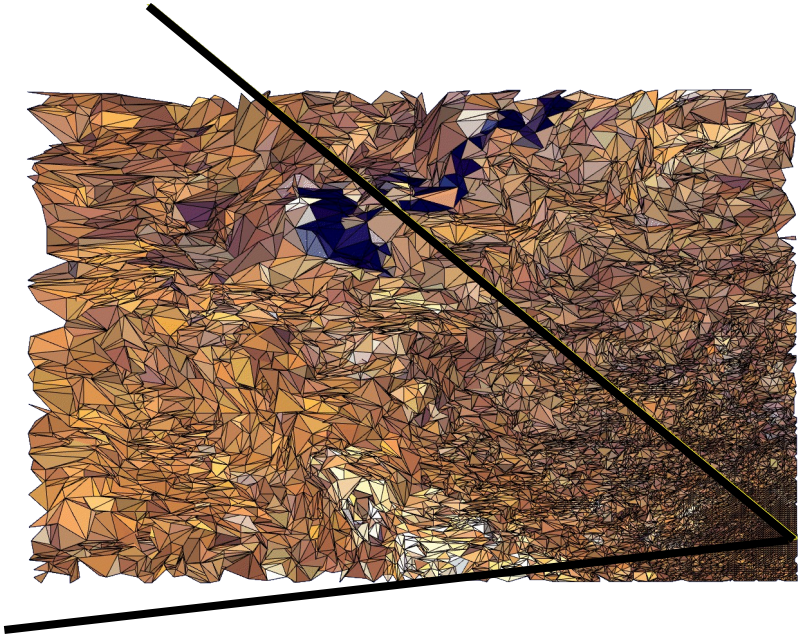
View-Dependent Simplification

- Any run-time criterion for folding and unfolding nodes may be used
- Examples of view-dependent simplification criteria:
  - Screenspace error threshold
  - Silhouette preservation
  - Triangle budget simplification
  - Gaze-directed perceptual simplification

Screenspace Error Threshold

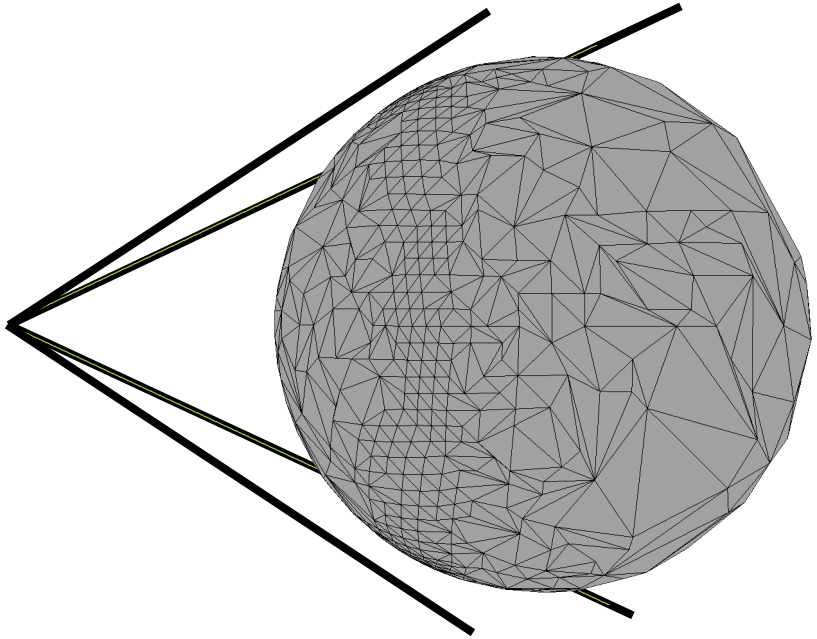
- Nodes chosen by projected area
  - User sets screenspace size threshold
  - Nodes which grow larger than threshold are unfolded





Ohranjevanje silhuet

- Retain more detail near silhouettes
  - A *silhouette node* supports triangles on the visual contour
  - Use tighter screenspace thresholds when examining silhouette nodes



Progressive Meshes

- We get more than just final approximation
  - sequence of contractions
  - corresponding intermediate approximations
- Re-encode as progressive mesh (PM)
  - take final approximation to be base mesh
  - reverse of contraction sequence is split sequence
  - can reconstruct any intermediate model
  - allow for progressive transmission & compression

PM's a Limited Multiresolution

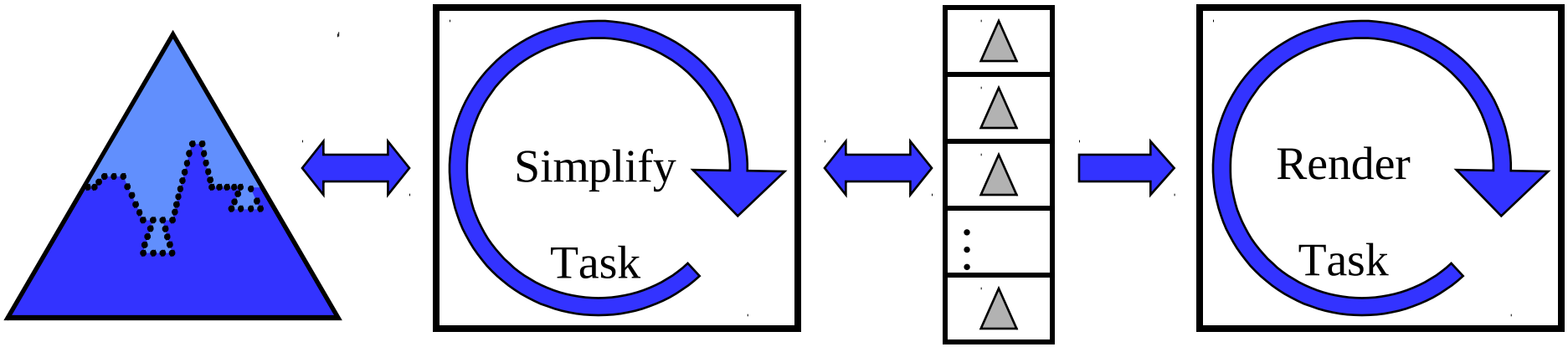
- More flexibility is required
  - local addition/subtraction of triangles
    - as conditions change, make small updates in LOD
    - this is the multi-triangulation framework
    - may require novel approximations
- Must encode dependency of contractions
  - PM's imply dependency on earlier contractions
  - but we can reorder non-overlapping contractions

Triangle Budget Simplification

- Minimize error within specified number of triangles
  - Sort nodes by screenspace error
  - Unfold node with greatest error, putting children into sorted list
  - Repeat until budget is reached

Asinhrona poenostavitev

- Algorithm partitions into two tasks:



- Run them in parallel

Časovna koherenca

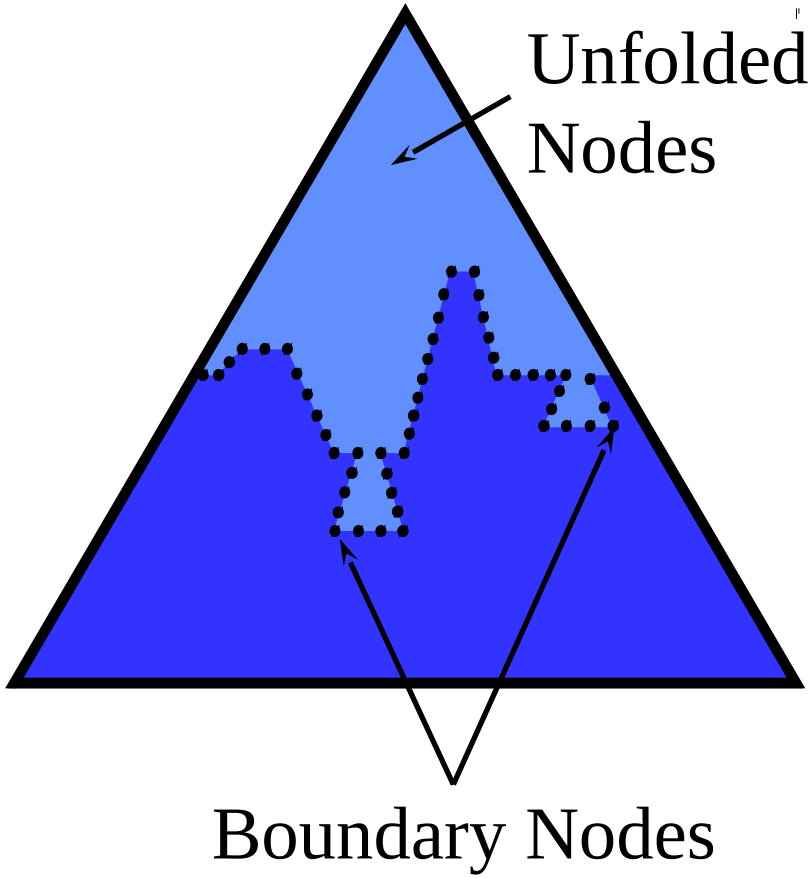
- Exploit the fact that frame-to-frame changes are small
- Three examples:
  - Active triangle list
  - Vertex tree
  - Budget-based simplification

