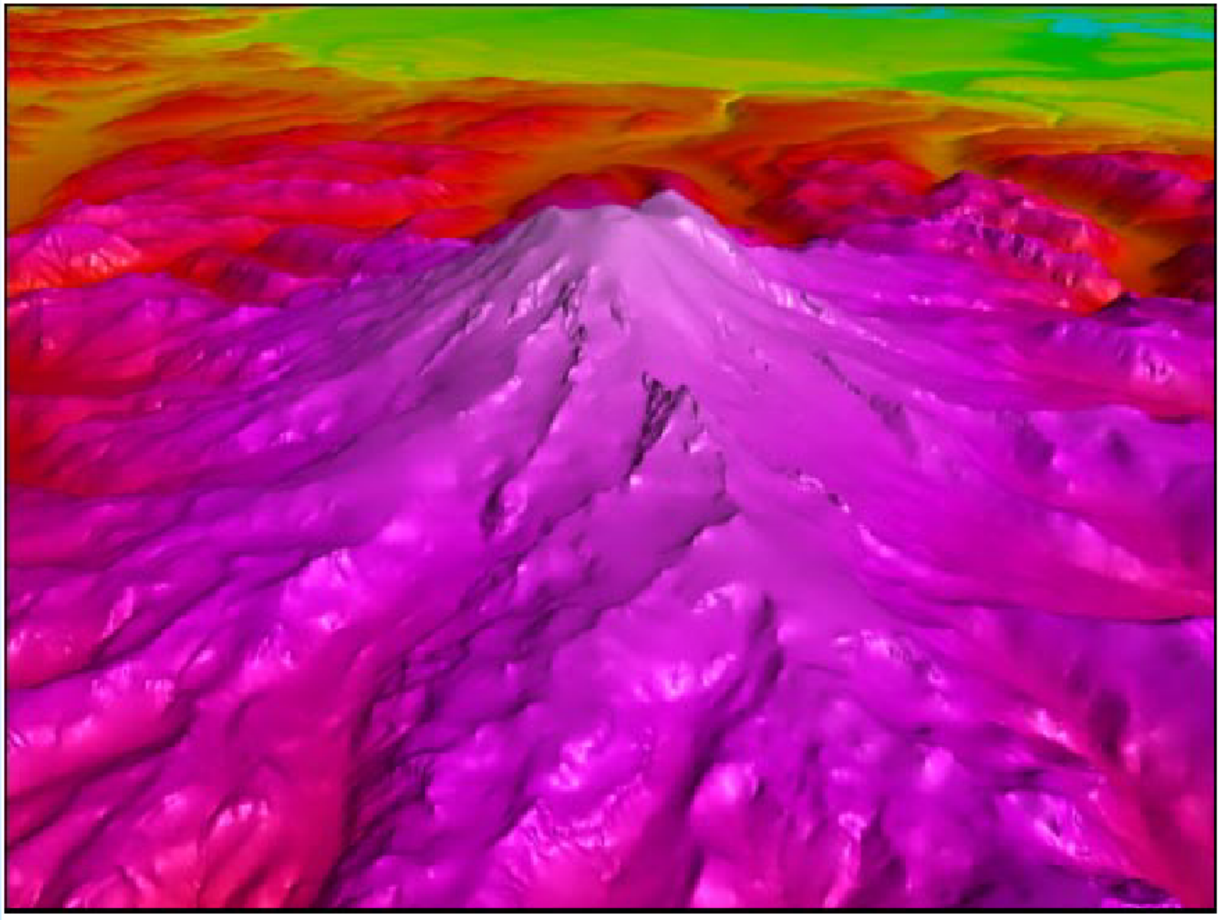
Visualization of Large Terrains Made Easy

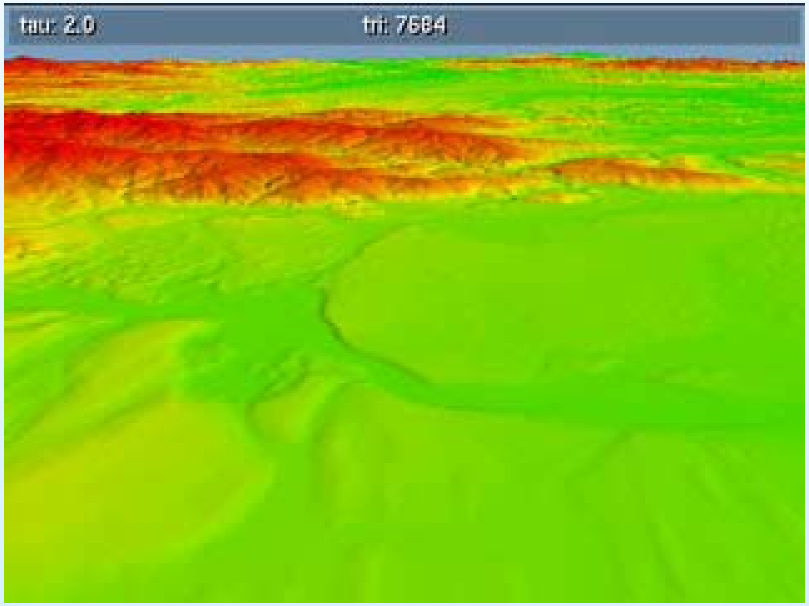
Peter Lindstrom and **Valerio Pascucci**

Lawrence Livermore National Laboratory



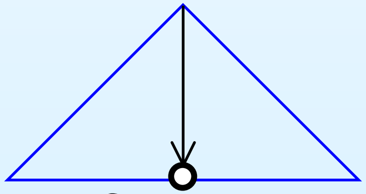
Hierarchical error + static data layout = large terrain visualization made easy.

- **Hierarchical error computation:**
  - Independent of error metric
  - Combined view culling
  - Near optimality
  - Asynchronous updates
- **Hierarchical indexing:**
  - Static data layout
  - Generic paging system
- **Simple:**
  - No explicit hierarchy
  - No priority queue
  - No specialized I/O system
  - No mesh data-structure (implicit stripping)

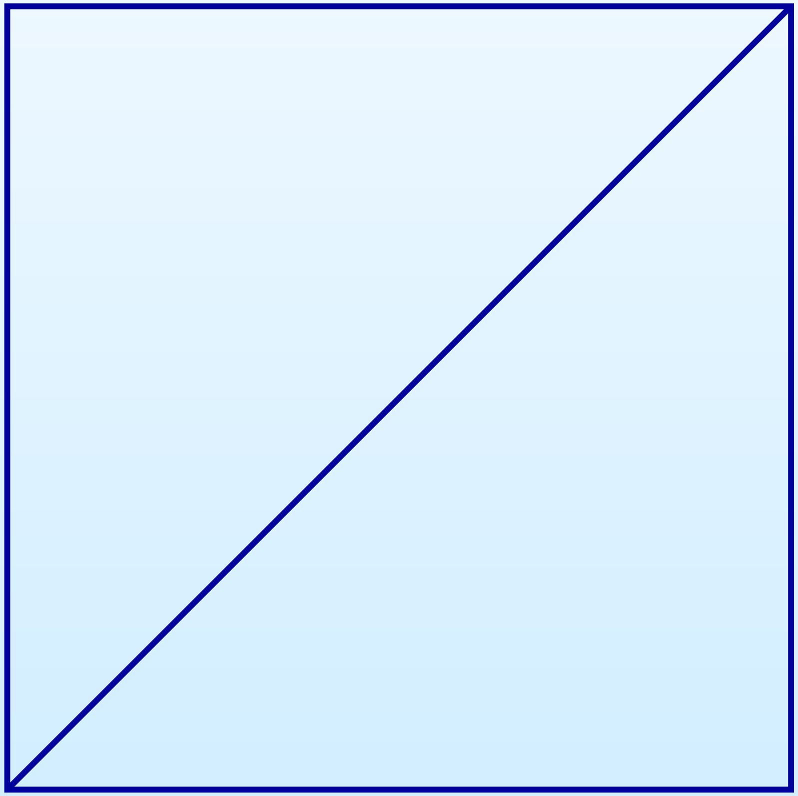


Uniform refinement of the rectilinear grid using longest edge bisection.

Level 0 (base mesh)

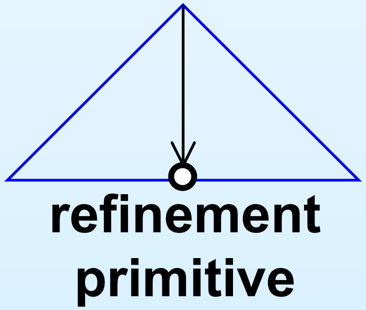


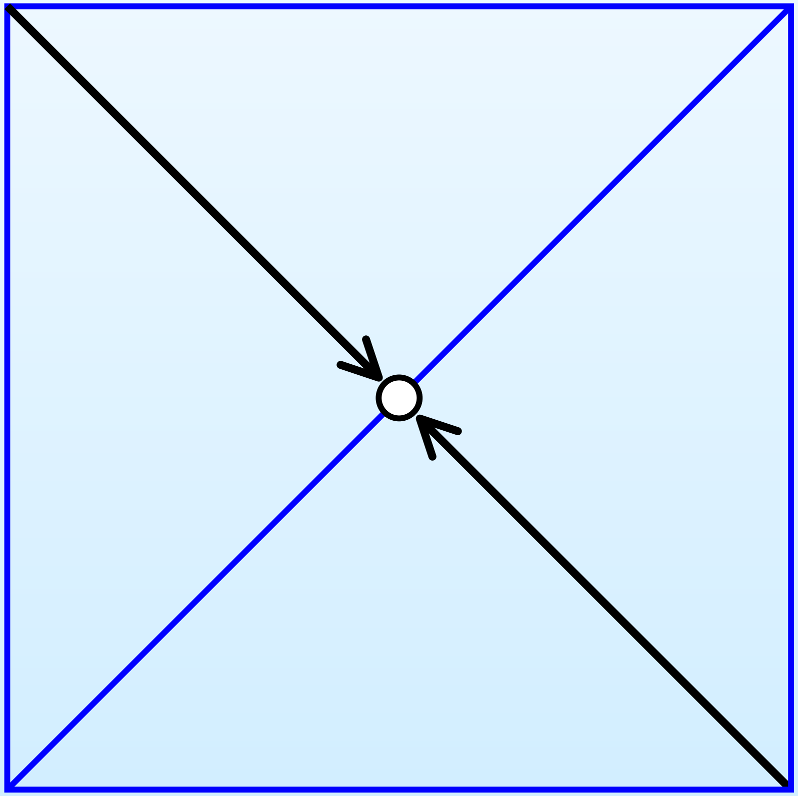
refinement primitive



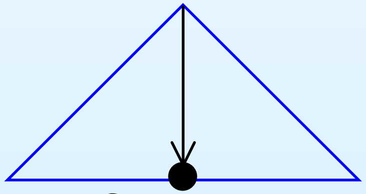
Uniform refinement of the rectilinear grid using longest edge bisection.

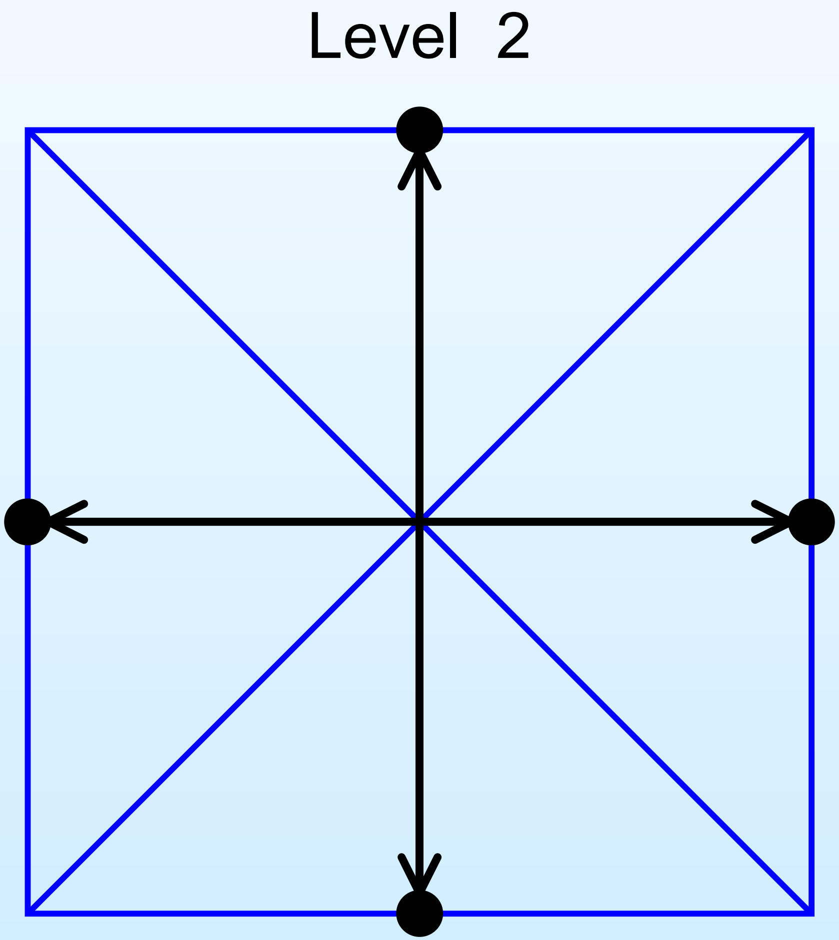
Level 1



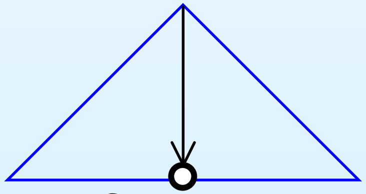


Uniform refinement of the rectilinear grid using longest edge bisection.

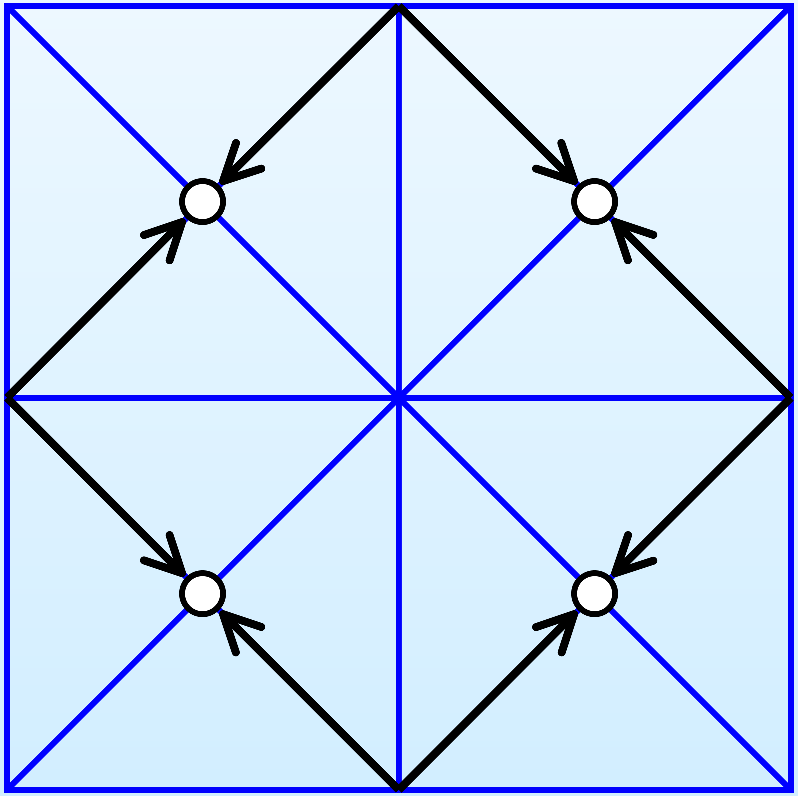




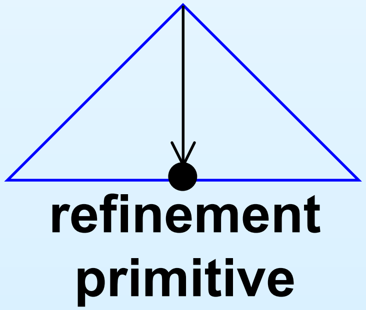
Uniform refinement of the rectilinear grid using longest edge bisection.

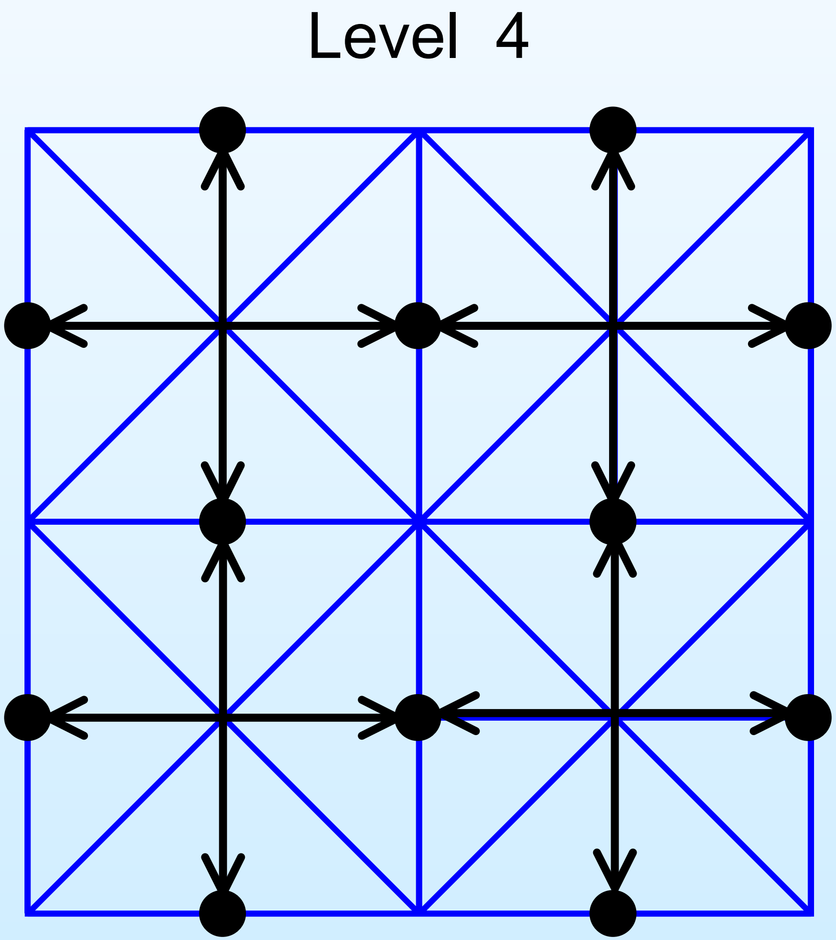


Level 3



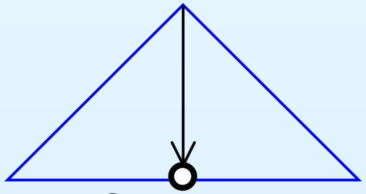
Uniform refinement of the rectilinear grid using longest edge bisection.



