Recursive Tetrahedral Meshes for Scientific Visualization

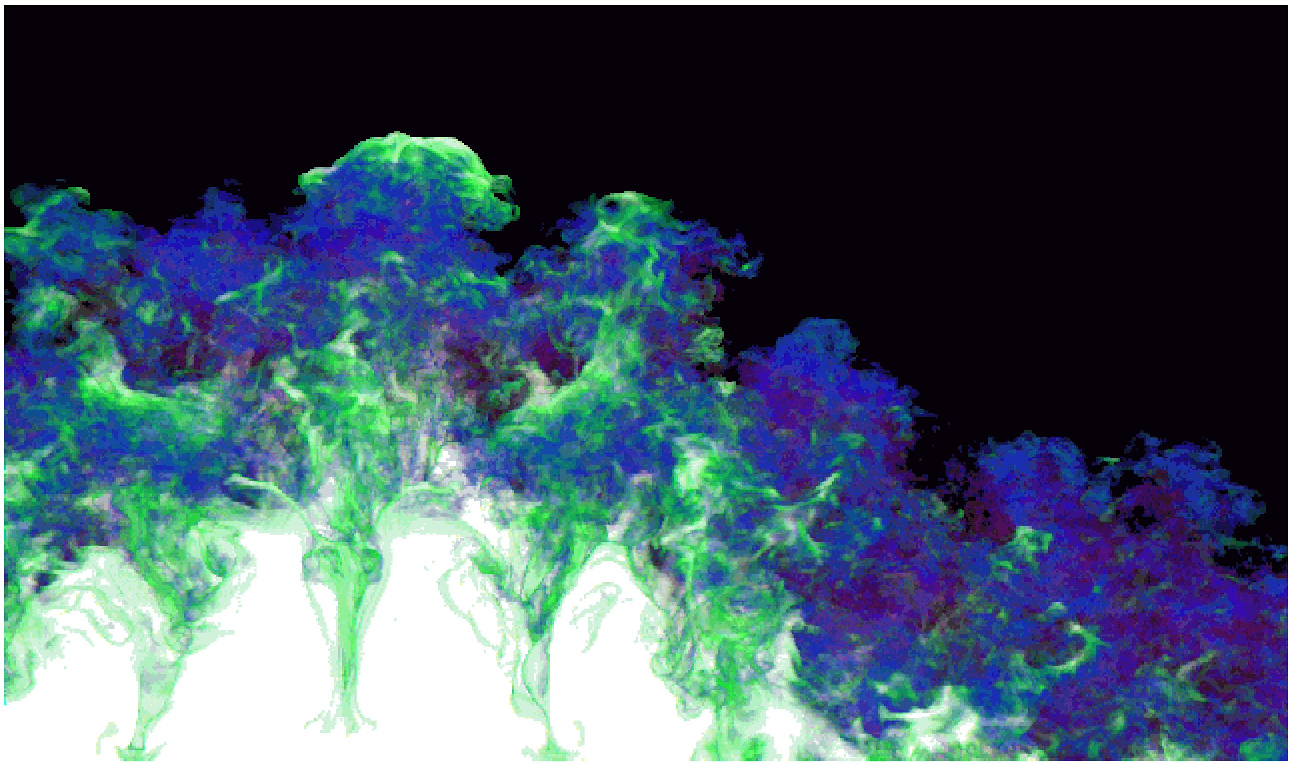
Benjamin F. Gregorski

Outline

- Motivation
- Longest Edge Bisection
  - 2D refinement with triangles
  - 3D refinement with tetrahedra
- Applications
  - Multiresolution Representation of Datasets with Material Interfaces
  - Fast View-dependent Isosurface Extraction
- Future Work


ASCI White
ASCI Blue-Pacific
ASCI Blue Mountain
NPACI Blue Horizon
ASCI Red
ASCI Cplant