Izkoriščanje časovne koherence

- Active triangle list
  - Could calculate active triangles every frame
  - But…few triangles are added or deleted each frame
  - Idea: make only incremental changes to an *active triangle list*
    - Simple approach: doubly-linked list of triangles
    - Better: maintain coherent arrays with swapping

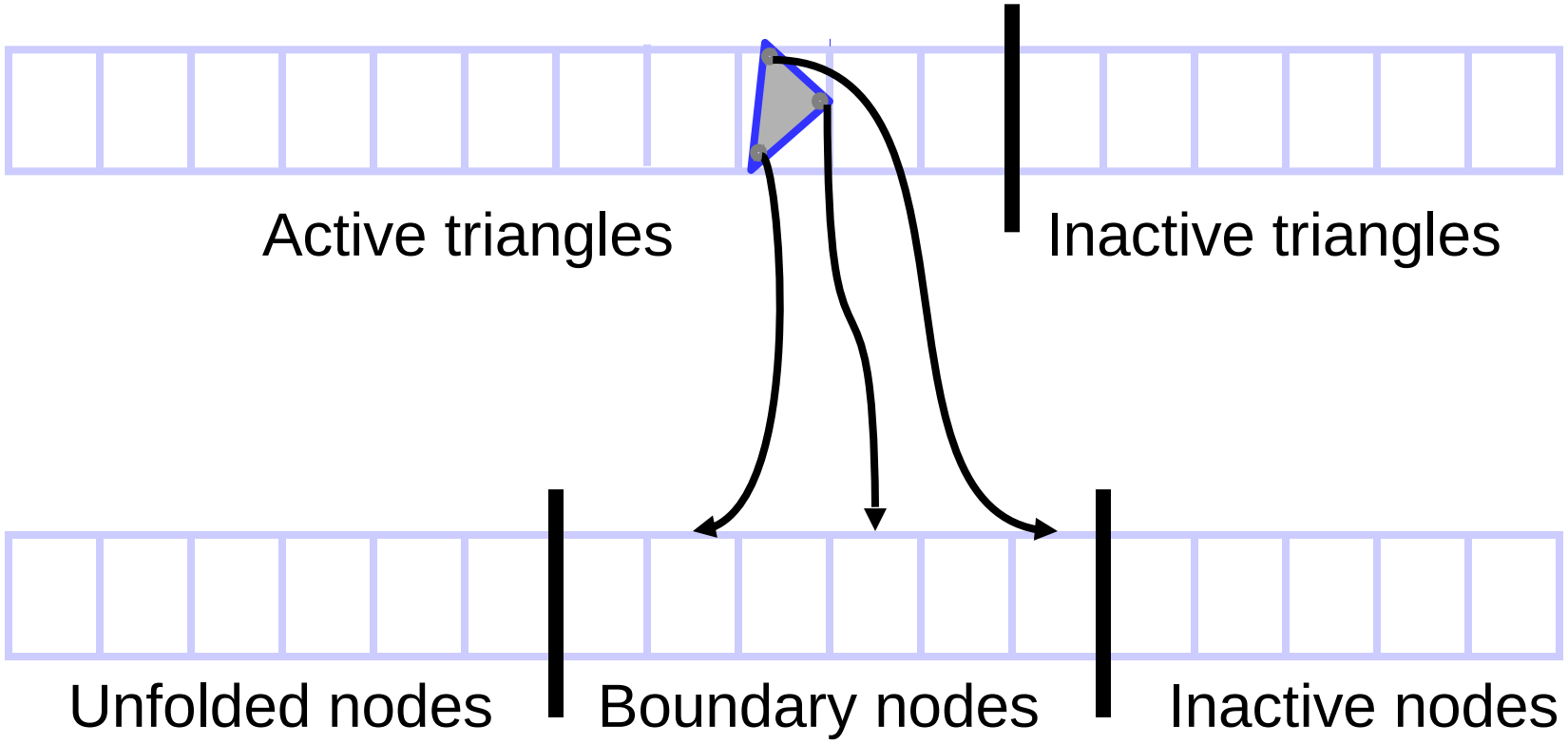
Izkoriščanje časovne koherence

- Vertex Tree
  - Few nodes change per frame
  - Don’t traverse whole tree
  - Do local updates only at *boundary nodes*



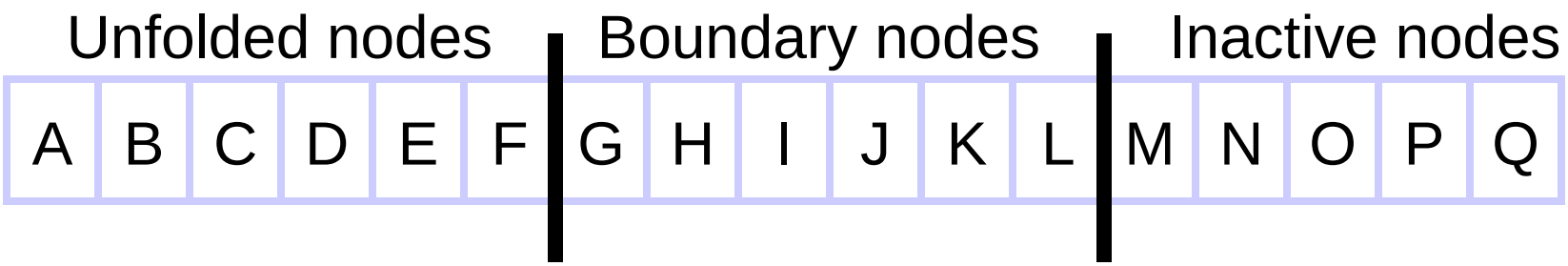
Optimizacija za upodabljanje

- Idea: maintain geometry in coherent arrays



Optimizacija za upodabljanje

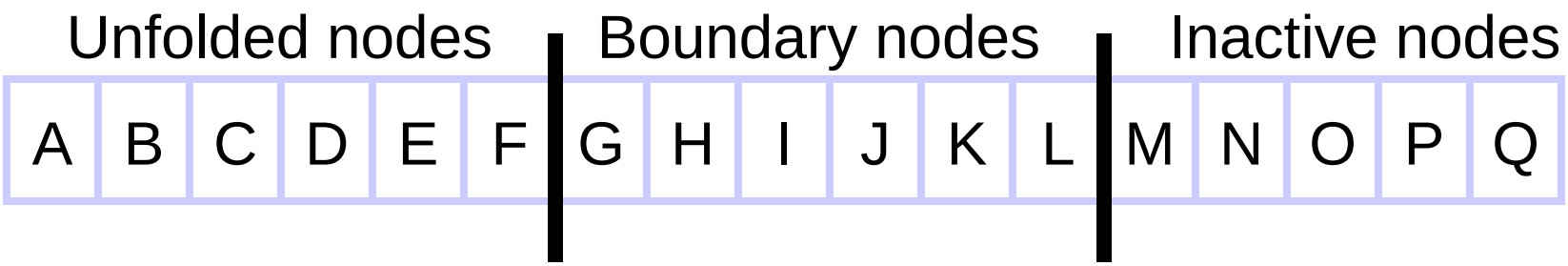
- Idea: use swaps to maintain coherence



Fold node D:

Optimizacija za upodabljanje

- Idea: use swaps to maintain coherence

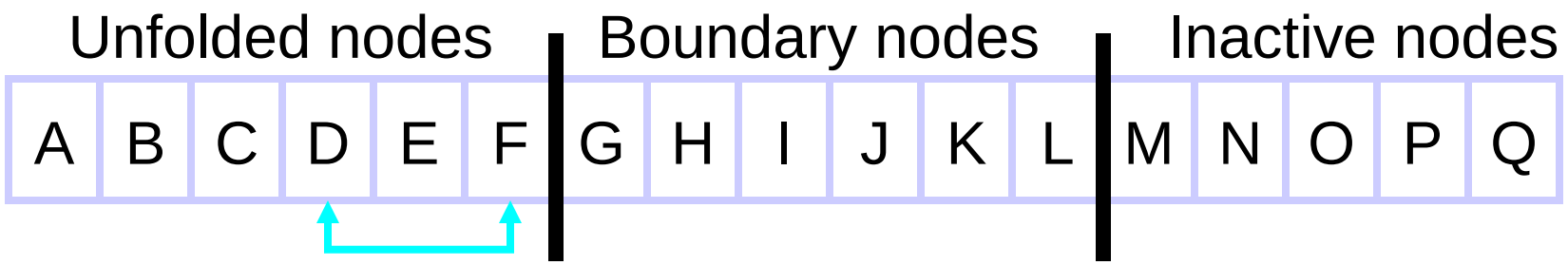


Fold node D:

Swap D with F

Optimizacija za upodabljanje

- Idea: use swaps to maintain coherence

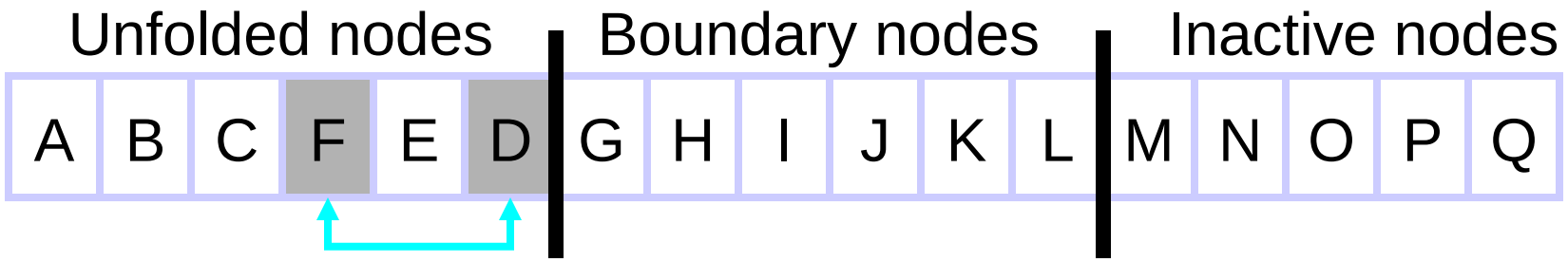


Fold node D:

Swap D with F

Optimizacija za upodabljanje

- Idea: use swaps to maintain coherence

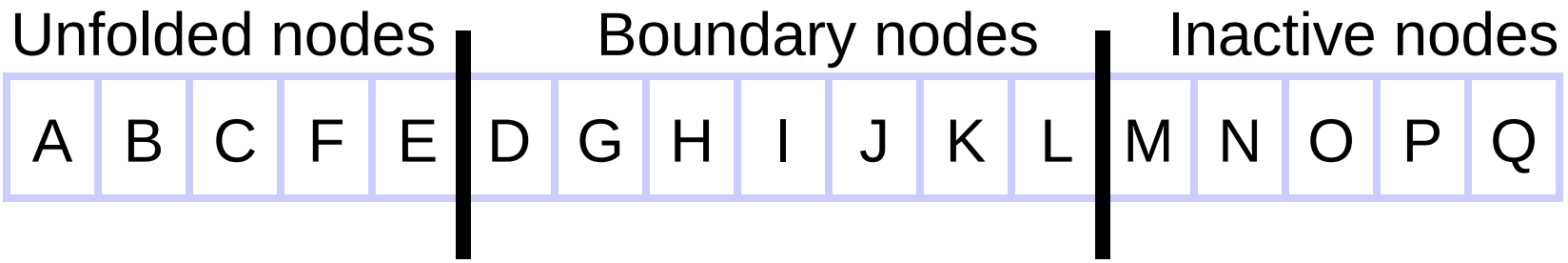


Fold node D:

Swap D with F

Optimizacija za upodabljanje

- Idea: use swaps to maintain coherence

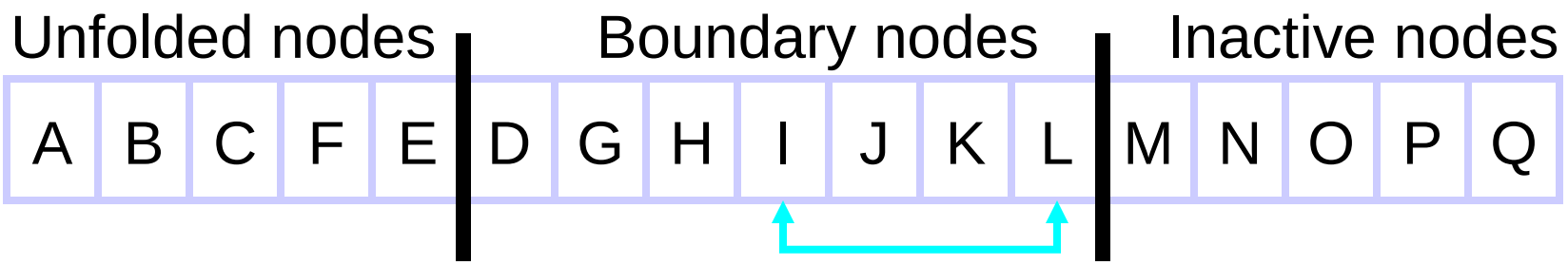


Fold node D:

Move Unfolded/Boundary Marker

Optimizacija za upodabljanje

- Idea: use swaps to maintain coherence

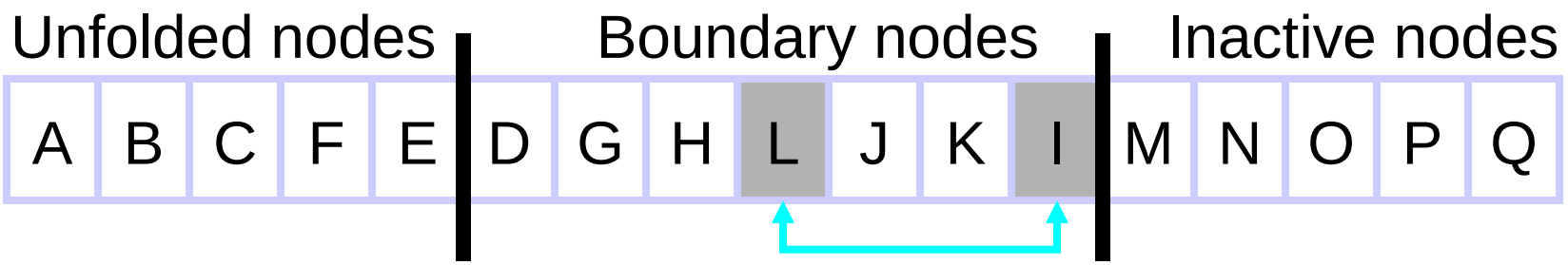


Fold node D:

Deactivate D’s children (swap w/ last boundary node)

Optimizacija za upodabljanje

- Idea: use swaps to maintain coherence

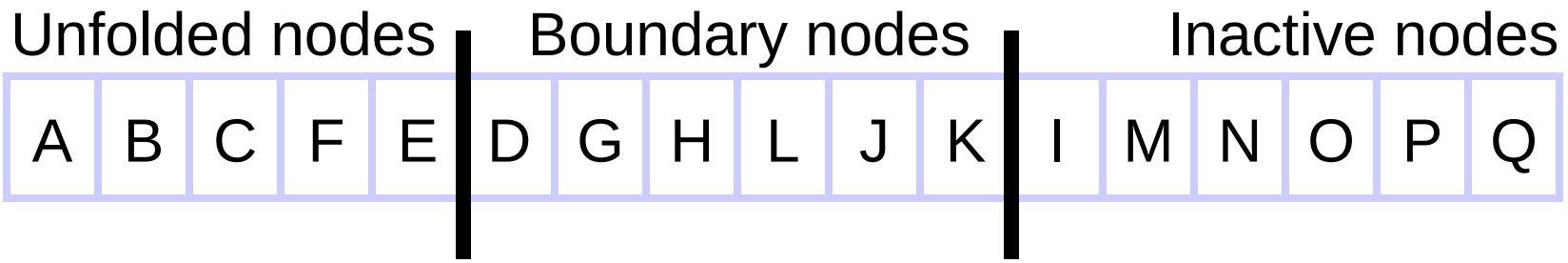


Fold node D:

Deactivate D’s children (swap w/ last boundary node)