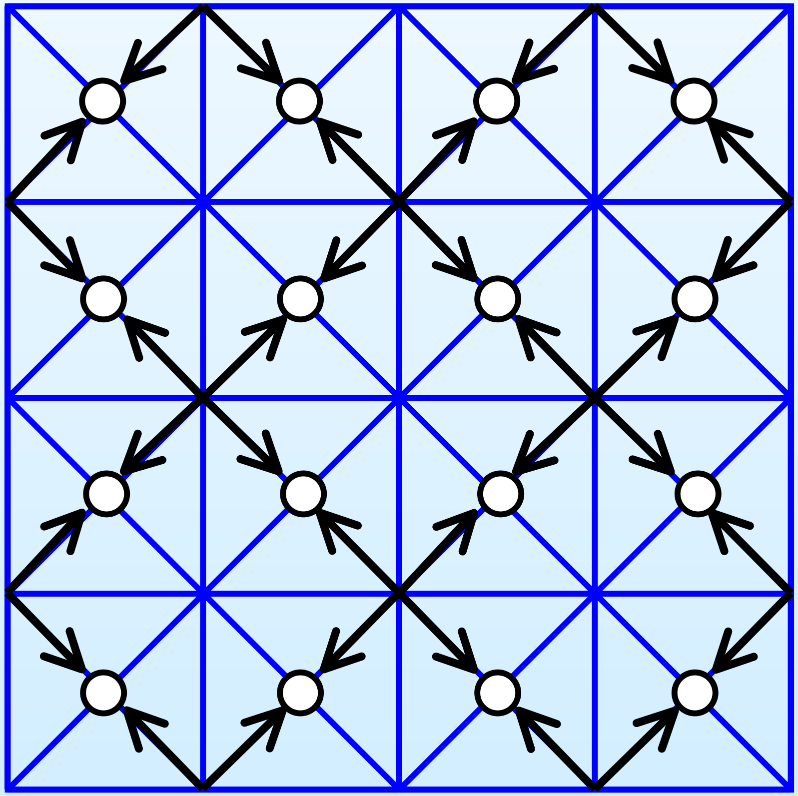


Uniform refinement of the rectilinear grid using longest edge bisection.

Level 5

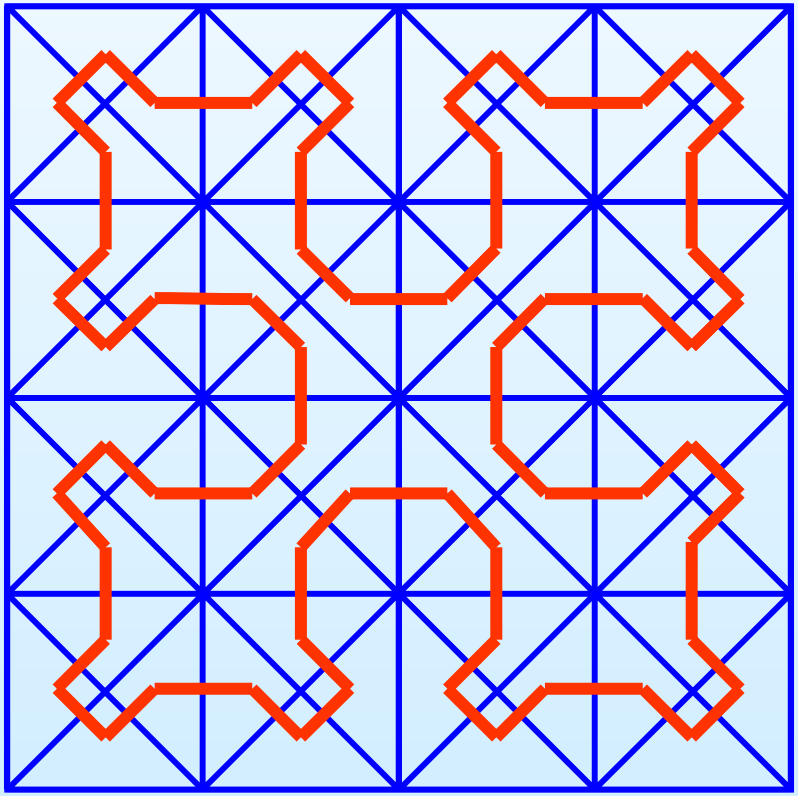


refinement primitive



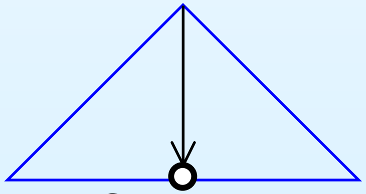
The stripping of any uniform refinement is the Sierpinski space filling curve.

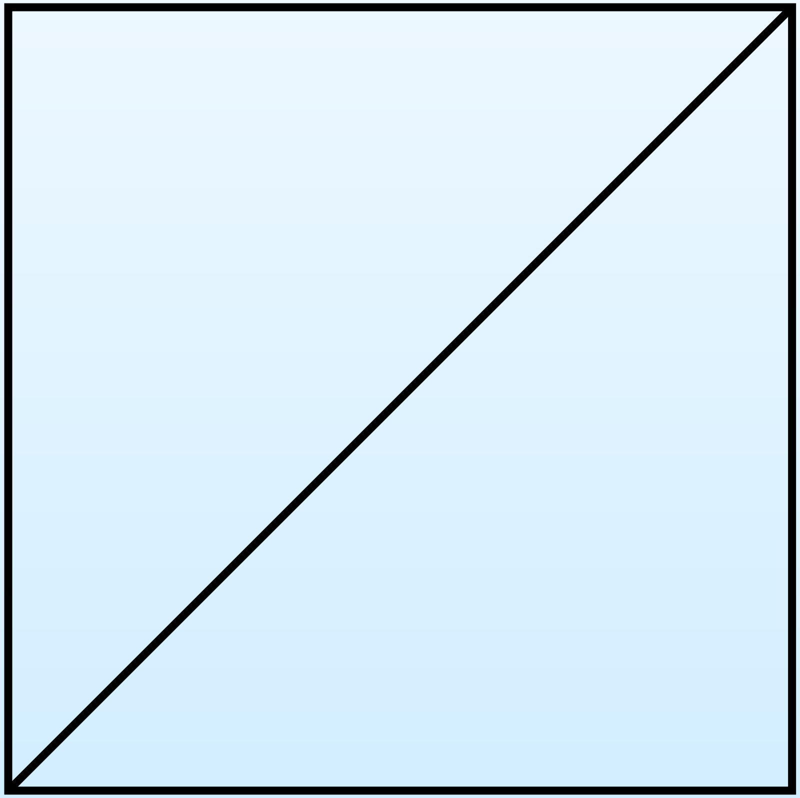
Level 5



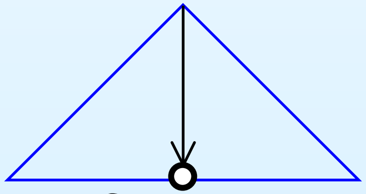
Adaptive refinement of the rectilinear edge bisection.

Level 0 (base mesh)

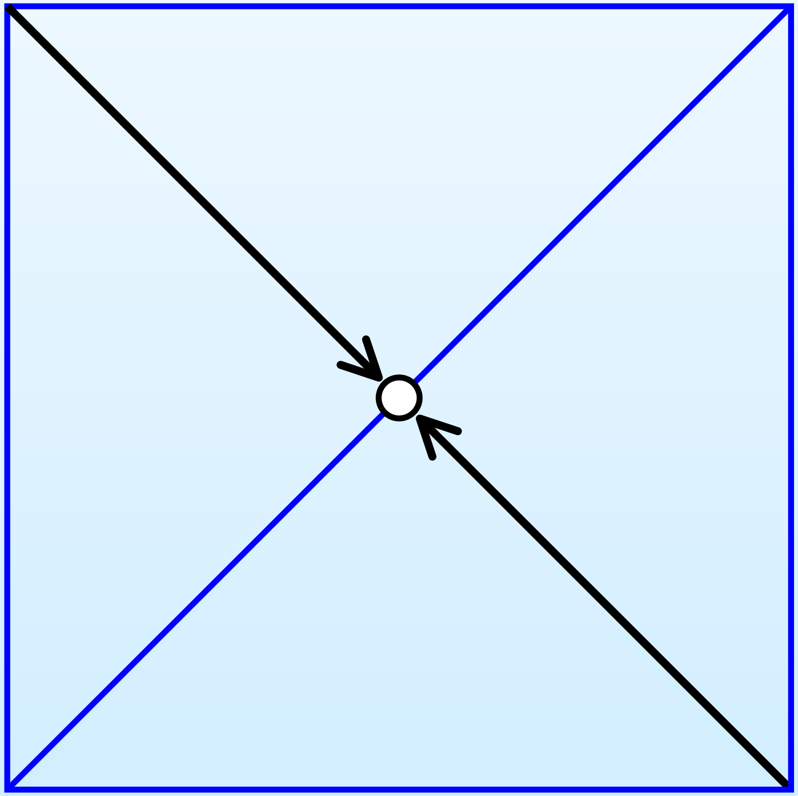




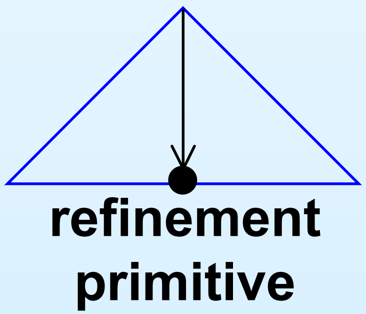
Adaptive refinement of the rectilinear edge bisection.

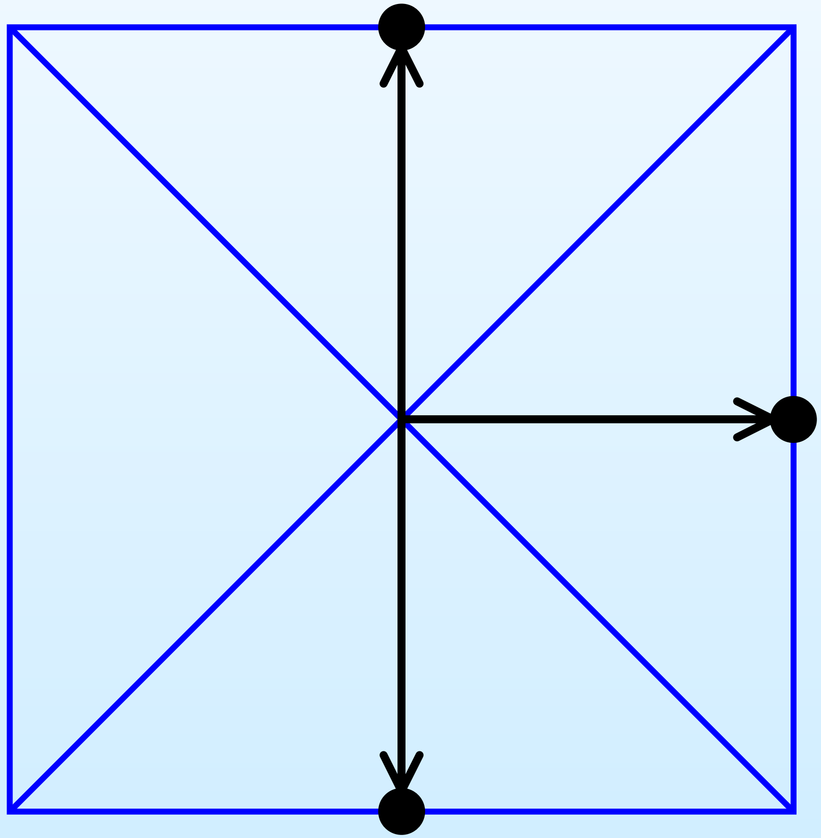


refinement primitive

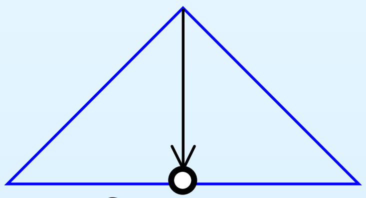


Adaptive refinement of the rectilinear edge bisection.

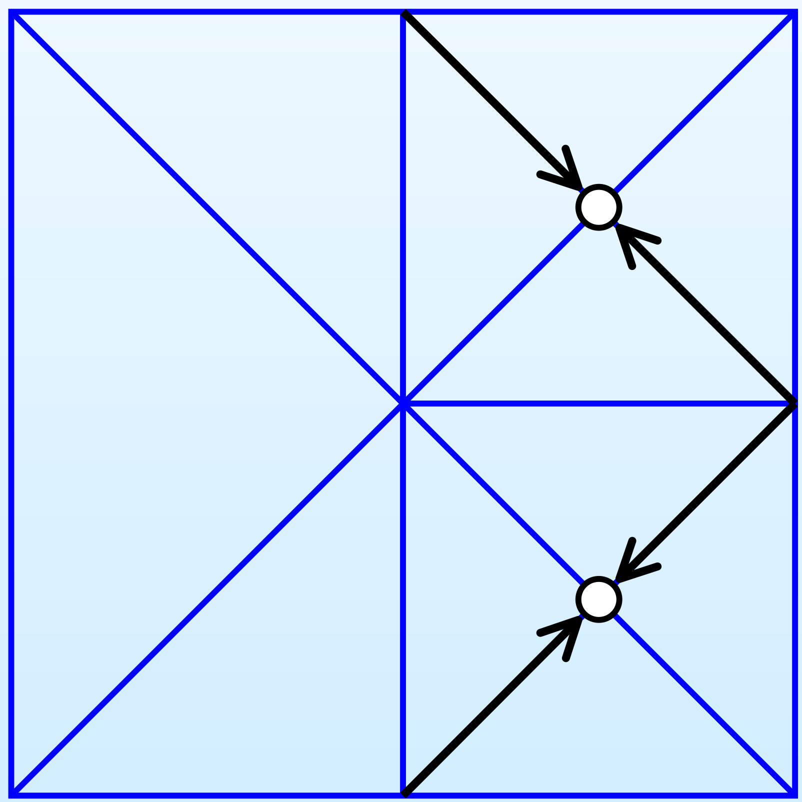




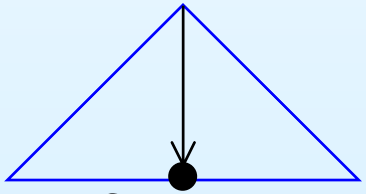
Adaptive refinement of the rectilinear edge bisection.

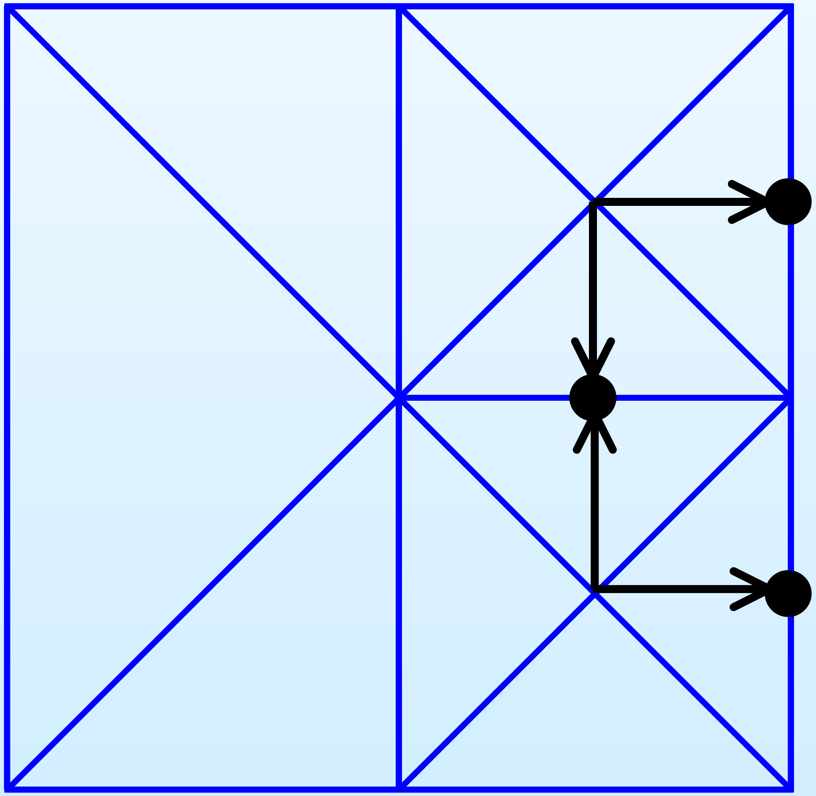


refinement
primitive

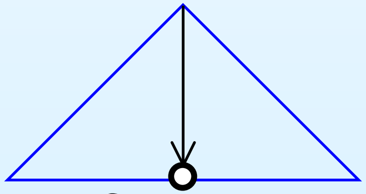


Adaptive refinement of the rectilinear edge bisection.