Simulation of Richtmyer-Meshkov instability

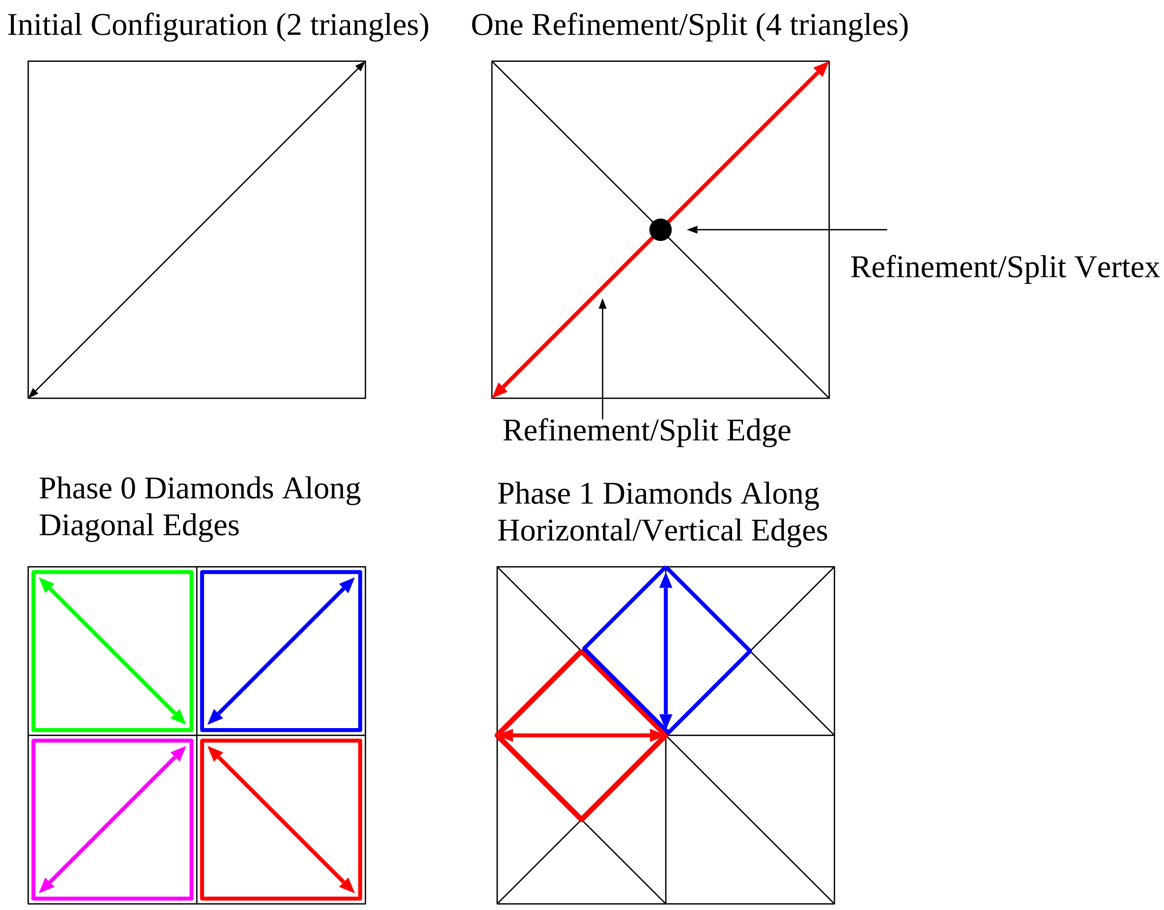


Winner of 1999 Gordon Bell Award for Performance Dec. 6, 1999 mirin12

Motivation

- Rapidly increasing dataset size
  - Numerical Simulations and Medical Scans
  - Too large to process and to fit in memory
  - 2k x 2k x 2k datasets with 300 timesteps
- Still need interactive visualization
  - Isosurface extraction
  - Volume rendering
  - Hybrid rendering (combine point, surface, and volume techniques)
- Process only what is necessary
  - Only keep necessary data in memory
  - Reuse computations