Optimizacija za upodabljanje

- Idea: use swaps to maintain coherence

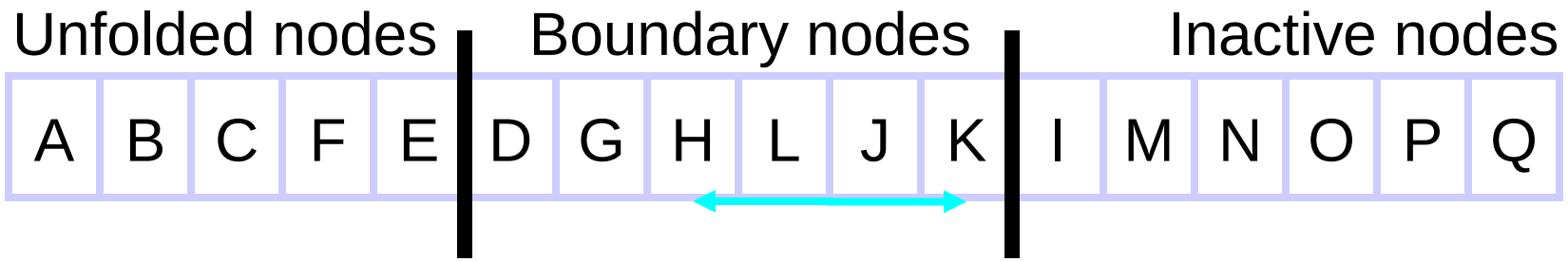


Fold node D:

Deactivate D’s children (swap w/ last boundary node)

Optimizacija za upodabljanje

- Idea: use swaps to maintain coherence

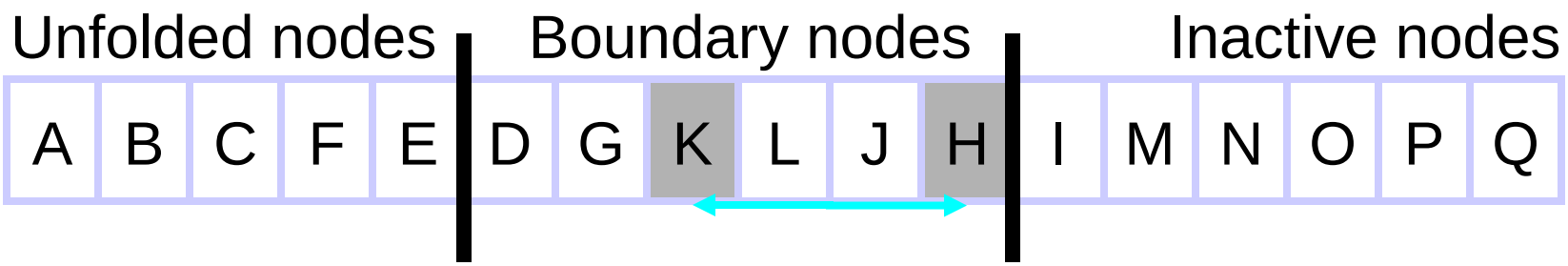


Fold node D:

Deactivate D’s children (swap w/ last boundary node)

Optimizacija za upodabljanje

- Idea: use swaps to maintain coherence

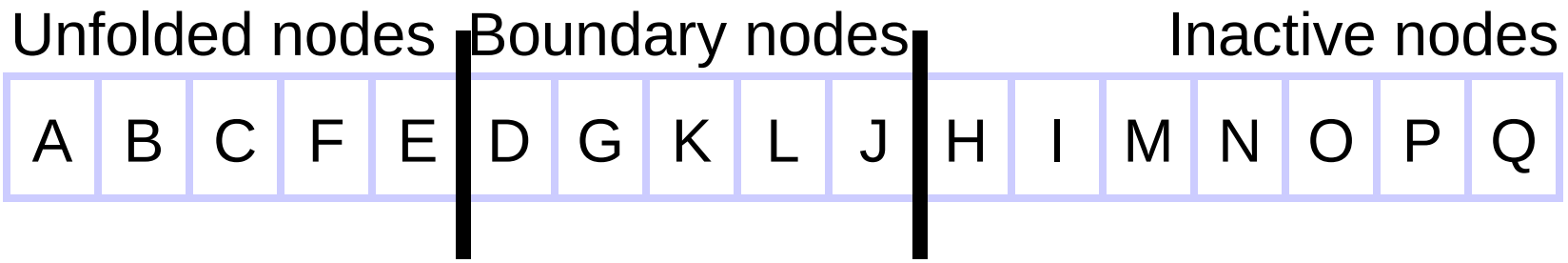


Fold node D:

Deactivate D’s children (swap w/ last boundary node)

Optimizacija za upodabljanje

- Idea: use swaps to maintain coherence

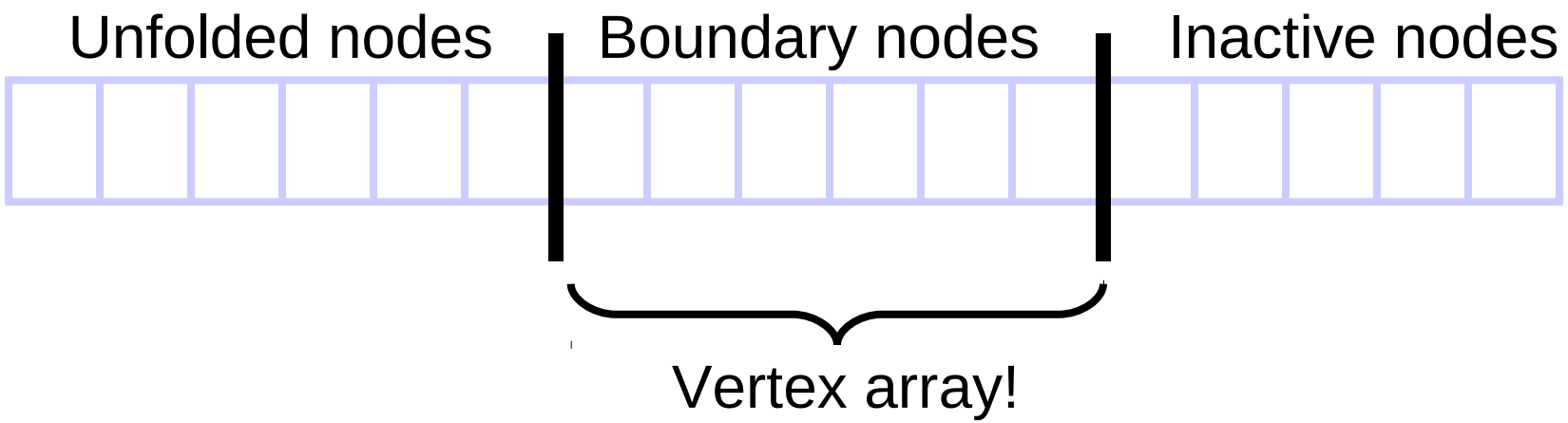


Fold node D:

Deactivate D’s children (swap w/ last boundary node)

Optimizacija za upodabljanje : polja verteksov

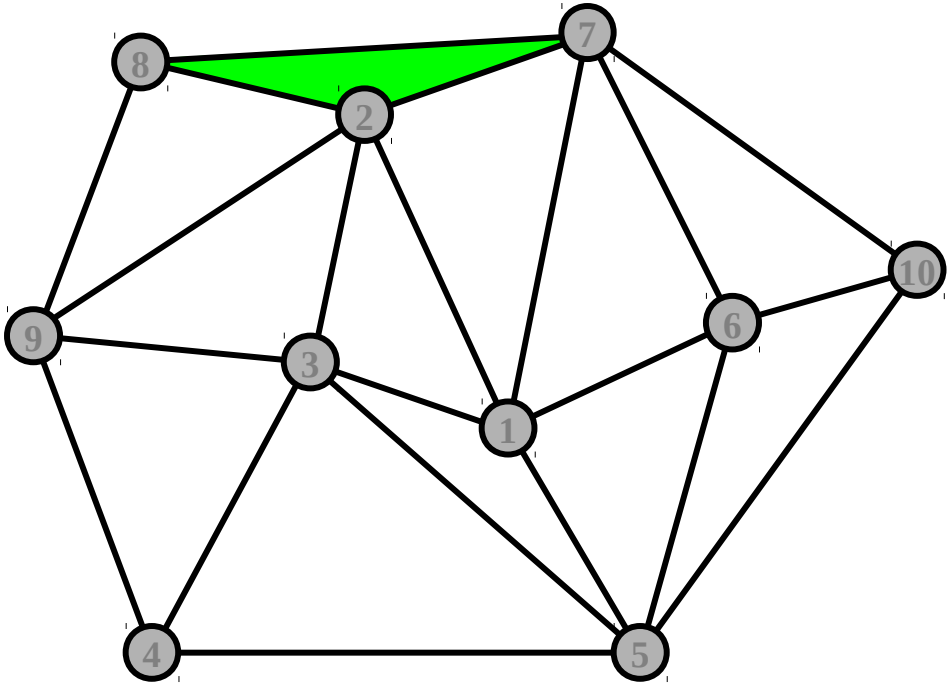
- Biggest win: vertex arrays



  - Actually, keep separate parallel arrays for rendering data (coords, colors, etc)