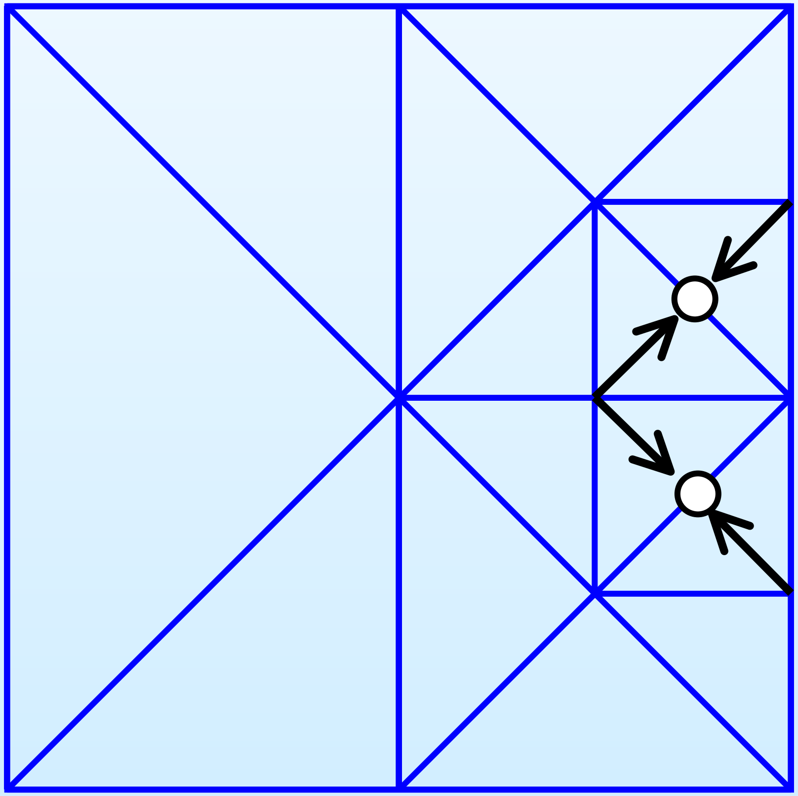




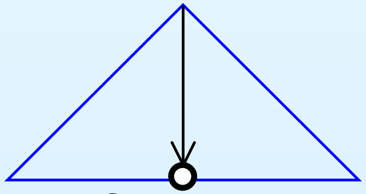
Adaptive refinement of the rectilinear edge bisection.



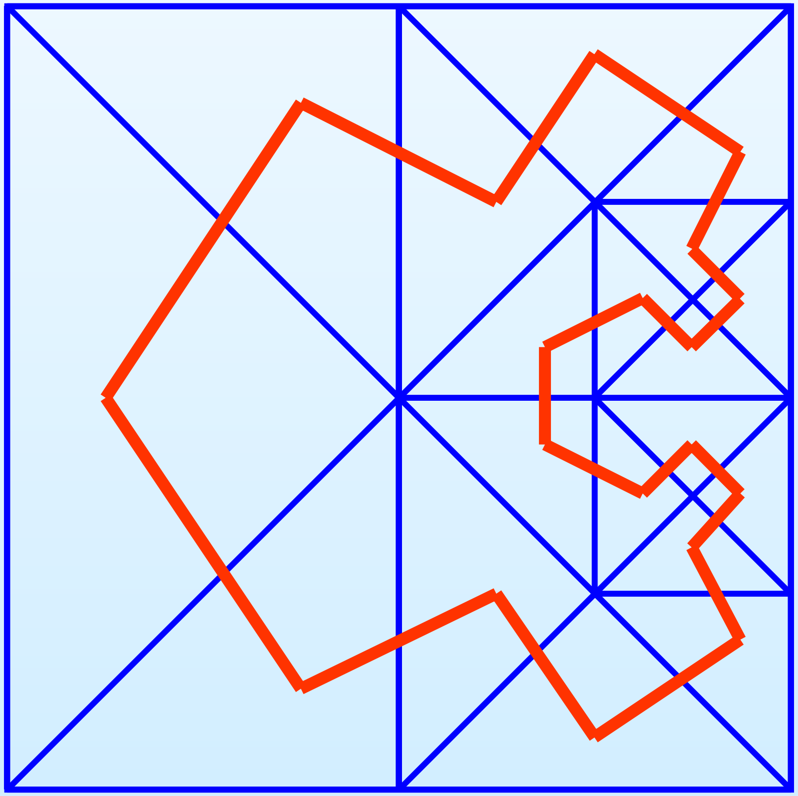
refinement primitive



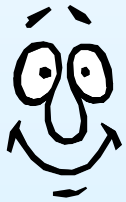
Adaptive refinement of the rectilinear edge bisection.



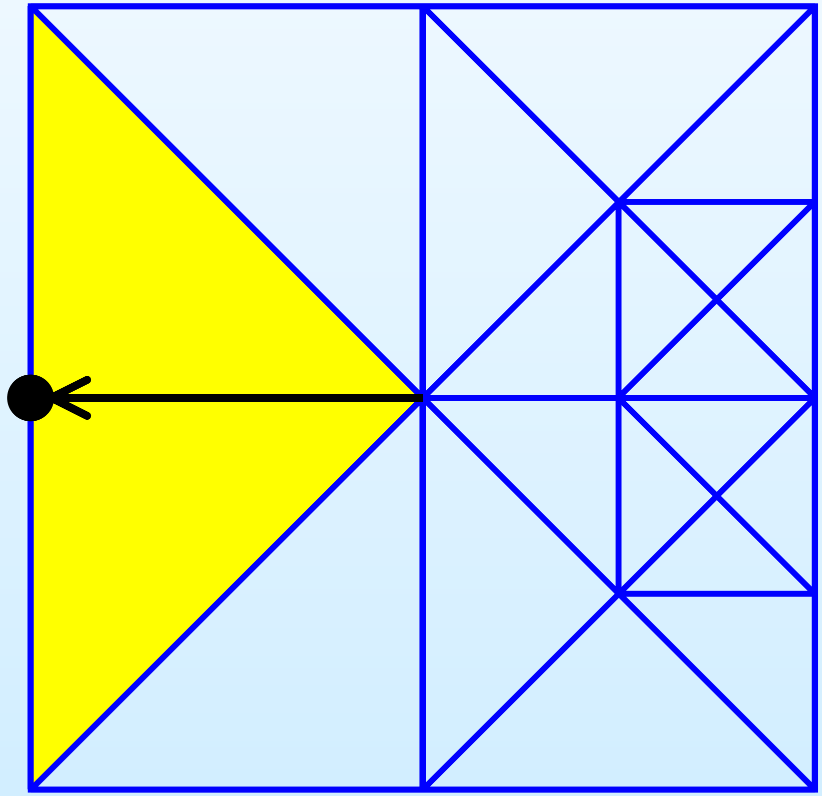
refinement
primitive



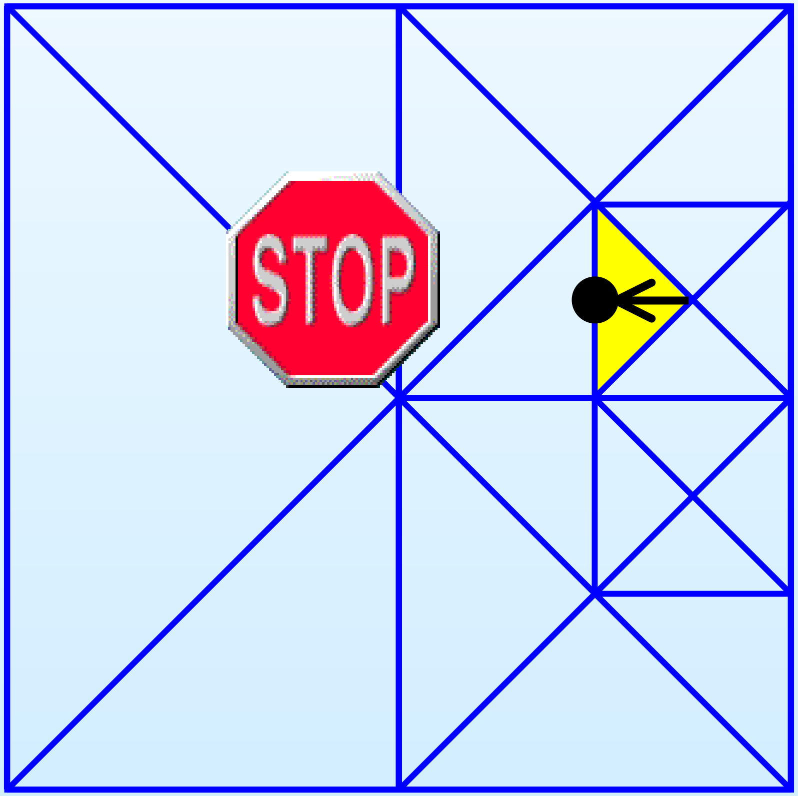
The dynamic update of an adaptive mesh may be very easy and efficient.

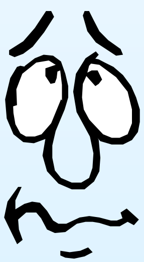


Do not rebuild.
Just refine a single triangle



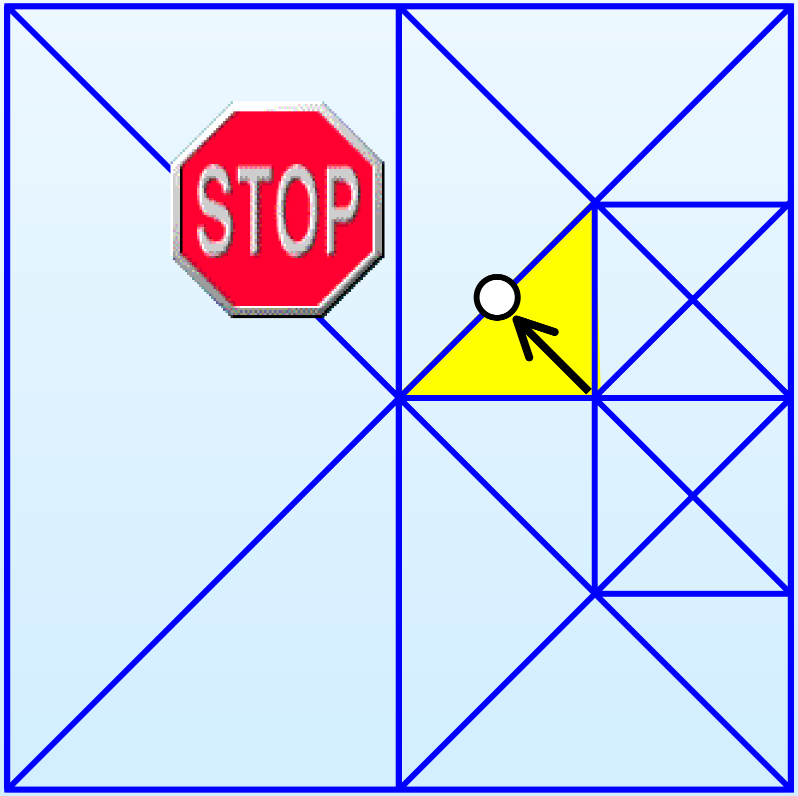
The dynamic update of an adaptive mesh may be NOT so easy and efficient.





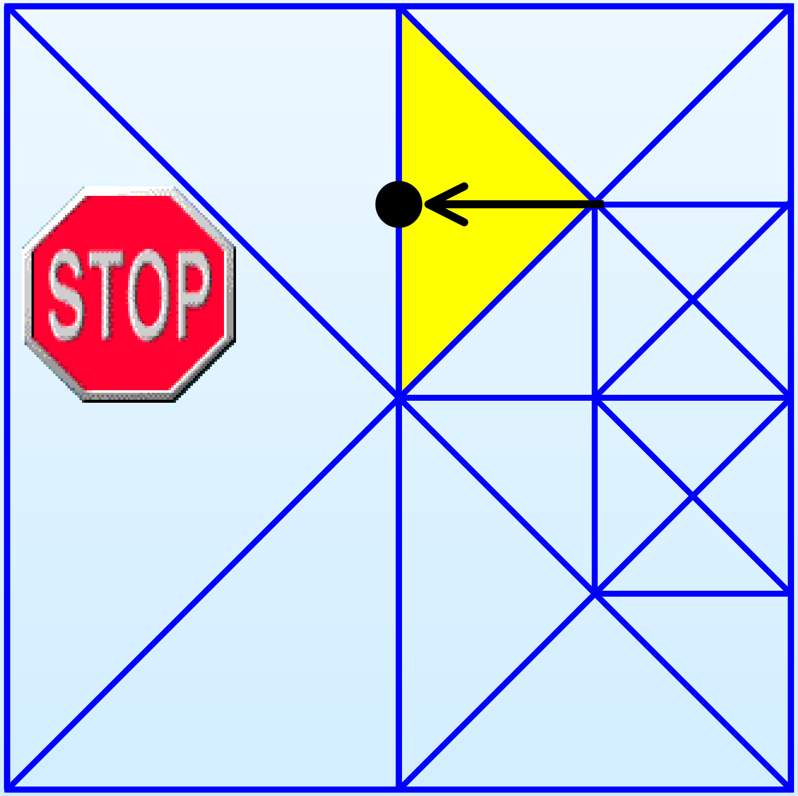
One refinement may trigger ripple effect

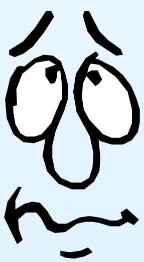
The dynamic update of an adaptive mesh may be NOT so easy and efficient.



One refinement may trigger ripple effect

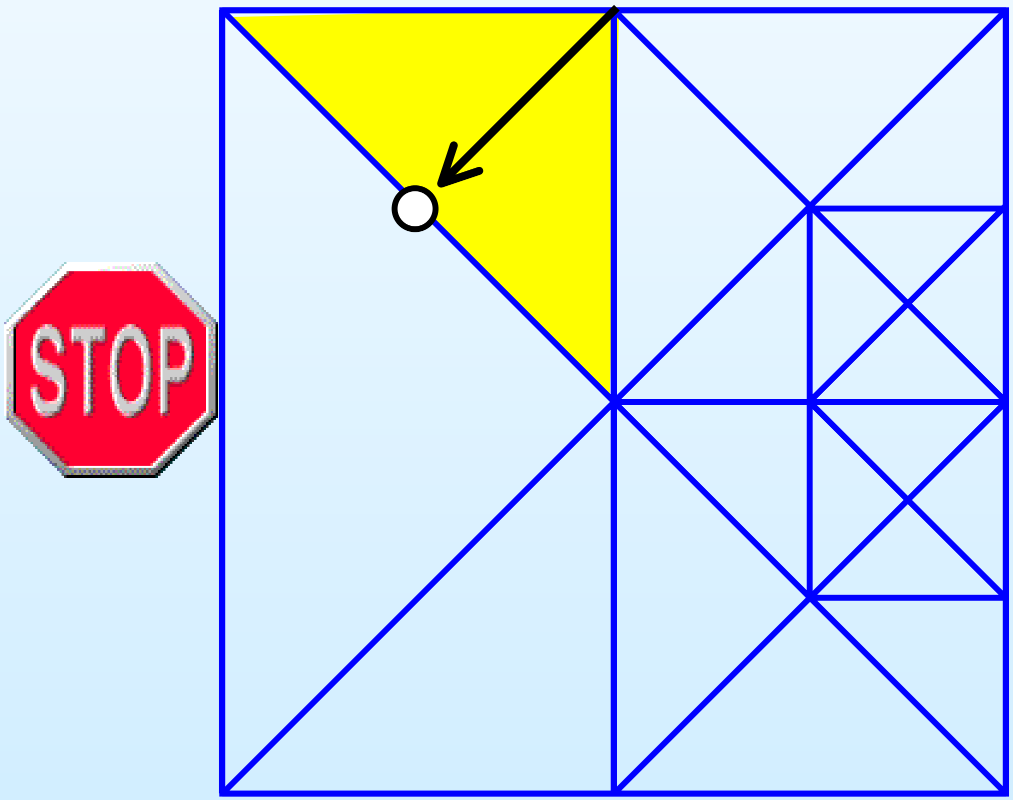
The dynamic update of an adaptive mesh may be NOT so easy and efficient.





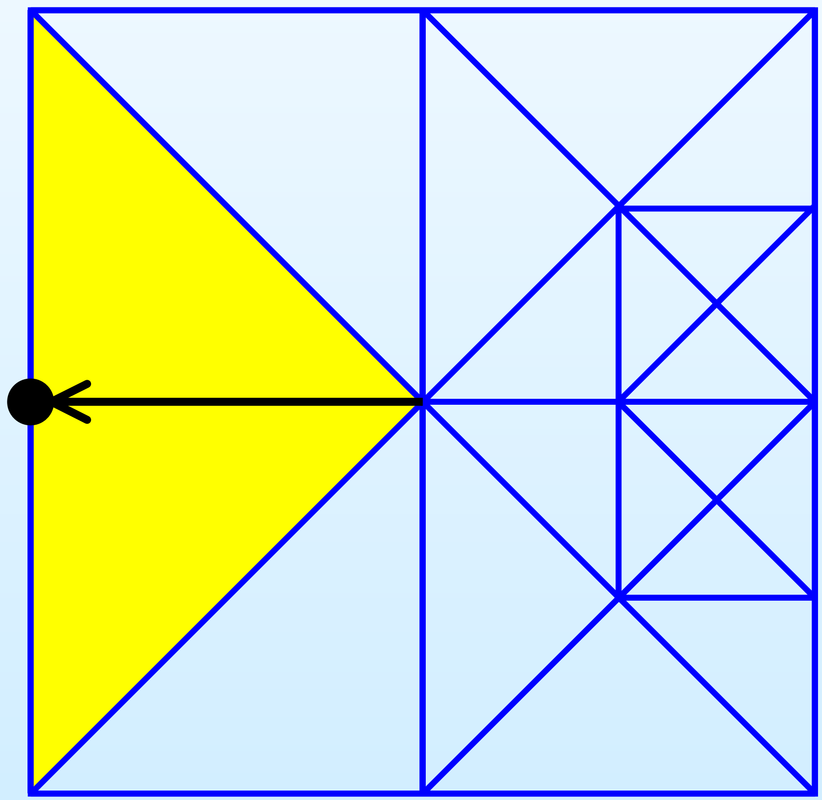
One refinement may trigger ripple effect

The dynamic update of an adaptive mesh may be NOT so easy and efficient.