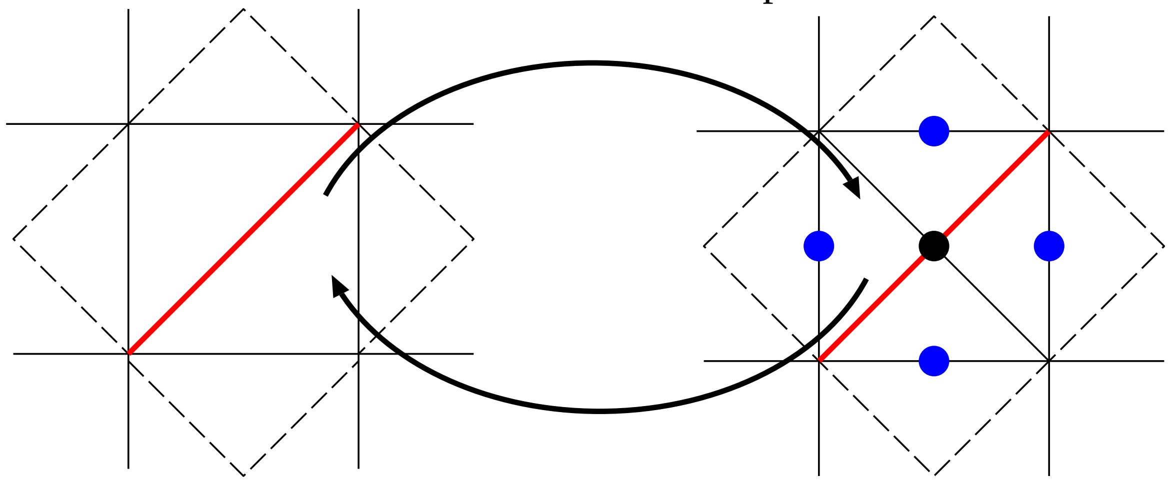
2D Longest Edge Bisection



Phase 0 Split/Merge

Splitting a Phase 0 Diamond

Gives 4 Phase 1 tets and child diamonds
Child split vertices are shown in blue



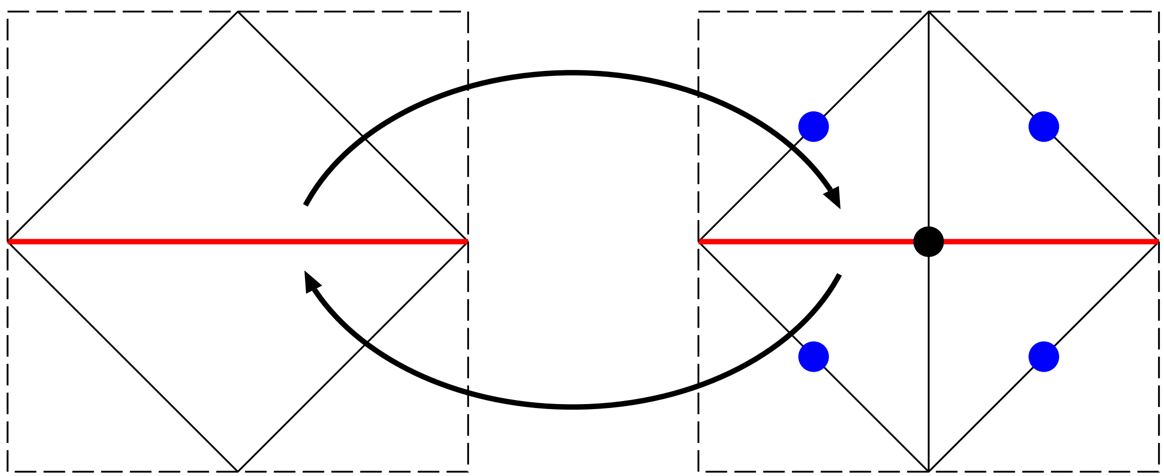
Removes 4 Phase 1 tets from child diamonds