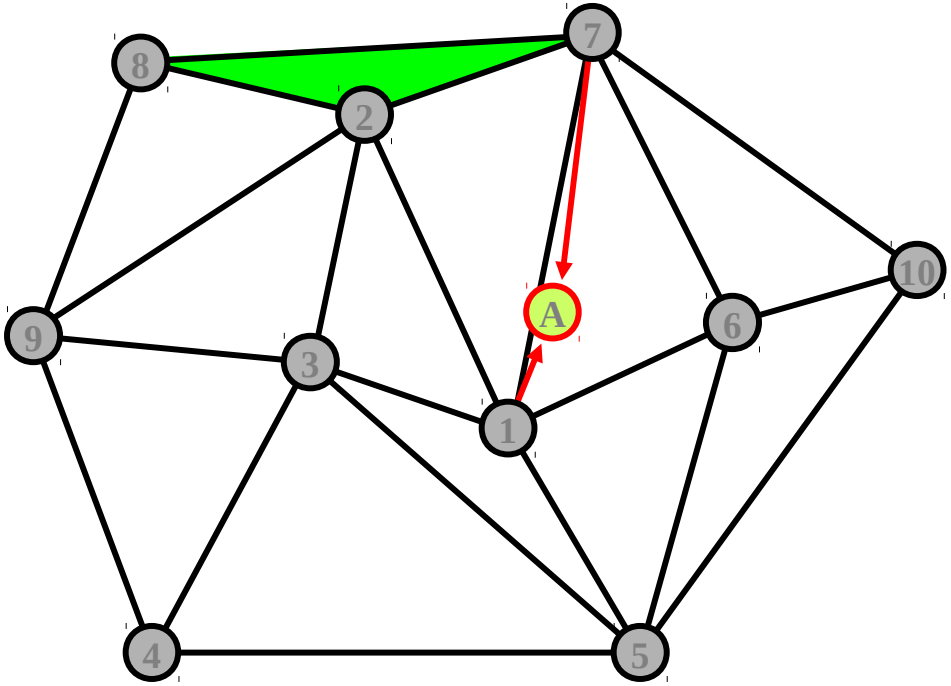
Povzetek: VDS Cons

- Increases CPU, memory overhead
- Hard to map efficiently onto GPU for efficient utilization
- Be aware of *mesh foldovers*



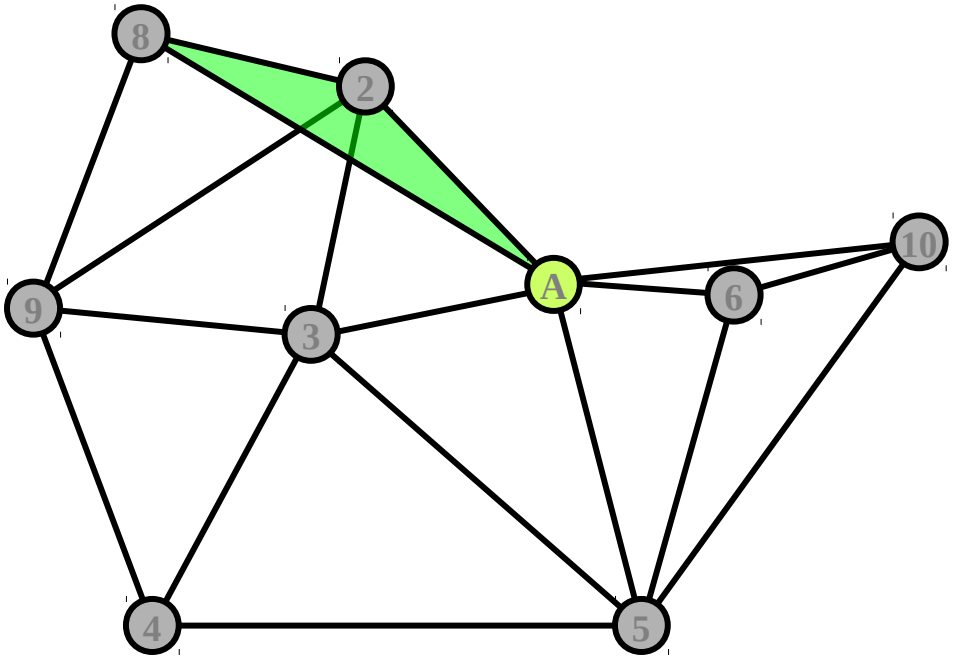
Povzetek: VDS Cons

- Be aware of *mesh foldovers:*



Povzetek: VDS Cons

- Be aware of *mesh foldovers:*



Povzetek: VDS Cons

- Be aware of *mesh foldovers:*
  - These can be very distracting artifacts
  - Can prevent them at run-time
    - Add a normal-flipping test to fold criterion
    - Use a clever numbering scheme proposed by El-Sana and Varshney

View-Dependent Versus Discrete LOD

- View-dependent LOD is superior to traditional discrete LOD when:
  - Models contain very large individual objects (e.g., terrains)
  - Simplification must be completely automatic (e.g., complex CAD models)
  - Experimenting with view-dependent simplification criteria

View-Dependent Versus Discrete LOD

- Discrete LOD is often the better choice:
  - Simplest programming model
  - Reduced run-time CPU load
  - Easier to leverage hardware:
    - Compile LODs into vertex arrays/display lists
    - Stripe LODs into triangle strips
    - Optimize vertex cache utilization and such

View-Dependent Versus Discrete LOD

- Applications that may want to use:
  - Discrete LOD
    - Video games (but much more on this later…)
    - Simulators
    - Many walkthrough-style demos
  - Dynamic and view-dependent LOD
    - CAD design review tools
    - Medical & scientific visualization toolkits
    - Terrain flyovers (much more later…)

Continuous LOD: The Sweet Spot?

- Continuous LOD may be the right compromise on modern PC hardware
  - Benefits of fine granularity without the cost of view-dependent evaluation
  - Can be implemented efficiently with regard to
    - Memory
    - CPU
    - GPU

Merjenje napake

- Most LOD algorithms measure error geometrically
  - What is the *distance* between the original and simplified surface?
  - What is the *volume* between the surfaces?
  - Etc
- Really this is just an approximation to the actual *visual* error, which includes:
  - Color, normal, & texture distortion
  - Importance of silhouettes, background illumination, semantic importance, etc etc etc

Merjenje geometrične napake

- Hausdorff distance
- Average distance
- Surface-surface vs
  vertex-surface vs
  vertex-plane vs
  vertex-vertex
- *Quadric error metrics*: vertex-plane measure that works well in practice

Cena krčenja

- Used to rank edges during simplification
  - reflects amount of geometric error introduced
  - main differentiating feature among algorithms
- Must address two interrelated problems
  - what is the best contraction to perform?
  - what is the best position v’ for remaining vertex?
    - can just choose one of the endpoints
    - but can often do better by optimizing position of v’

Cena krčenja

- Simple heuristics
  - edge length, dihedral angle, surrounding area, …
- Sample distances to original surface
  - projection to closest point [Hoppe]
  - restricted projection [Soucy–Laurendeau, Klein *et al*, Ciampalini *et al*]
- Alternative characterization of error
  - quadric error metrics [Garland–Heckbert]
  - local volume preservation [Lindstrom–Turk]