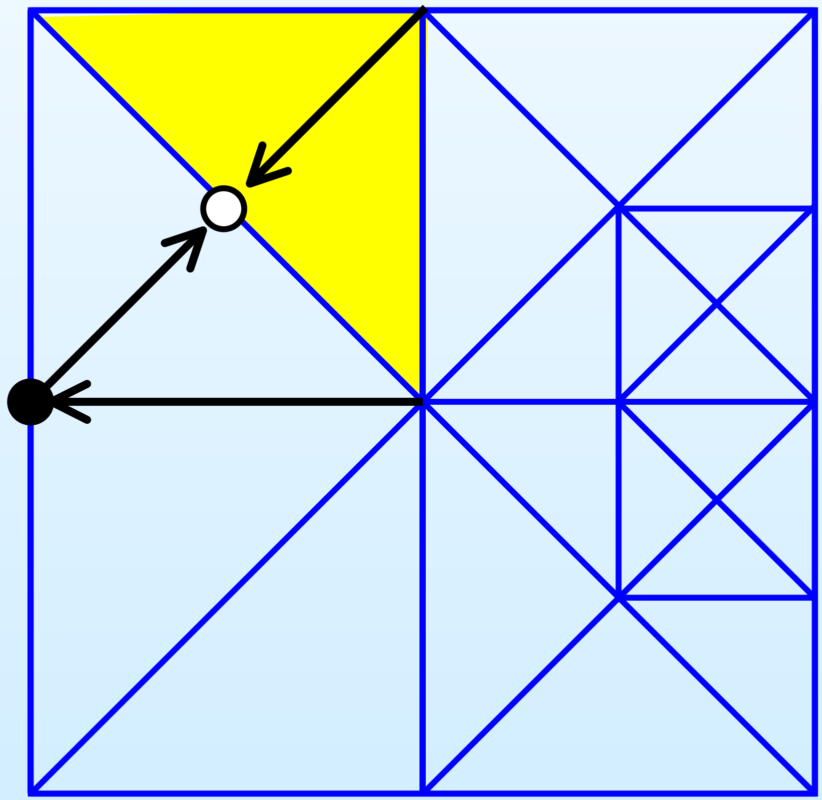
One refinement may trigger ripple effect

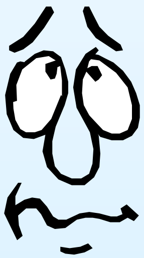
The dynamic update of an adaptive mesh may be NOT so easy and efficient.



One refinement may trigger ripple effect

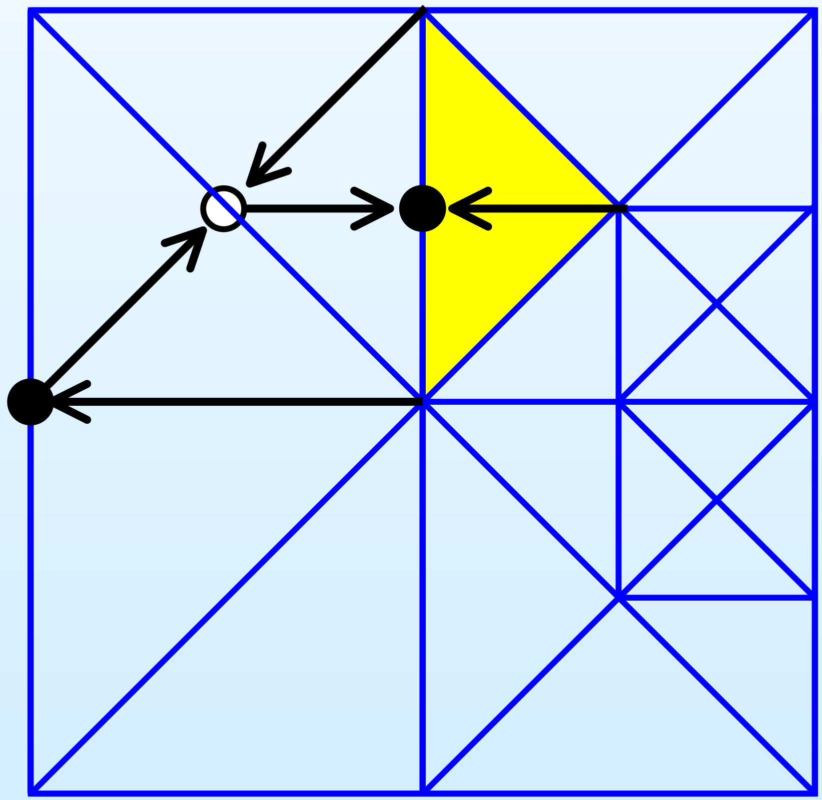
The dynamic update of an adaptive mesh may be NOT so easy and efficient.





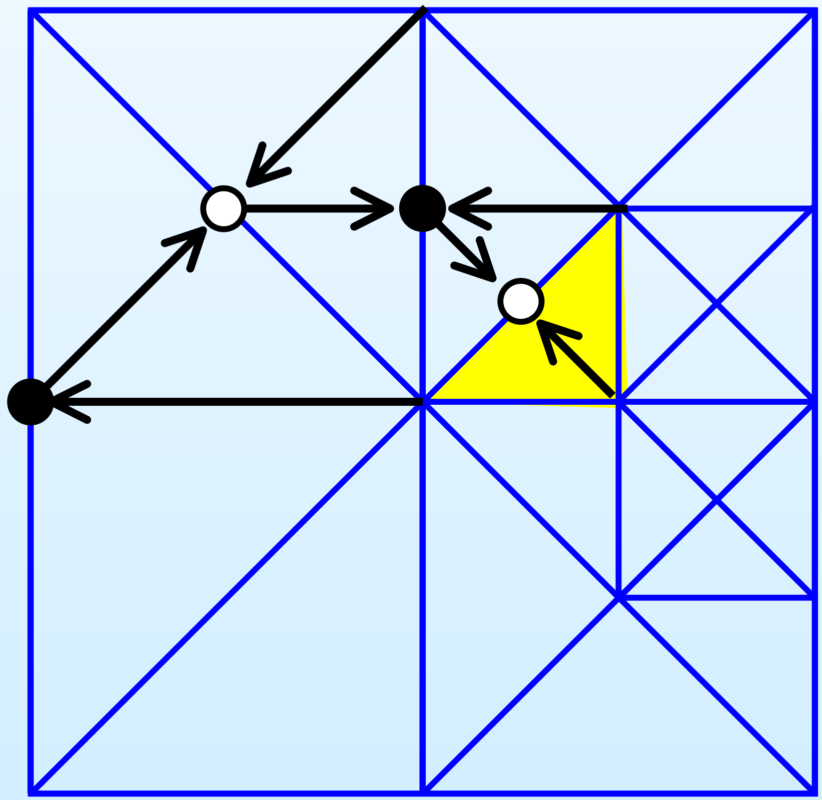
One refinement may trigger ripple effect

The dynamic update of an adaptive mesh may be NOT so easy and efficient.



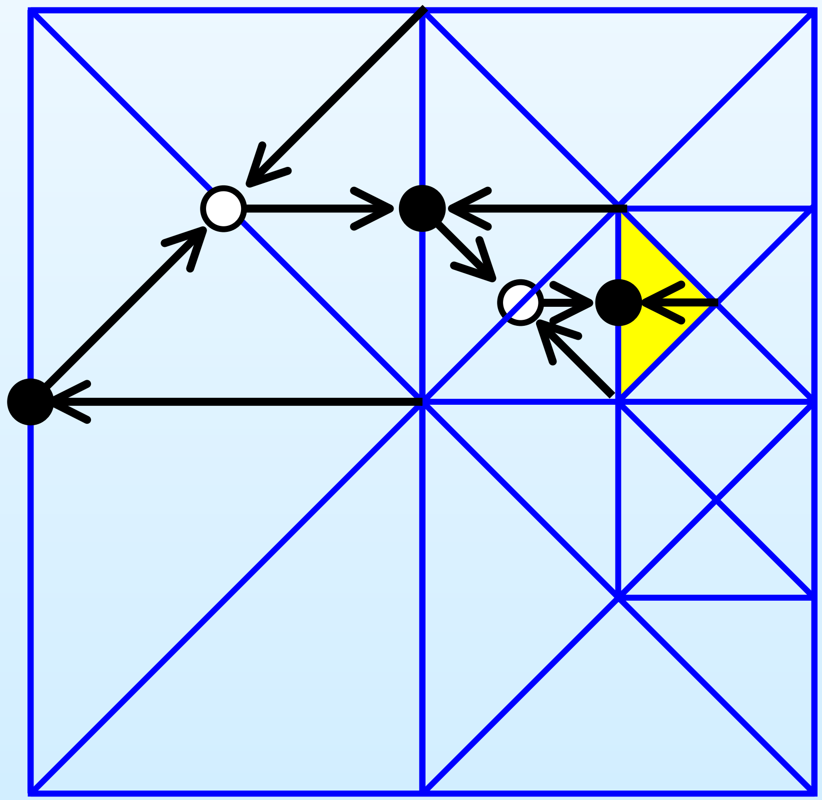
One refinement may trigger ripple effect

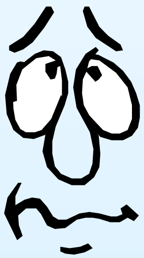
The dynamic update of an adaptive mesh may be NOT so easy and efficient.



One refinement may trigger ripple effect

The dynamic update of an adaptive mesh may be NOT so easy and efficient.

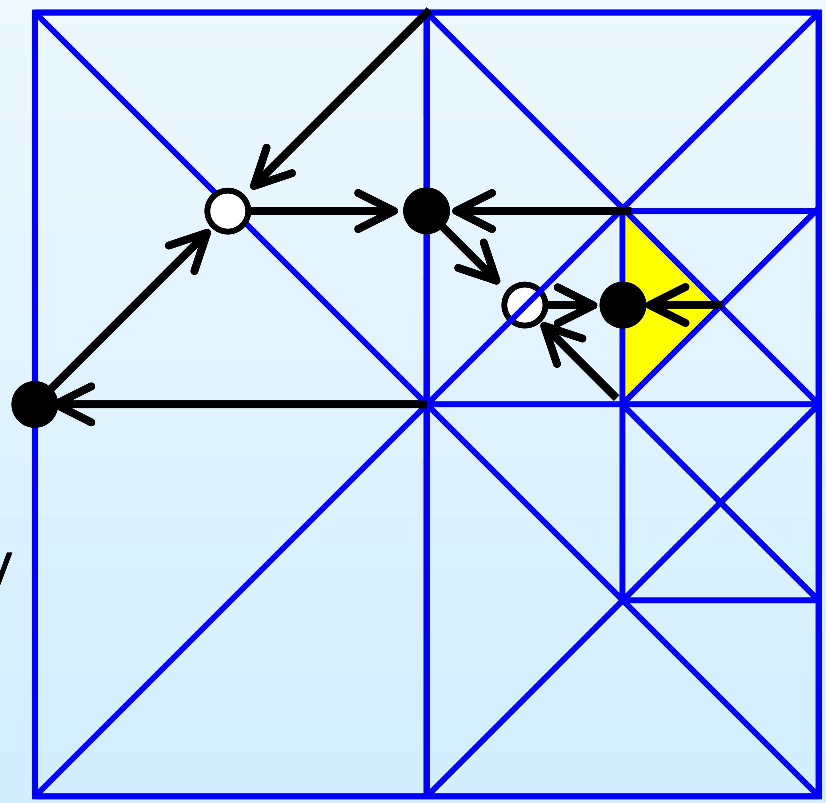




One refinement may trigger ripple effect

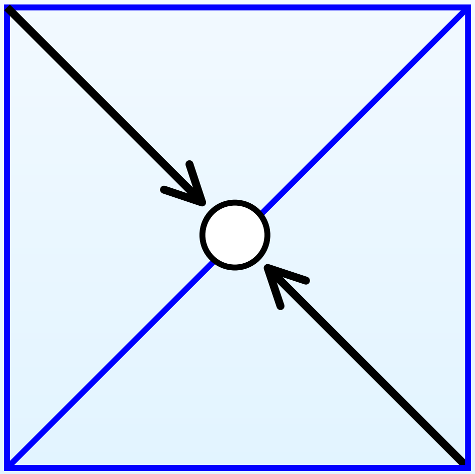
The dynamic update of an adaptive mesh may be NOT so easy and efficient.

Must maintain:
- mesh topology
- priority queue

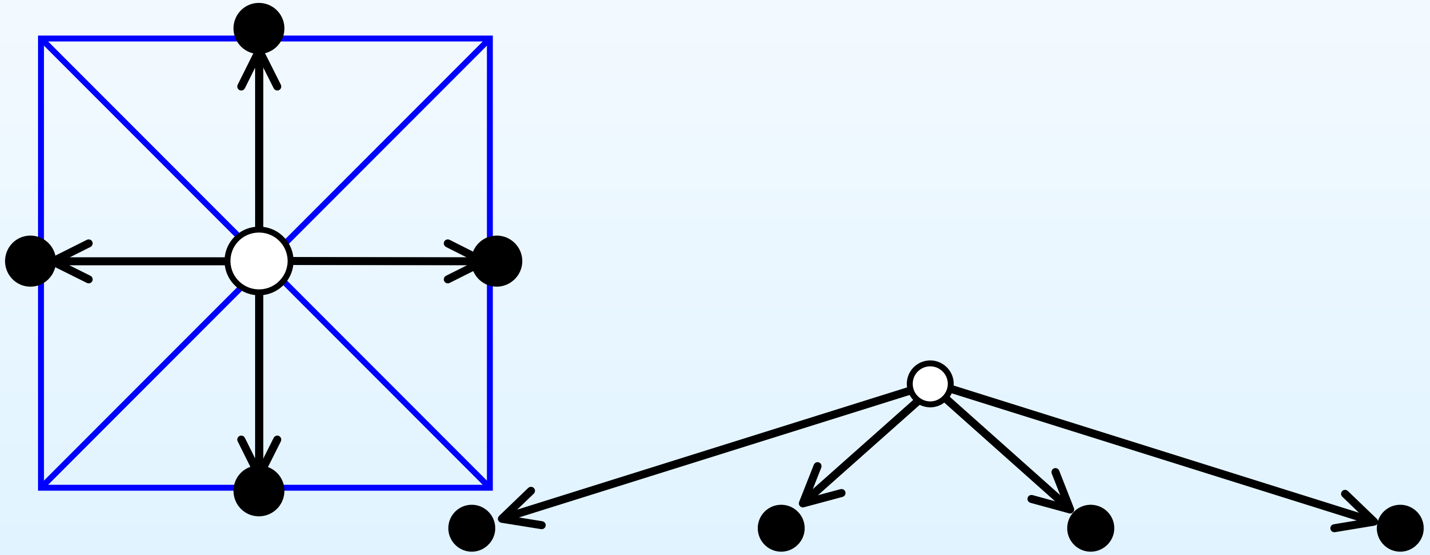


One refinement may trigger ripple effect

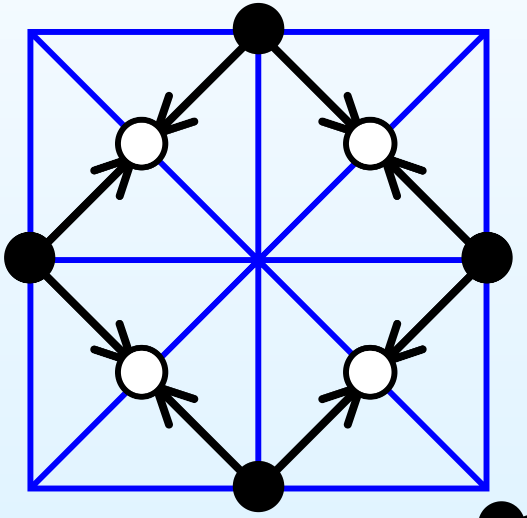
The rectilinear edge bisection can be defined in terms of a vertex hierarchy.

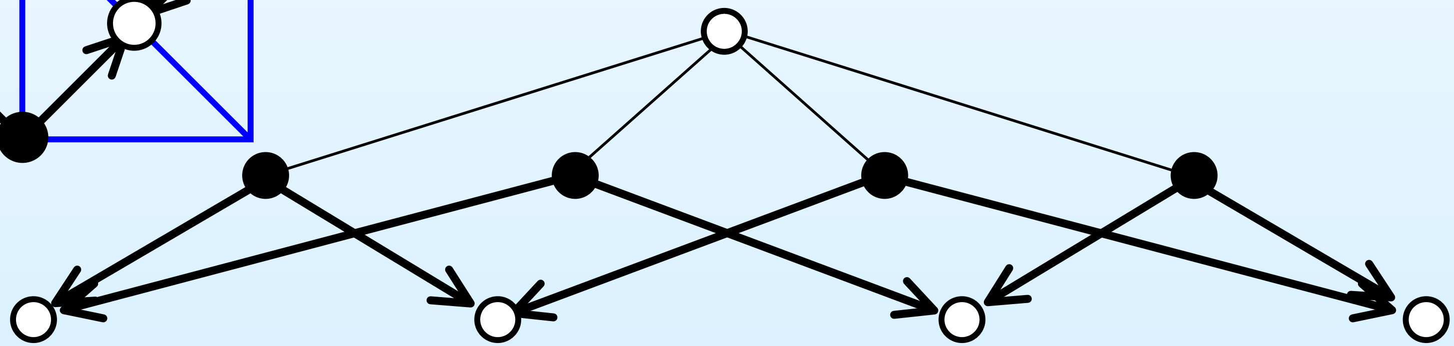


The rectilinear edge bisection can be defined in terms of a vertex hierarchy.