Measuring Error with Planes

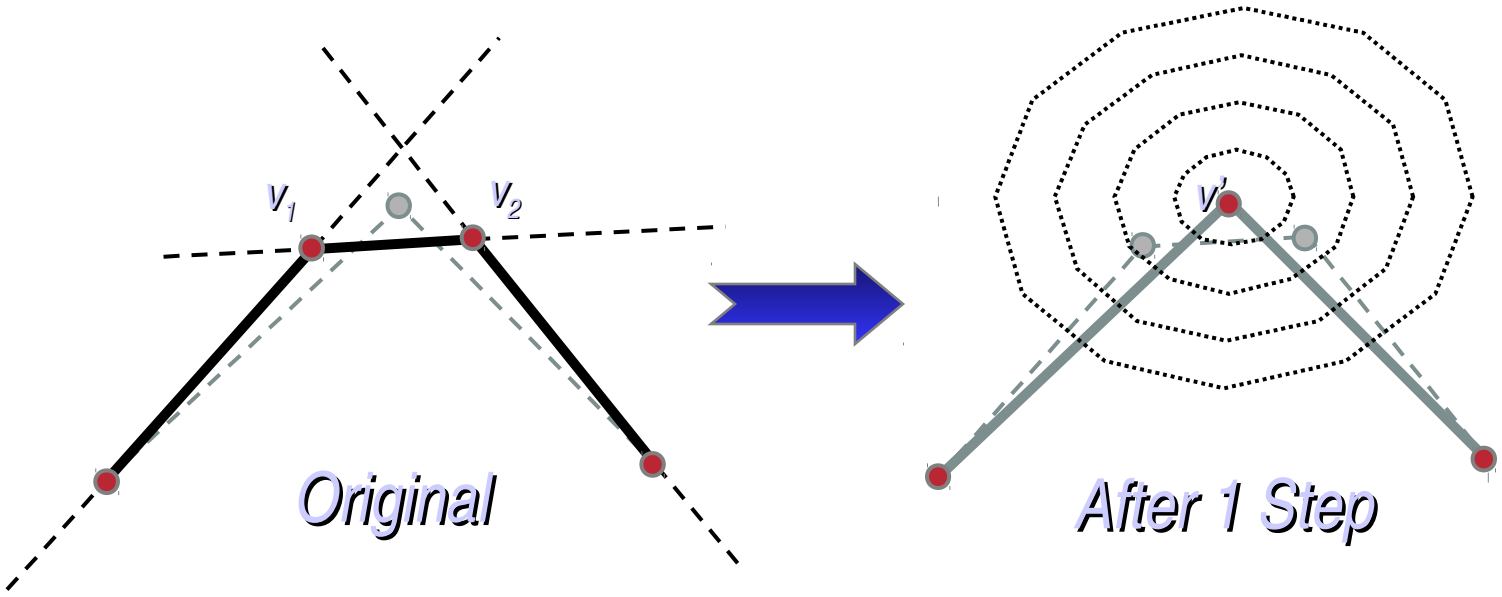
- Each vertex has a (conceptual) set of planes
  - Error ≡ sum of squared distances to planes in set

$$\text{Error}(\mathbf{v}) = \sum_i (\mathbf{n}_i^{\mathsf{T}}\mathbf{v} + d_i)^2$$

- Initialize with planes of incident faces
  - Consequently, all initial errors are 0
- When contracting pair, use plane set union
  - planes(v') = planes(v_1) ∪ planes(v_2)

A Simple Example: Contraction & “Planes” in 2D

- Lines defined by neighboring segments
  - Determine position of new vertex
  - Accumulate lines for ever larger areas



Measuring Error with Planes

- Why base error on planes?
  - Faster, but less accurate, than distance-to-face
  - Simple linear system for minimum-error position
  - Efficient implicit form; no sets required
  - Drawback: unlike surface, planes are infinite
- Related error metrics
  - Ronfard & Rossignac — max vs. sum
  - Lindstrom & Turk — similar form; volume-based

The Quadric Error Metric

- Given a plane, we can define a quadric Q

$$Q = (\mathbf{A}, \mathbf{b}, c) = (\mathbf{n}\mathbf{n}^{\mathsf{T}}, d\mathbf{n}, d^2)$$

measuring squared distance to the plane as

$$Q(\mathbf{v}) = \mathbf{v}^{\mathsf{T}}\mathbf{A}\mathbf{v} + 2\mathbf{b}^{\mathsf{T}}\mathbf{v} + c$$

$$Q(\mathbf{v}) = \begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} a^2 & ab & ac \\ ab & b^2 & bc \\ ac & bc & c^2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + 2 \begin{bmatrix} ad & bd & cd \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + d^2$$

The Quadric Error Metric

- Sum of quadrics represents set of planes

$$\sum_i (\mathbf{n}_i^{\mathsf{T}}\mathbf{v} + d_i)^2 = \sum_i Q_i(\mathbf{v}) = \left(\sum_i Q_i\right)(\mathbf{v})$$

- Each vertex has an associated quadric
  - Error(v_i) = Q_i (v_i)
  - Sum quadrics when contracting $(v_i, v_j) \rightarrow v'$
  - Cost of contraction is $Q(v')$

$$Q = Q_i + Q_j = (\mathbf{A}_i + \mathbf{A}_j,\ \mathbf{b}_i + \mathbf{b}_j,\ c_i + c_j)$$

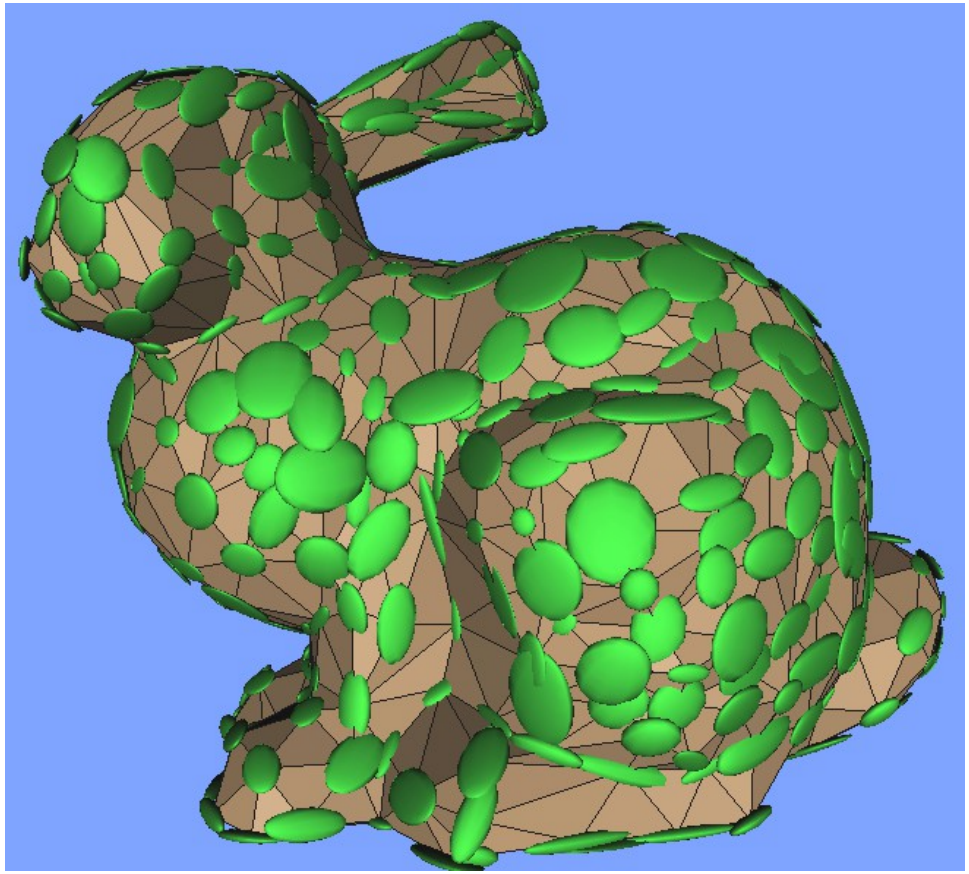
The Quadric Error Metric

- Sum of endpoint quadrics determines v'
  - Fixed placement: select v_1 or v_2
  - Optimal placement: choose *v'* minimizing $Q(v')$

$$\nabla Q(\mathbf{v}') = 0 \;\Rightarrow\; \mathbf{v}' = -\mathbf{A}^{-1}\mathbf{b}$$