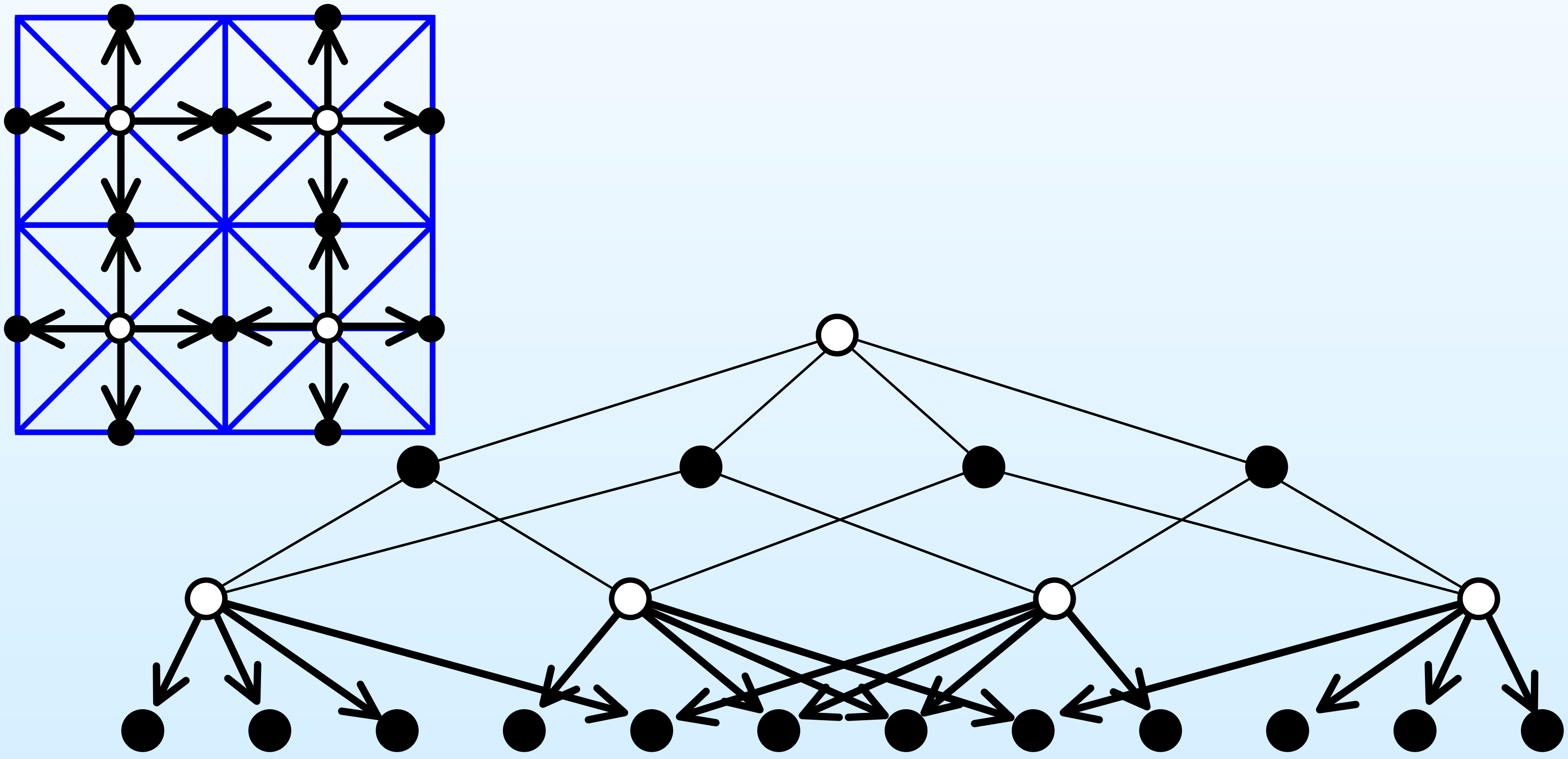


The rectilinear edge bisection can be defined in terms of a vertex hierarchy.

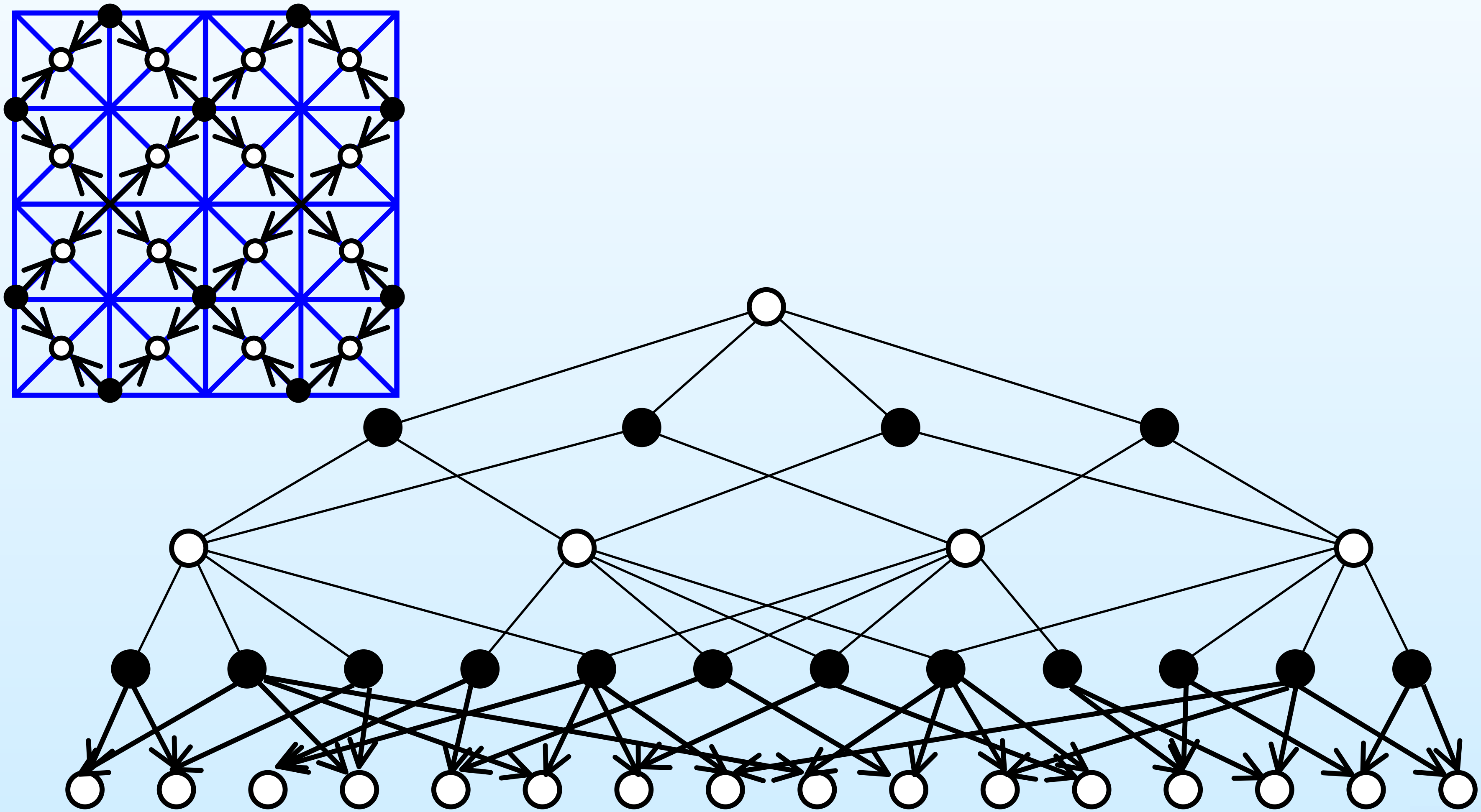




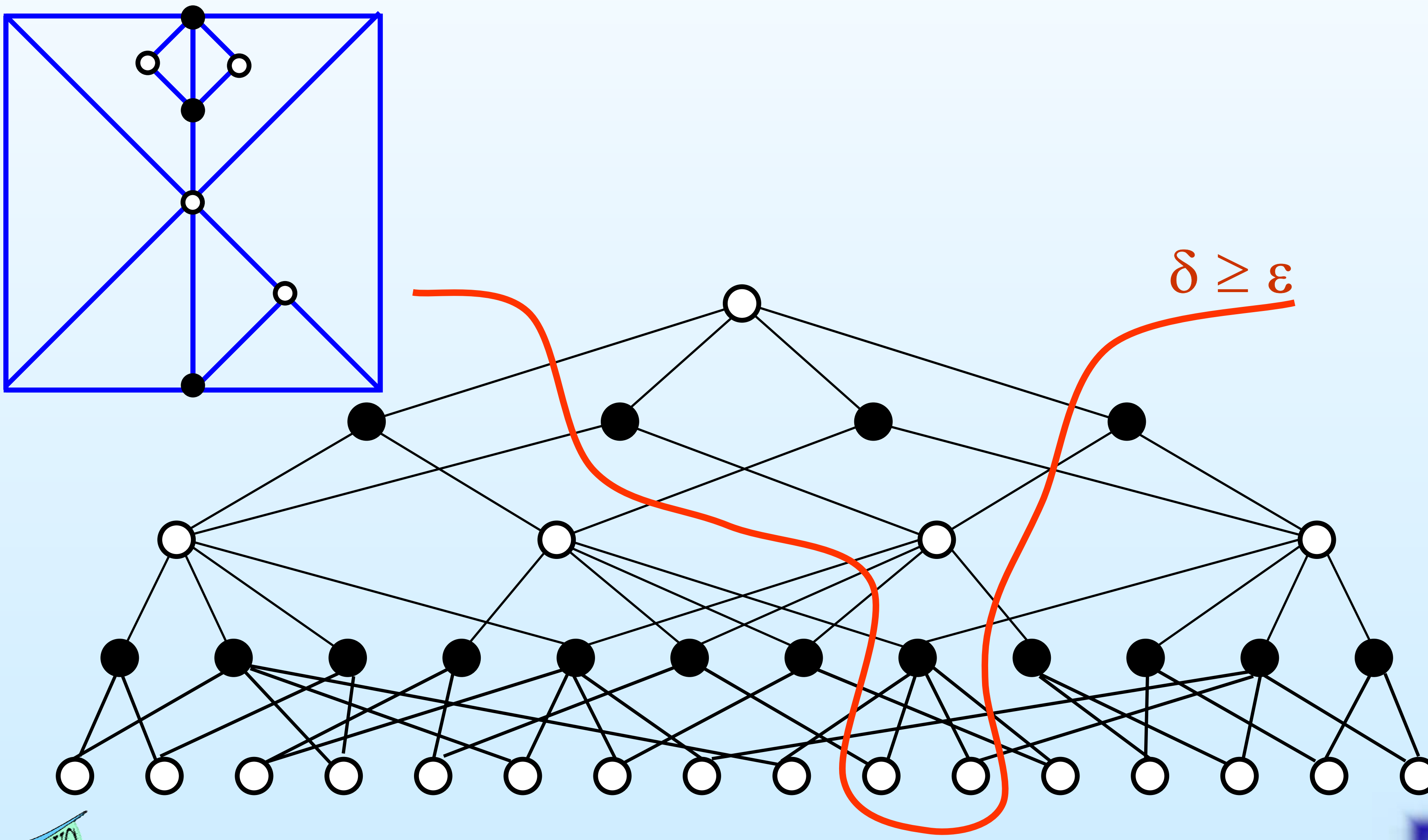
The rectilinear edge bisection can be defined in terms of a vertex hierarchy.



The rectilinear edge bisection can be defined in terms of a vertex hierarchy.



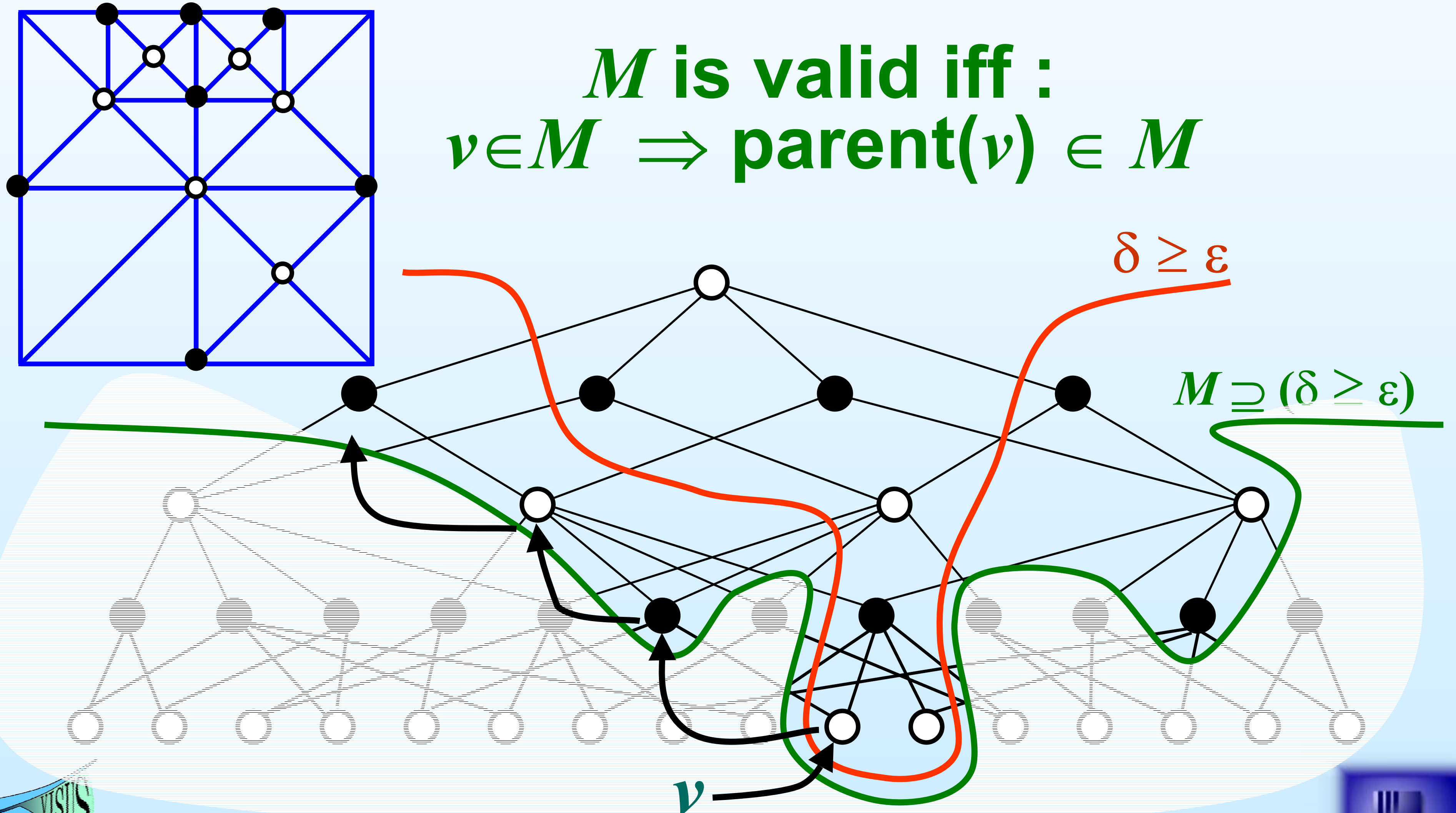
Selecting nodes on the basis of the error alone does not guarantee a valid mesh *M*.



Selecting nodes on the basis of the error alone does not guarantee a valid mesh *M*.

M is valid iff :

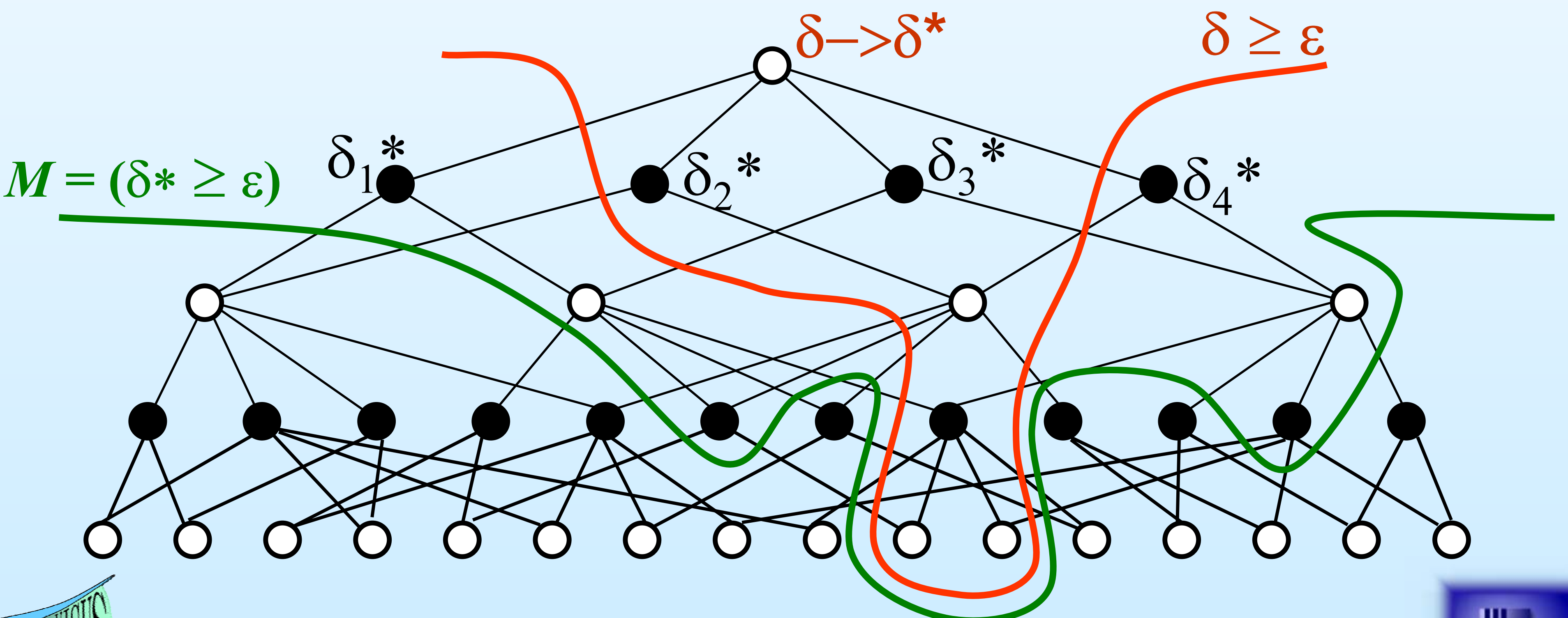
$$v \in M \Rightarrow \text{parent}(v) \in M$$



A hierarchical error metric overcomes the dichotomy of error vs. consistency.