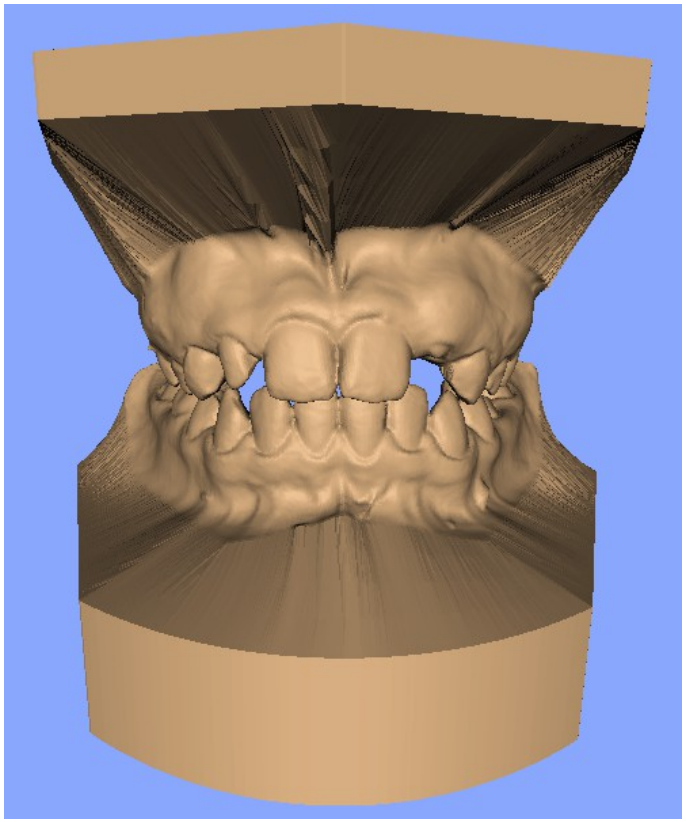
  - Fixed placement is faster but lower quality
  - But it also gives smaller progressive meshes
  - Fallback to fixed placement if A is non-invertible

Visualizing Quadrics in 3-D



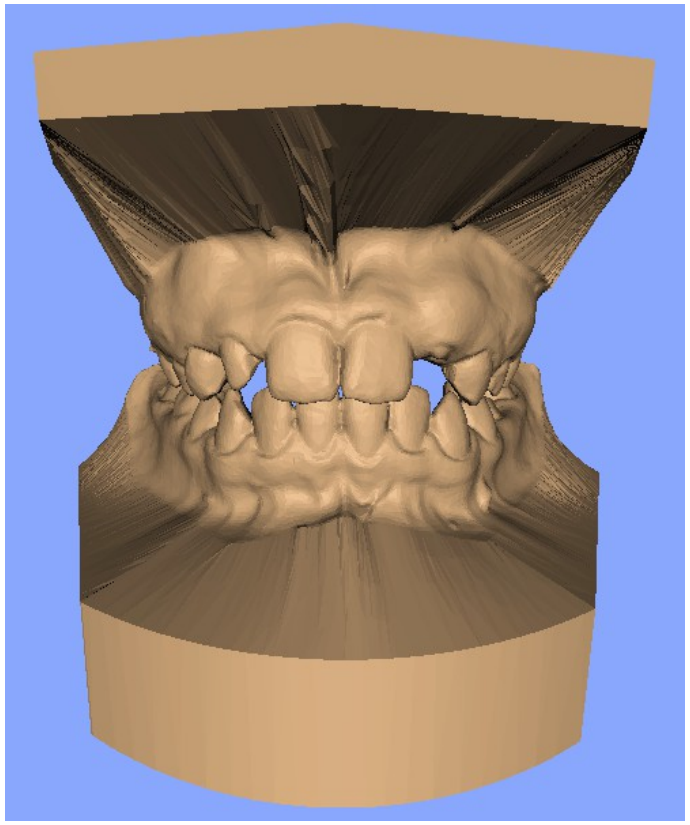
- Quadric isosurfaces
  - Are ellipsoids (maybe degenerate)
  - Centered around vertices
  - Characterize shape
  - Stretch in least-curved directions