We inflate the geometric error from δ to δ*

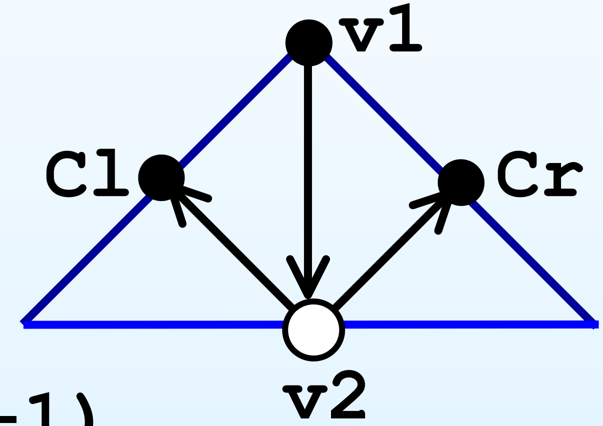
$$\delta^* = \max\left\{ \delta, \delta_1^*, \ldots, \delta_4^* \right\}$$



A hierarchical error metric simplifies the mesh construction and stripping.

vertex buffer ↓ level ↓

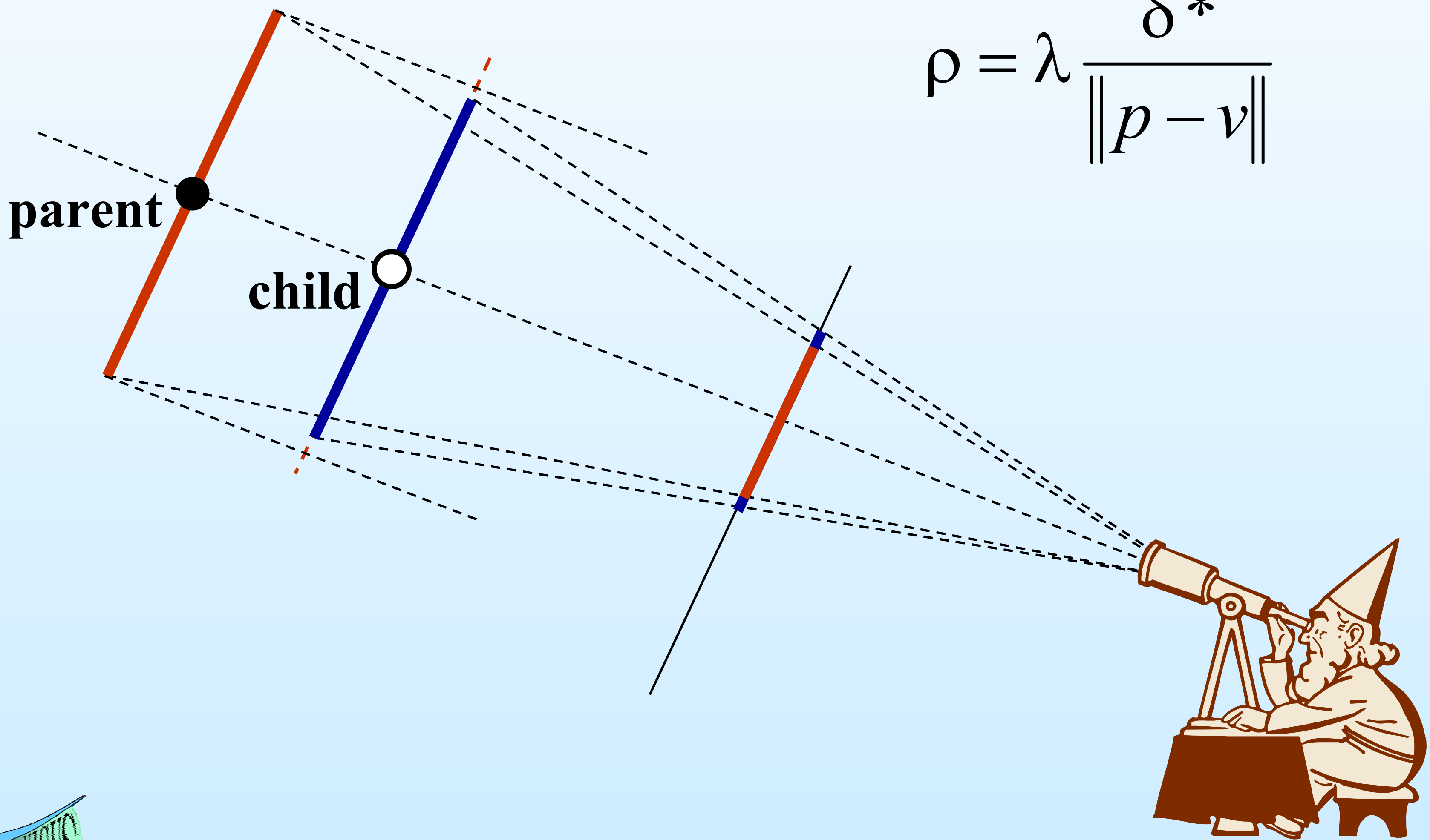
```
mesh-refine(VB,v1,v2,l)
  if l>0 and δ*(v1)≥ε then
    mesh-refine (VB,v2,Cl,l-1)
    strip-append(VB,v1,l mod 2)
    mesh-refine (VB,v2,Cr,l-1)
```



VB

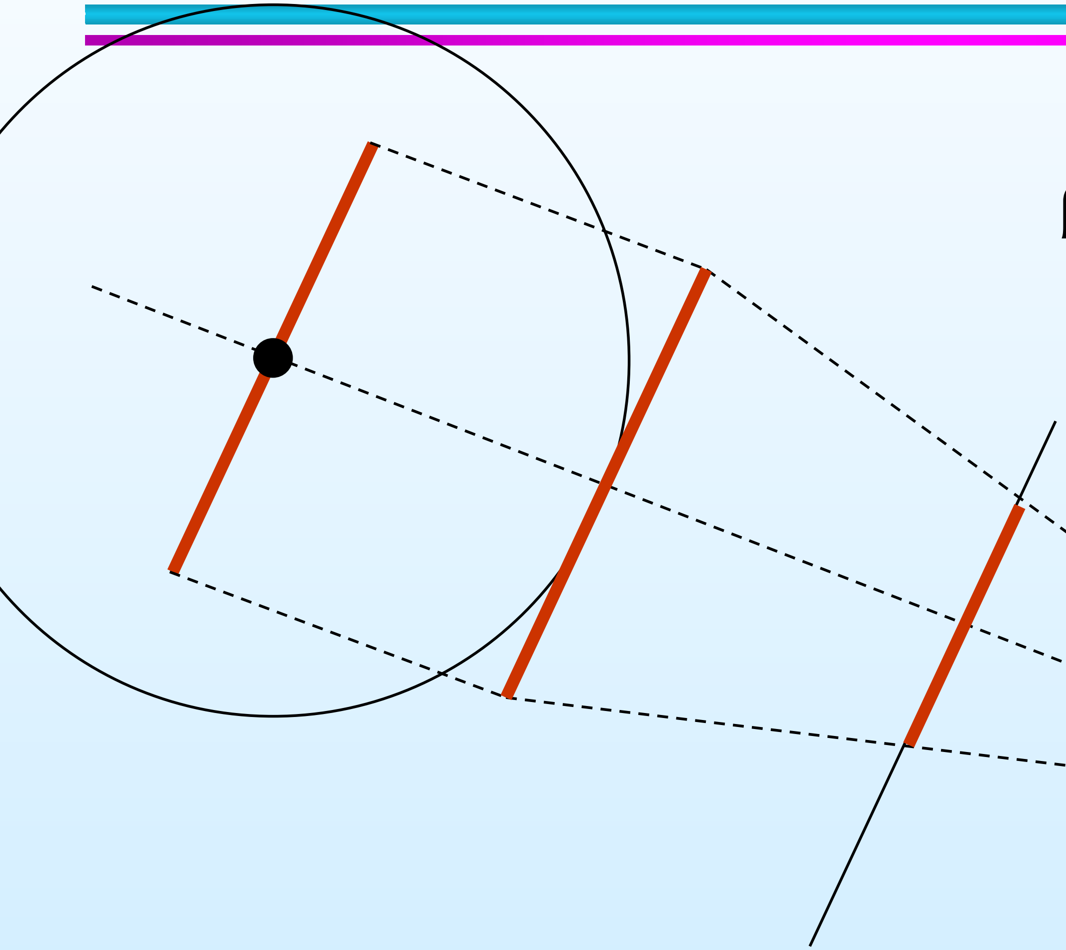
va	vb	vc	vd	ve	vf	vg	vh	vi	v1					
								0	1/0					

The error projection can destroy the nested structure of the metric.



$$\rho = \lambda \frac{\delta^*}{\| p - v \|}$$

We inflate the projected error by replacing points with spheres.

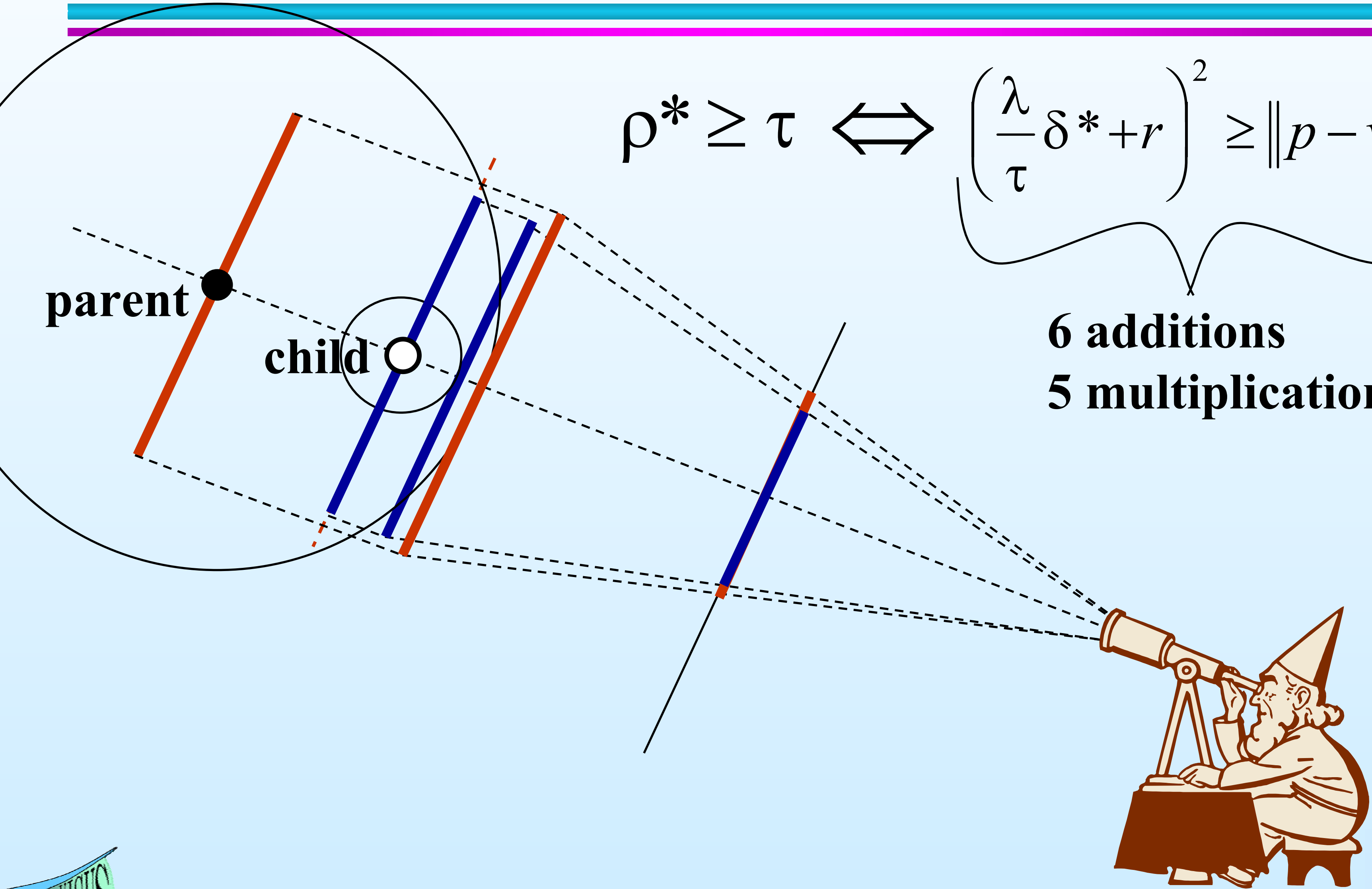


$$\rho^* = \lambda \frac{\delta^*}{\left\| p - v \right\| - r}$$

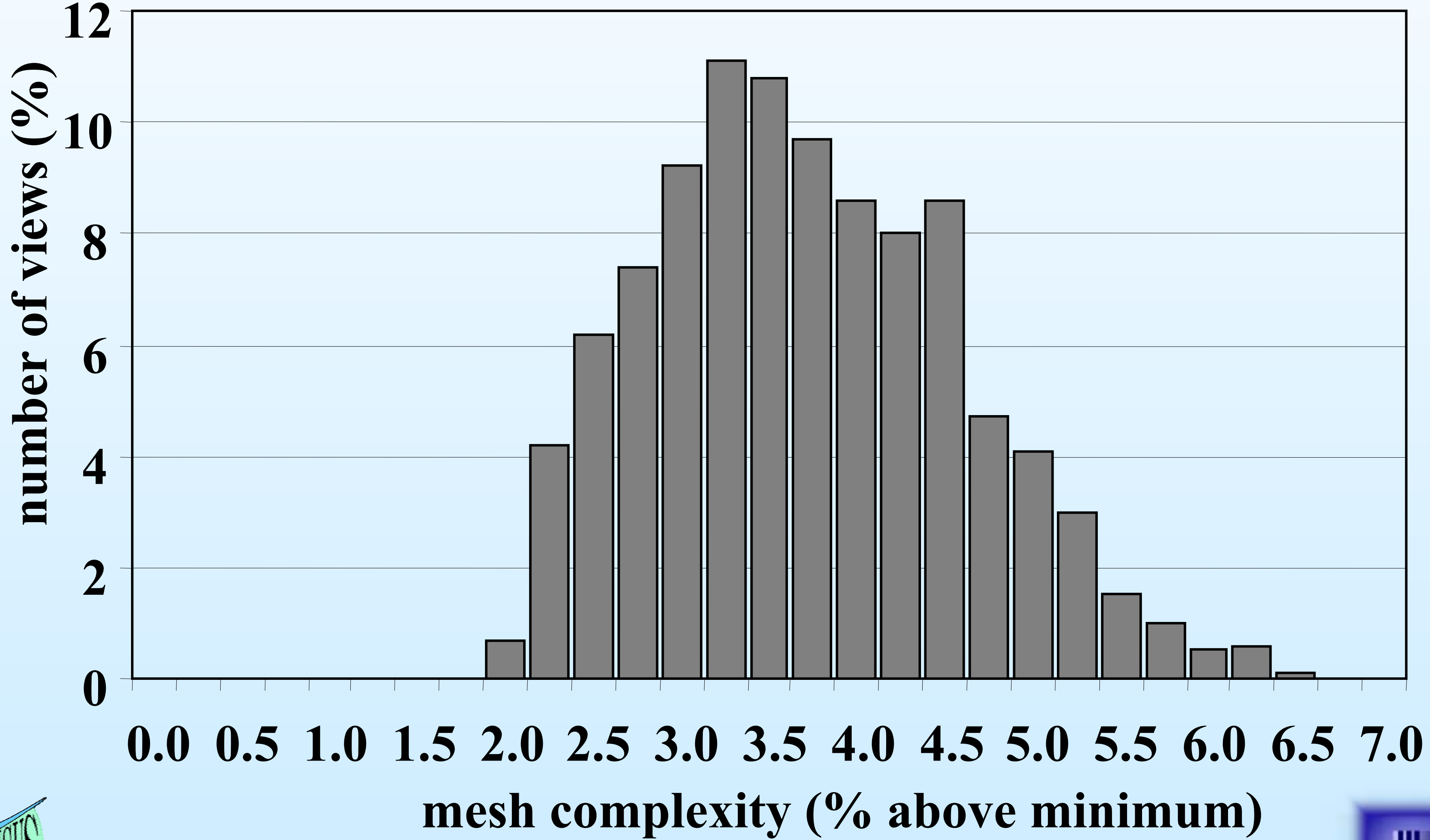
Nested spheres yield a view dependent hierarchical error metric.

$$\rho^* \geq \tau \iff \underbrace{\left(\frac{\lambda}{\tau}\delta^* + r\right)^2 \geq \|p - v\|^2}$$

6 additions
5 multiplications



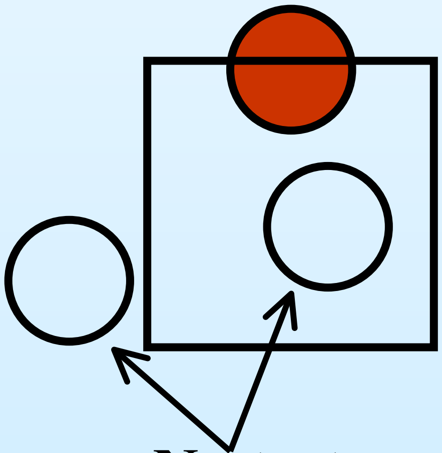
The view dependent hierarchical error metric is not far from the optimal.



Nested spheres allow fast and simple integrated view culling.