Sample Model: Dental Mold



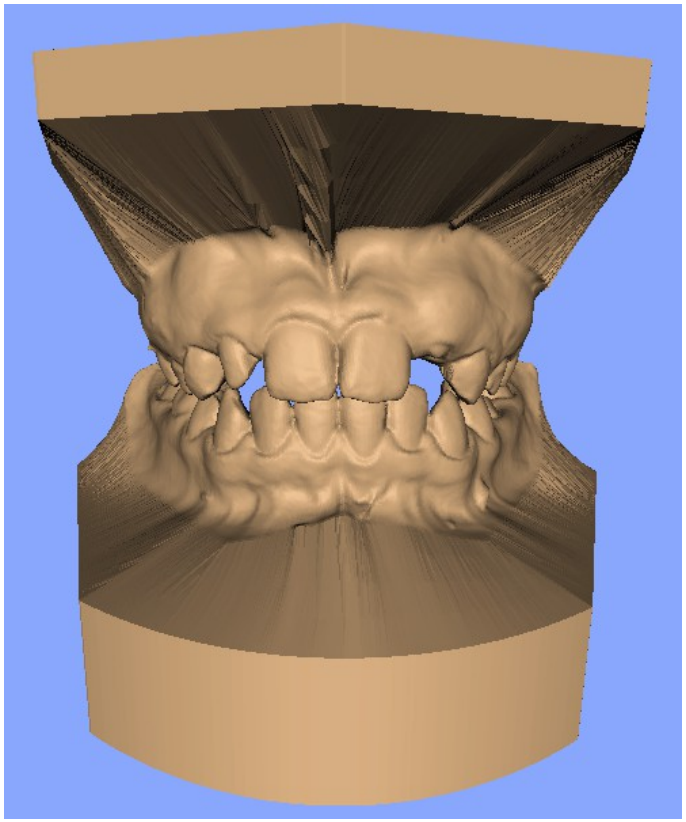
424,376 faces

50 sec



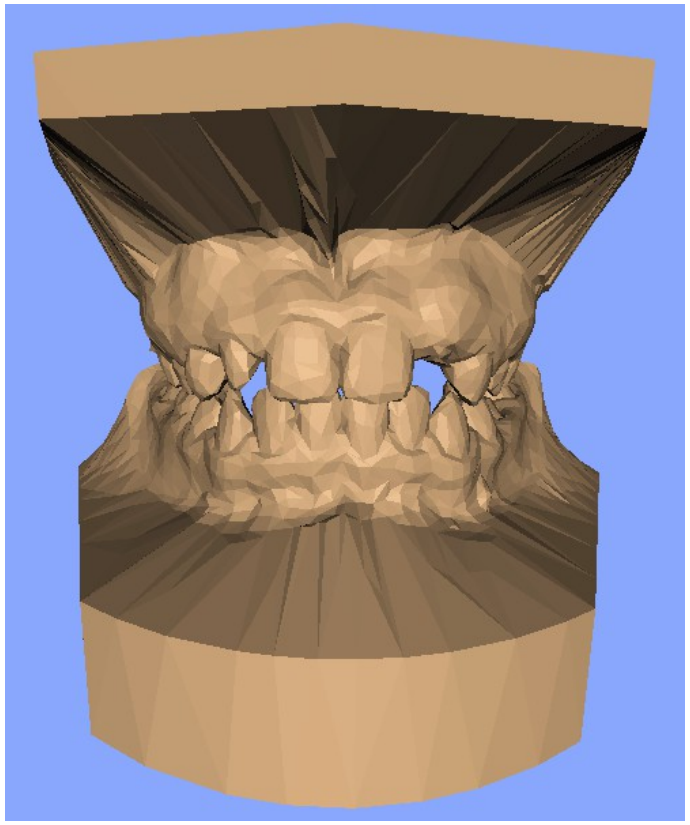
60,000 faces

Sample Model: Dental Mold



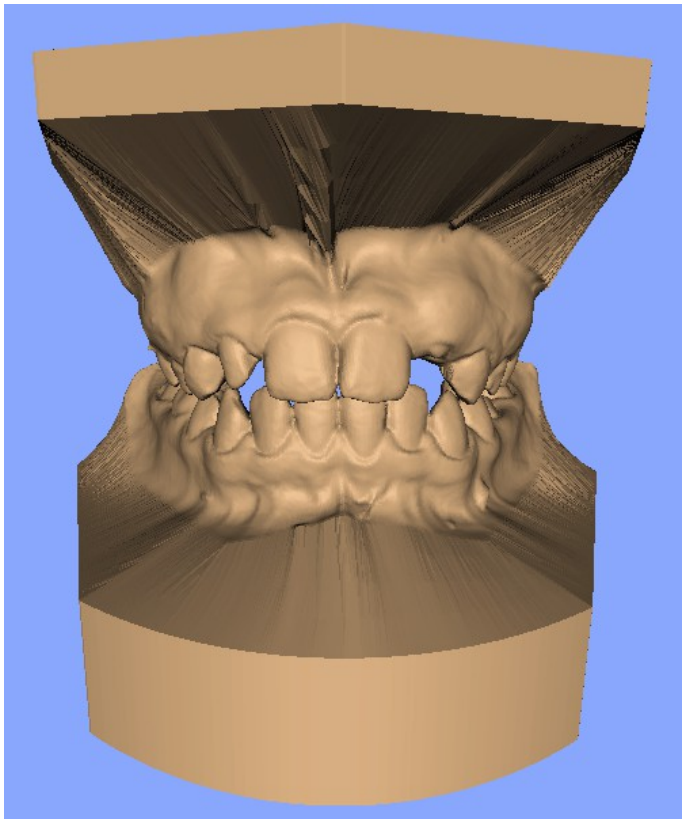
424,376 faces

55 sec



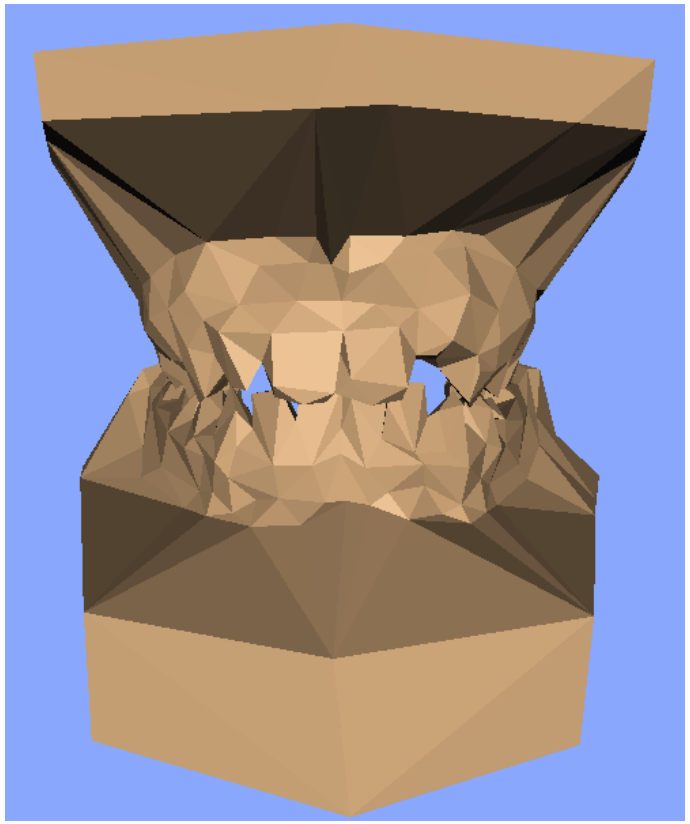
8000 faces

Sample Model: Dental Mold



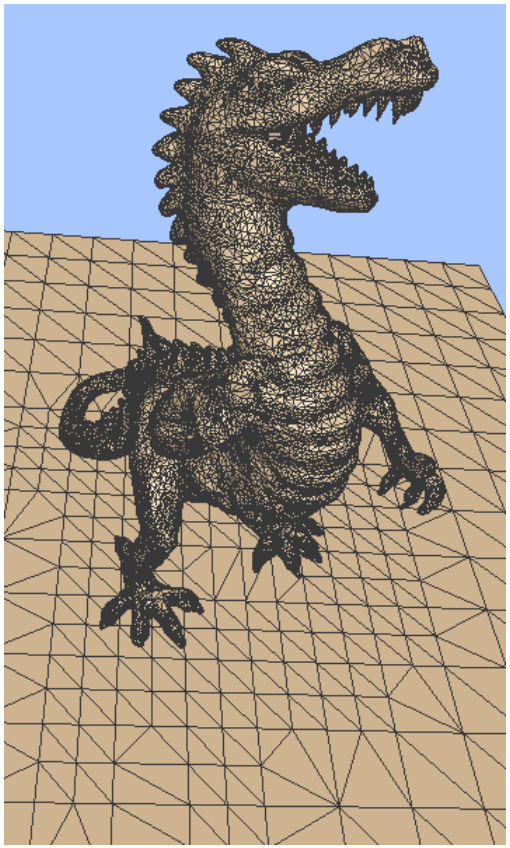
424,376 faces

56 sec



1000 faces

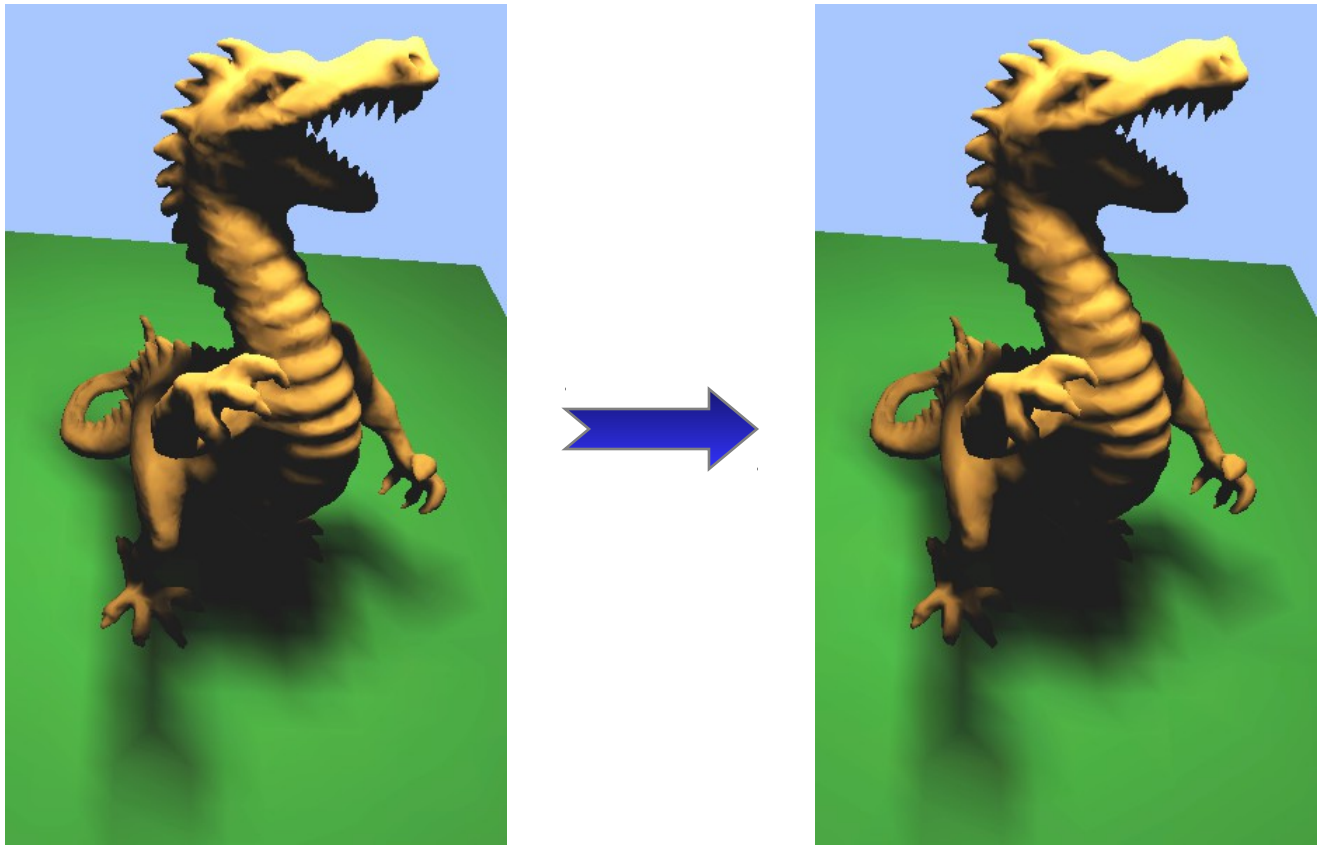
Must Also Consider Attributes



Mesh for solution

Radiosity solution

Must Also Consider Attributes



50,761 faces

10,000 faces

Simplification Summary

- Spectrum of effective methods developed
  - high quality; very slow *[Hoppe et al, Hoppe]*
  - good quality; varying speed
    [Schroeder et al; Klein et al; Ciampalini et al; Guéziec Garland-Heckbert; Ronfard-Rossignac; Lindstrom-Turk]
  - lower quality; very fast *[Rossignac–Borrel; Low–Tan]*
  - result usually produced by transforming original
- Various other differentiating factors
  - is topology simplified? restricted to manifolds?
  - attributes simplified or re-sampled into maps?

Applications Beyond Display

- Other important applications are appearing
  - surface editing
  - surface morphing
  - multiresolution radiosity
- Still others seem promising
  - hierarchical bounding volumes
  - object matching
  - shape analysis / feature extraction

Multiresolution Model Summary

- Representations are available to support
  - progressive transmission
  - view-dependent refinement
  - hierarchical computation (e.g., radiosity)
- But limitations remain
  - vertex hierarchies may over-constrain adaptation
  - adaptation overhead not suitable for all cases
  - interacting multiresolution objects ignored