- **The culling test is performed only if the sphere of the parent intersects the boundary of the view frustum.**

$\leq$ 6 times dot product and comparison



view frustum

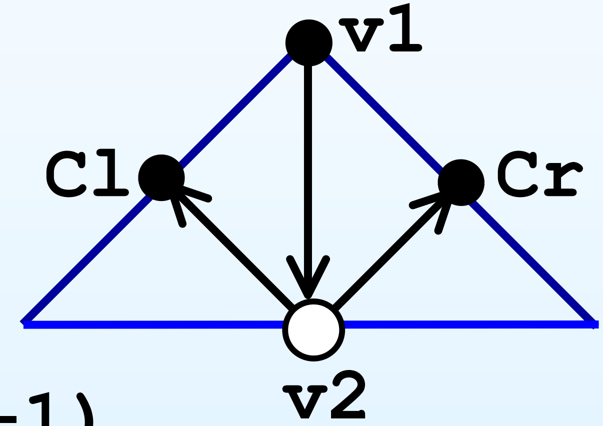
No test on descendants

VISUS

A hierarchical error metric simplifies the mesh construction and stripping.

vertex buffer   level

```
mesh-refine(VB,v1,v2,l)
  if l>0 and ρ*(v1)≥τ then
    mesh-refine (VB,v2,Cl,l-1)
    strip-append(VB,v1,l mod 2)
    mesh-refine (VB,v2,Cr,l-1)
```

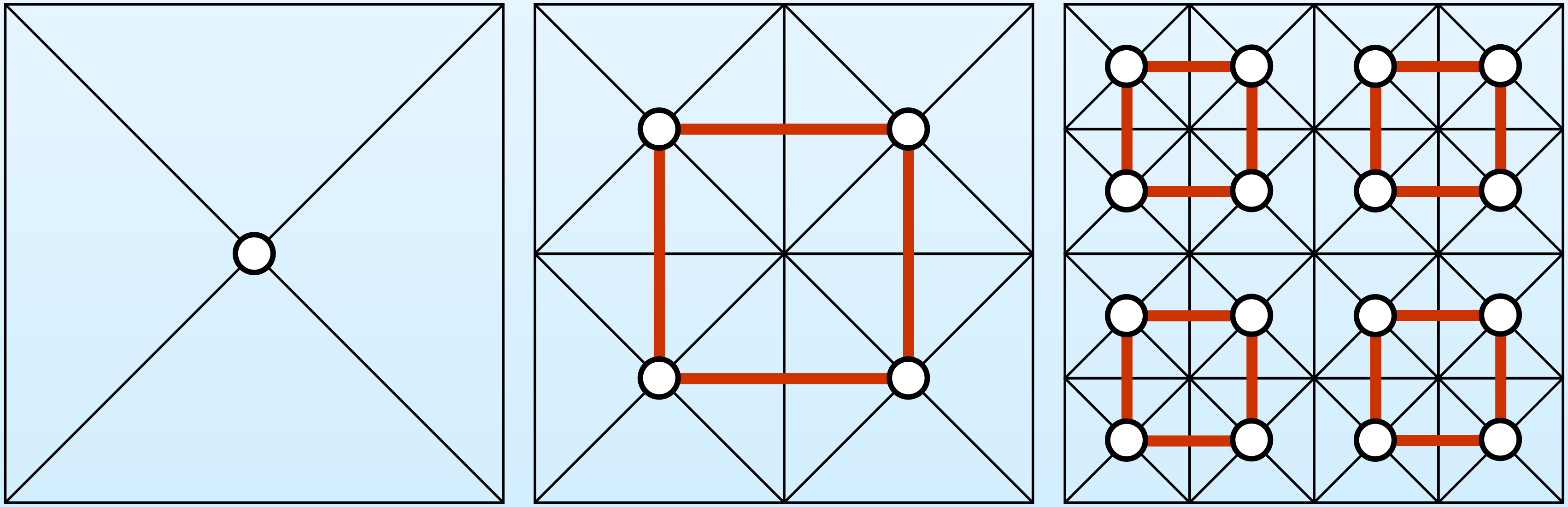


VB

va	vb	vc	vd	ve	vf	vg	vh	vi	v1				
								0	1/0				

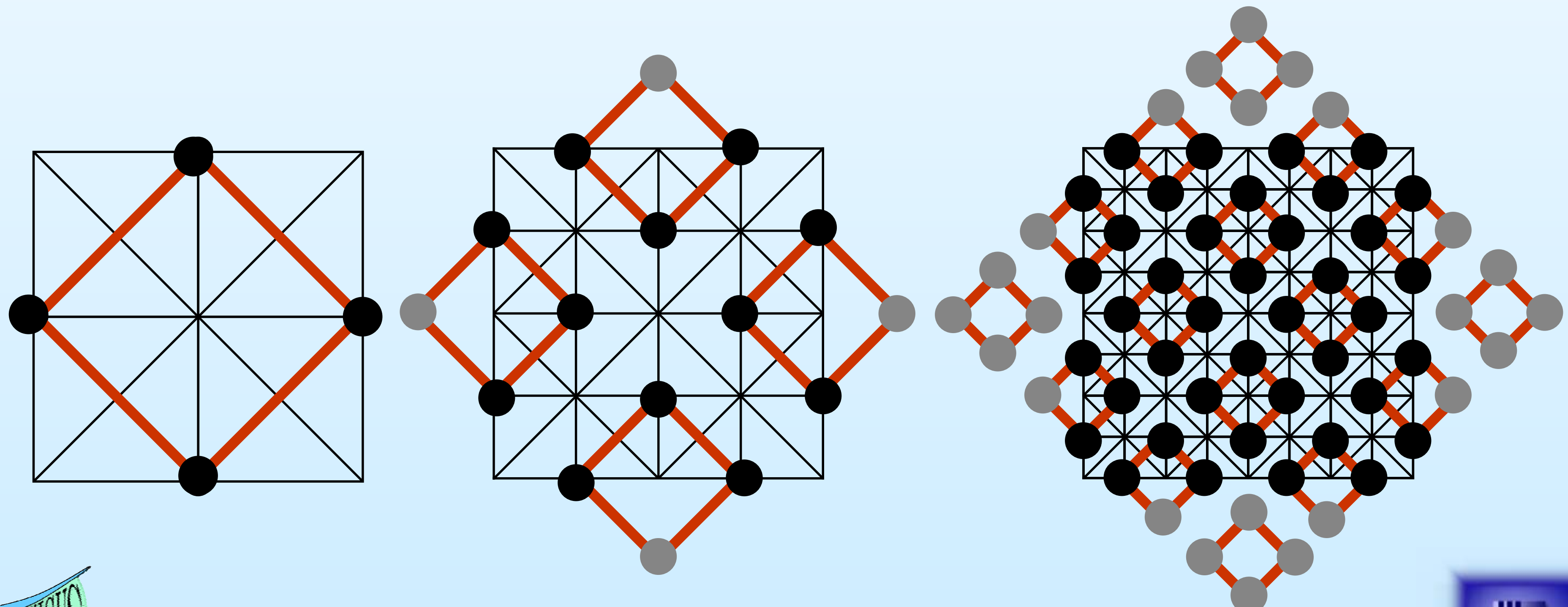
We develop a simple data layout based on quad-trees.

The vertices inserted at the even levels of refinement are the centers of each square and form a (white) quad-tree.

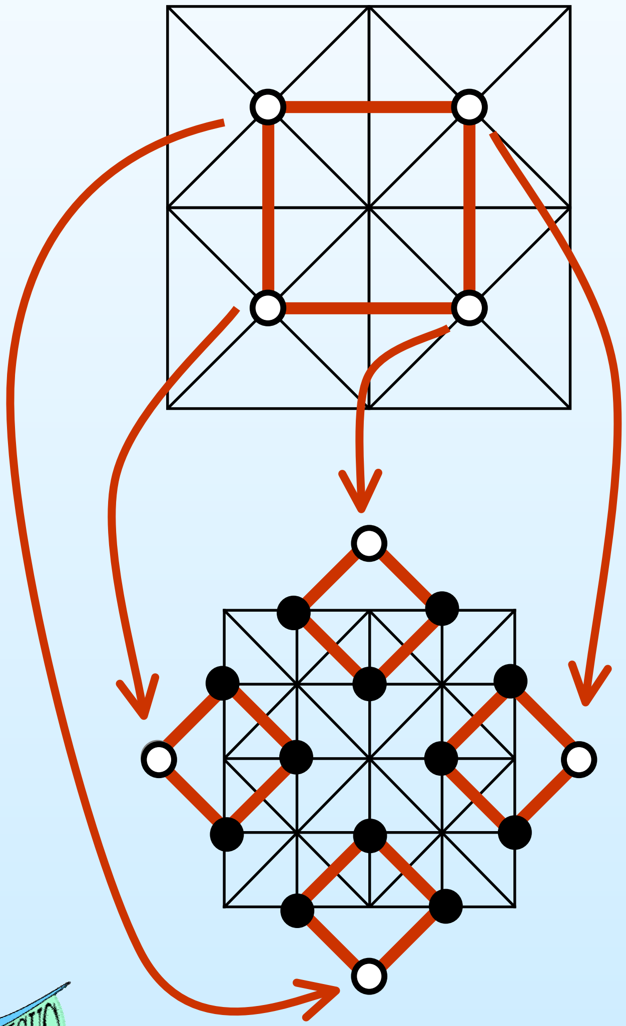


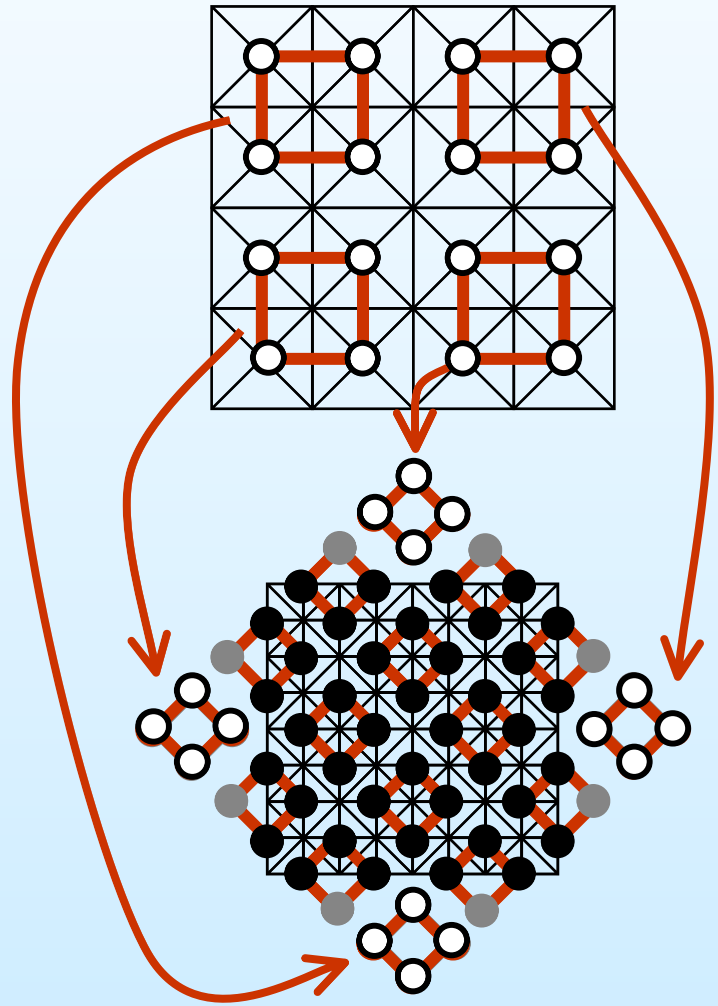
We develop a simple data layout based on quad-trees.

The black vertices inserted at the odd levels are the corners of each square and form a quad-tree if gray vertices are added.



We store the white nodes in place of the gray nodes.

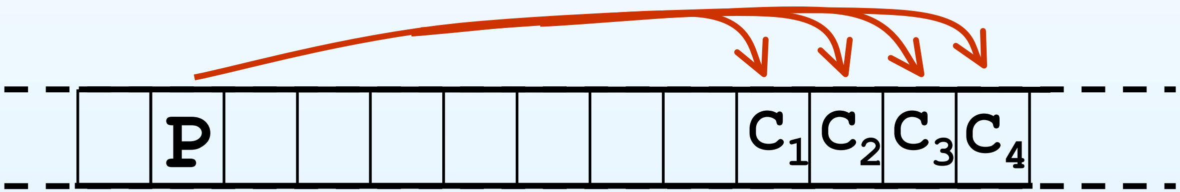


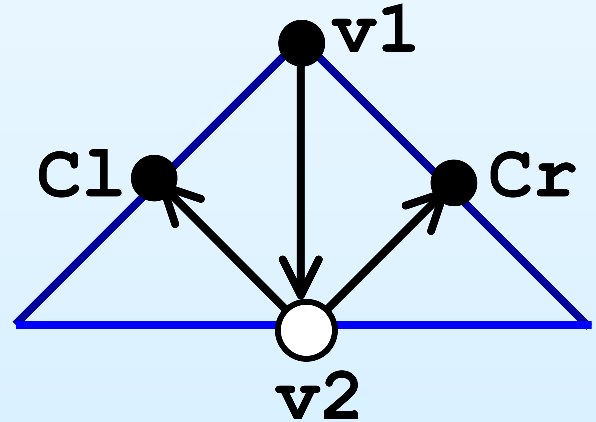


We simply layout the data element level by level starting from

The index of `Ci` is computed from the index of the parent `P`

$$C_i = 4*P+i$$





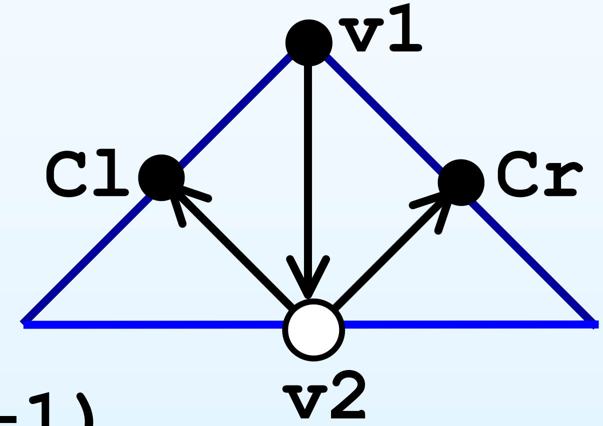
$$Cl = 4*v1-11+((2*v1+v2+2) \bmod 4)$$

$$Cr = 4*v1-11+((2*v1+v2+3) \bmod 4)$$

A hierarchical error metric simplifies the mesh construction and stripping.

vertex buffer level

```
mesh-refine(VB,v1,v2,l)
  if l>0 and ρ*(v1)≥τ then
    mesh-refine (VB,v2,Cl,l-1)
    strip-append(VB,v1,l mod 2)
    mesh-refine (VB,v2,Cr,l-1)
```



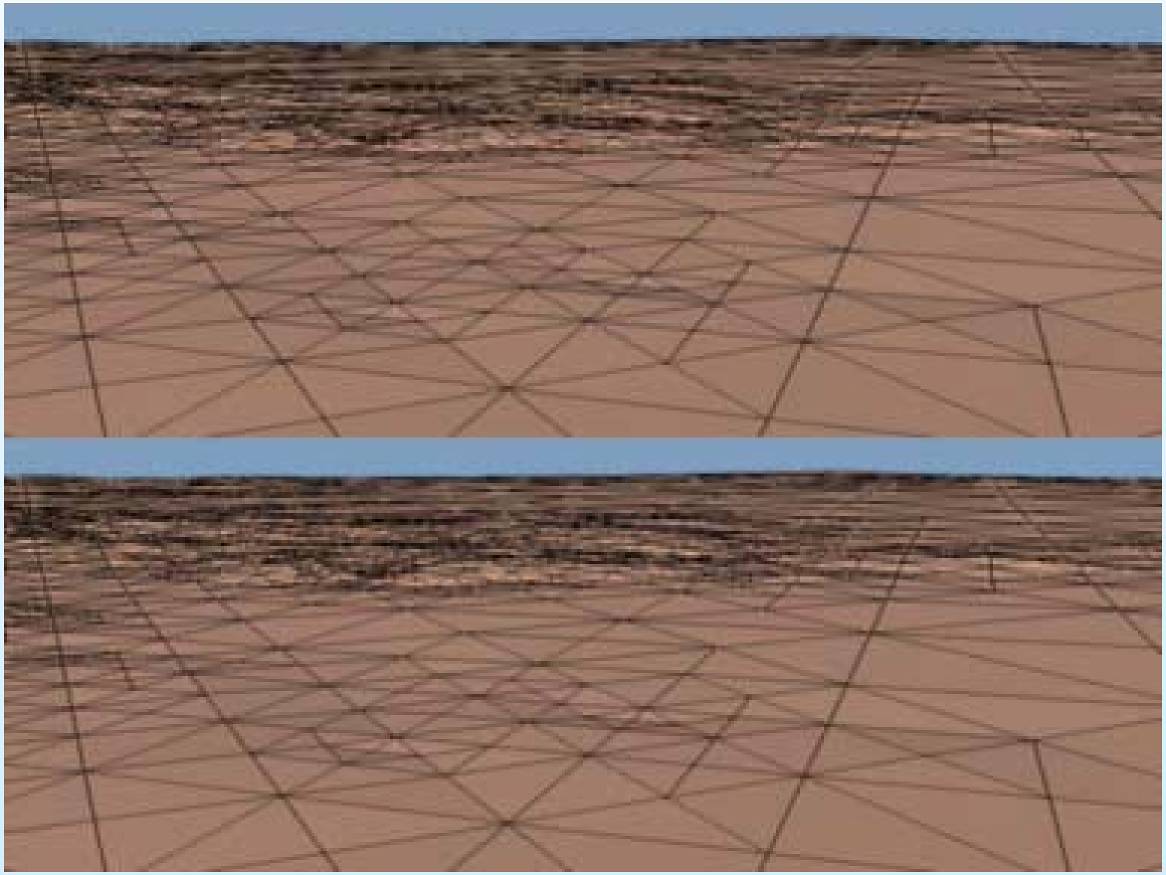
VB

va	vb	vc	vd	ve	vf	vg	vh	vi	v1				
								0	1/0				

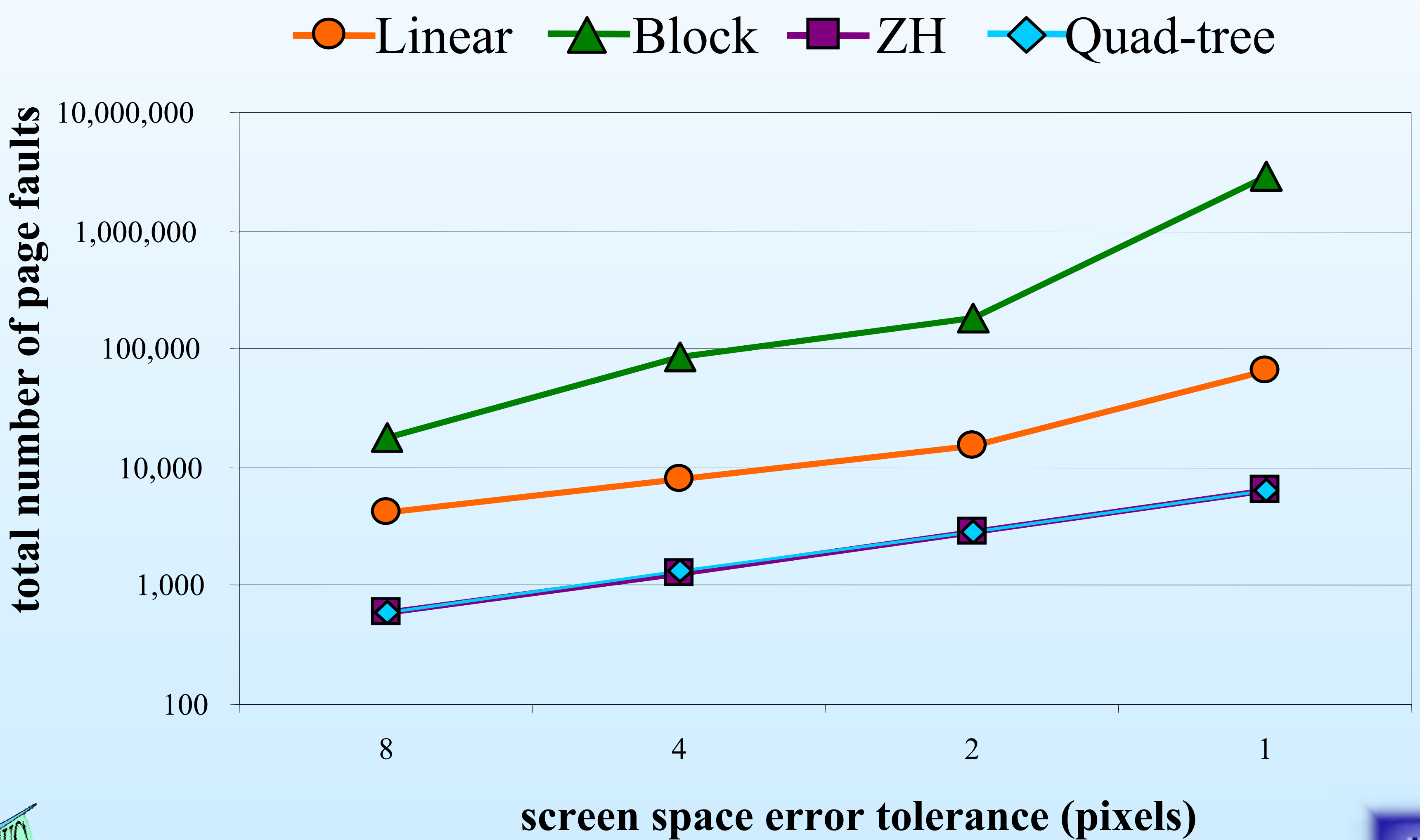
We implemented the scheme with four alternative data layout schemes.

Practical comparison:

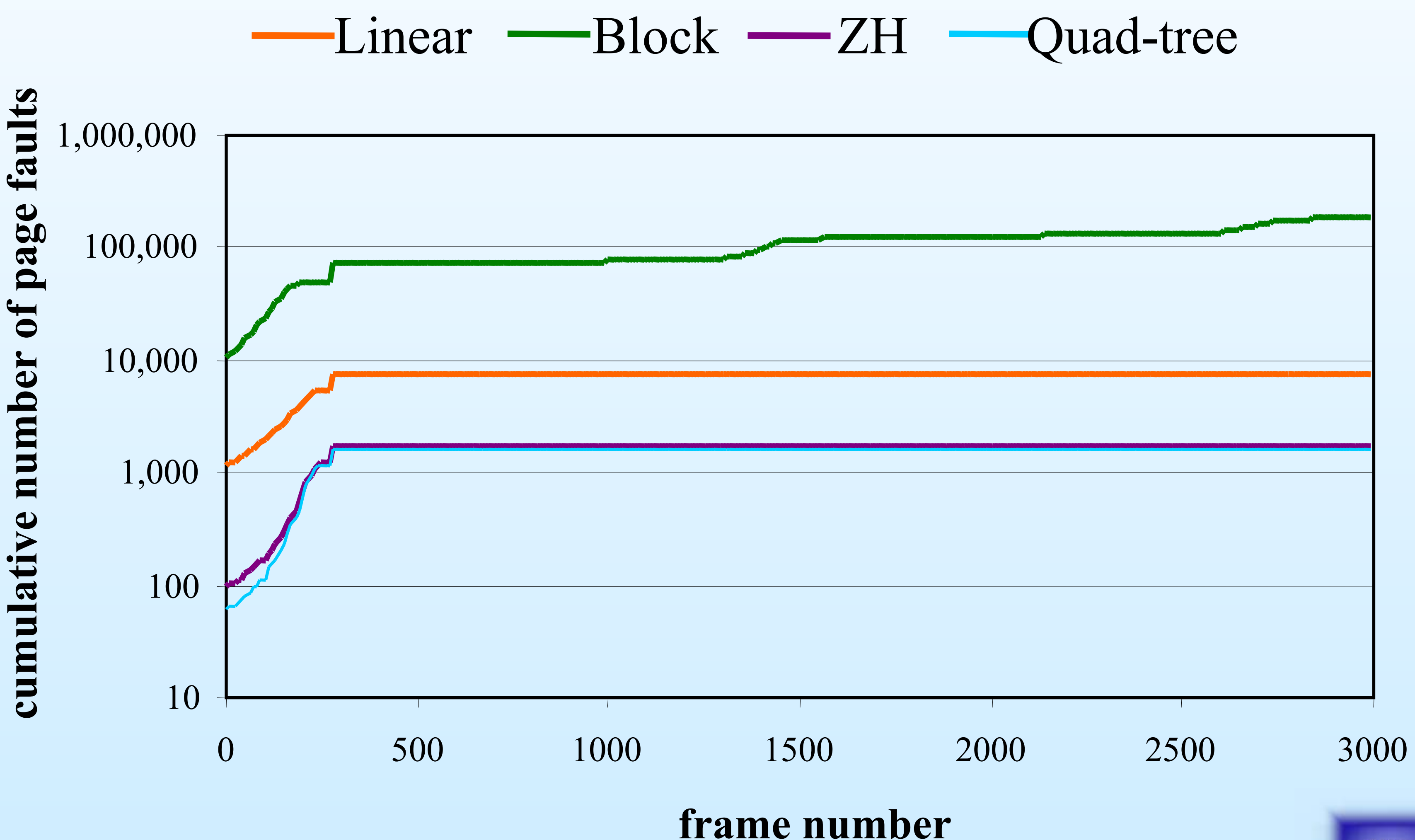
- Linear →
- Block
- ZH-order
- Quad-tree →



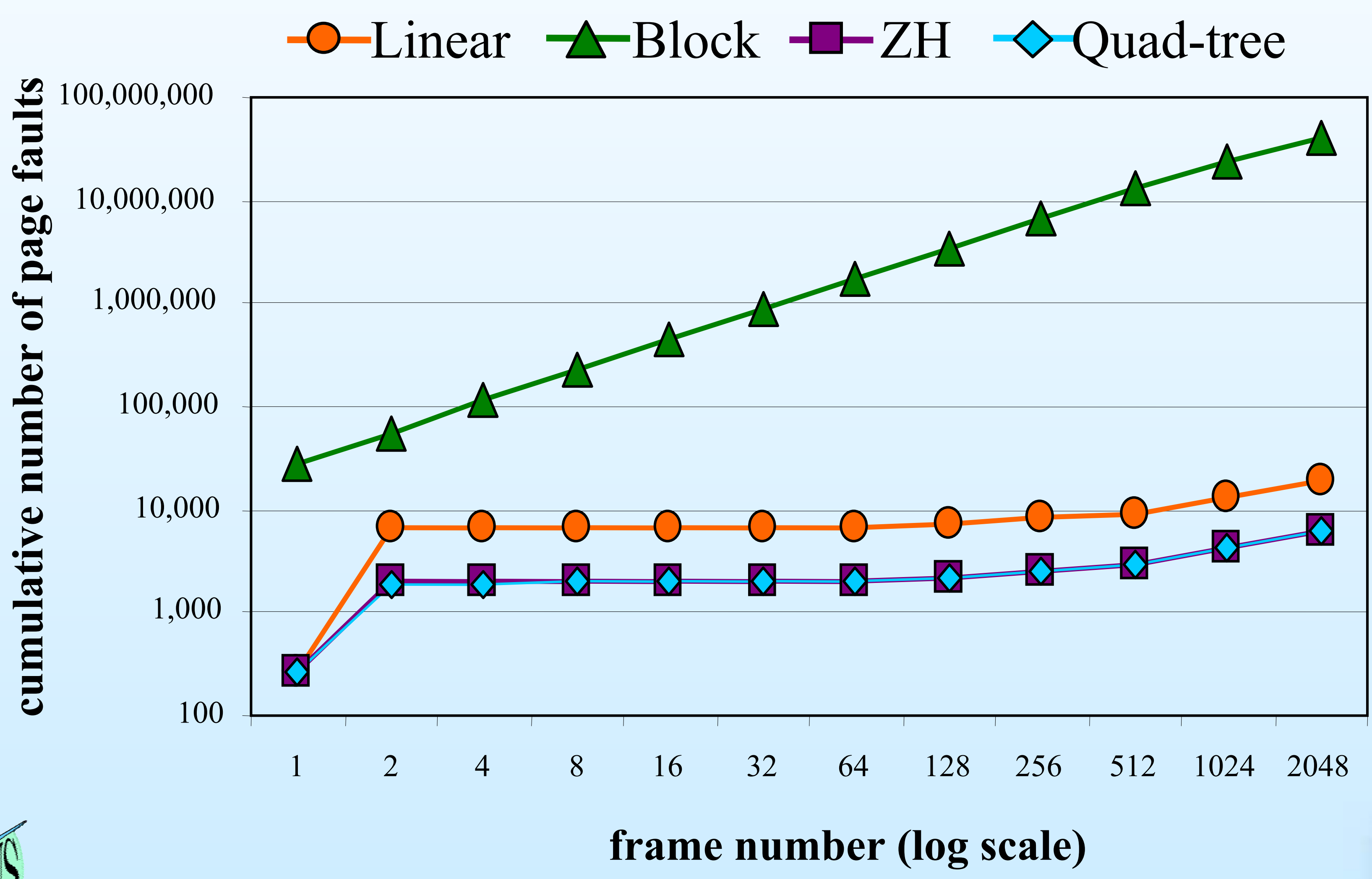
Performance Tests
(5GB dataset on a 800MB SGI)



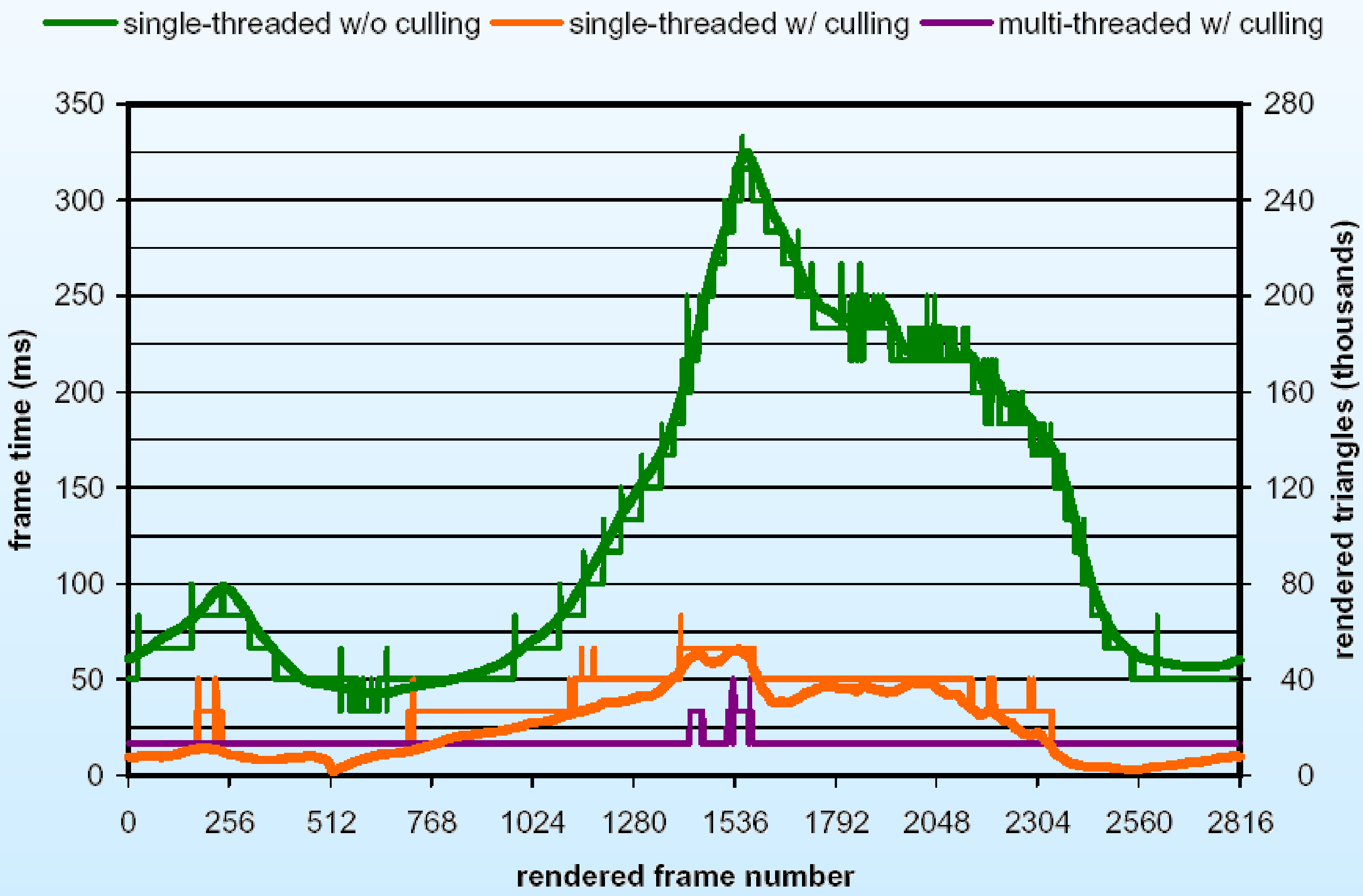
Performance Tests
(5GB dataset on a 800MB SGI)



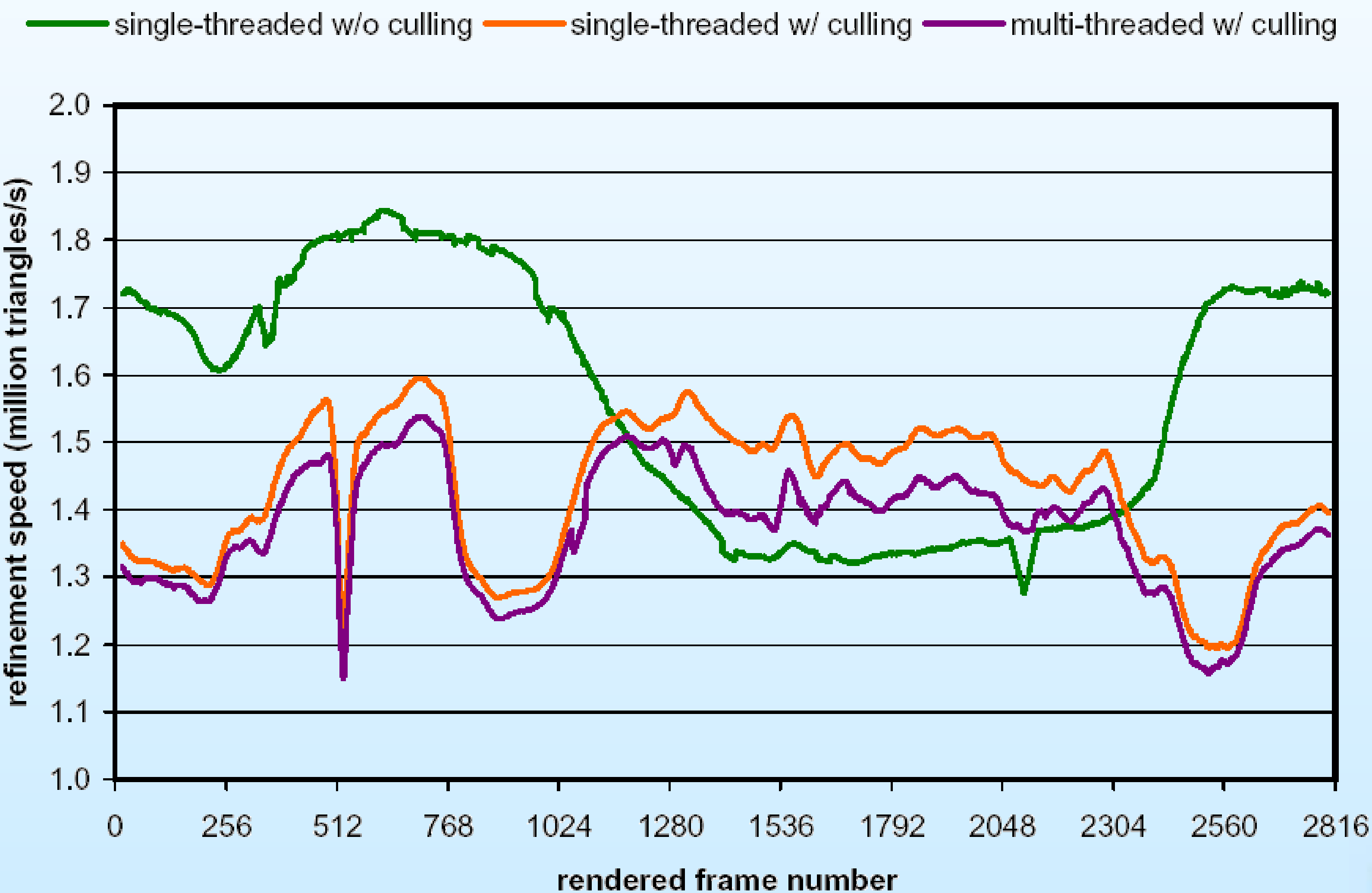
Performance Tests
(1.25GB dataset on a 64MB PC)



Comparison of in core performance with respect to threading and culling.



Comparison of in core performance with respect to threading and culling.



Conclusions and future directions.

- ✓ Incore speedup 2X
- ✓ Sustained 40k per frame – 30 HZ
- ✓ Good multithreading
- ✓ 1.5 Millions triangles per second

- Texture
- Mem efficiency
- Compression
- Explore more general 4-k meshes
- Geomorphing
- Add more sophisticate paging and prefetching

UCRL-PRES-154167

- This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.