Merging a Phase 0 diamond

Phase 1 Split/Merge

Splitting a Phase 1 Diamond

Gives 4 Phase 0 tets and child diamonds
Child split vertices are shown in blue



Removes 4 Phase 9 tets from child diamonds

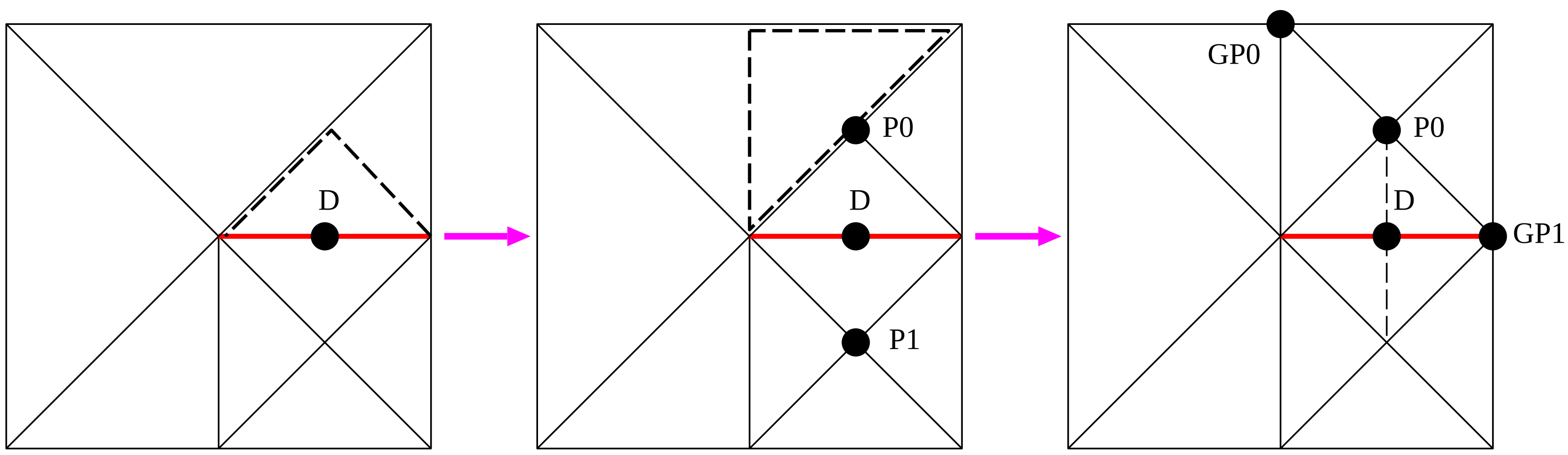
Merging a Phase 1 diamond

Adaptive Refinement

Splitting a Diamond D all of whose tets are not in the mesh

D's parents (P0 & P1) must be split

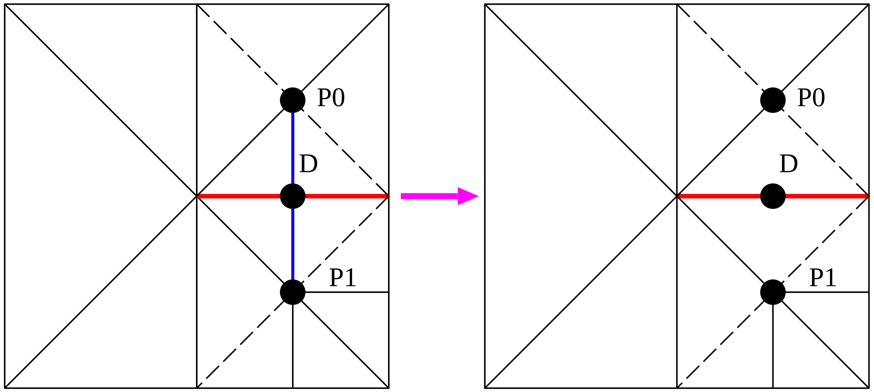
P0's parents (GP0 & GP1) must be split



Merging Diamonds

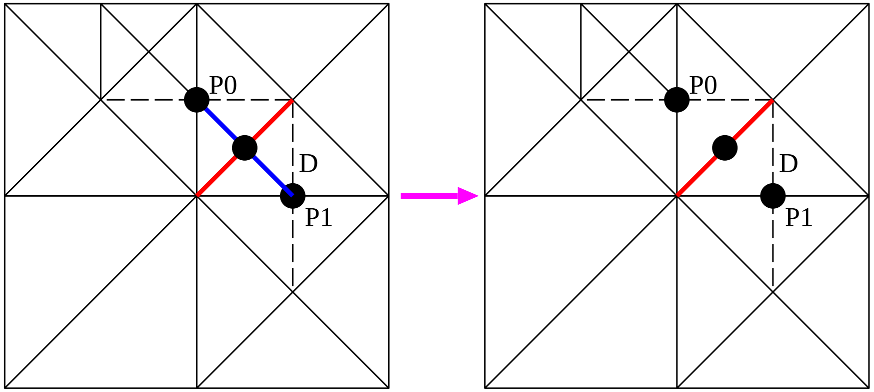
Merging a phase 1 diamond

Parent P0 can be merged
but P1 cannot be merged



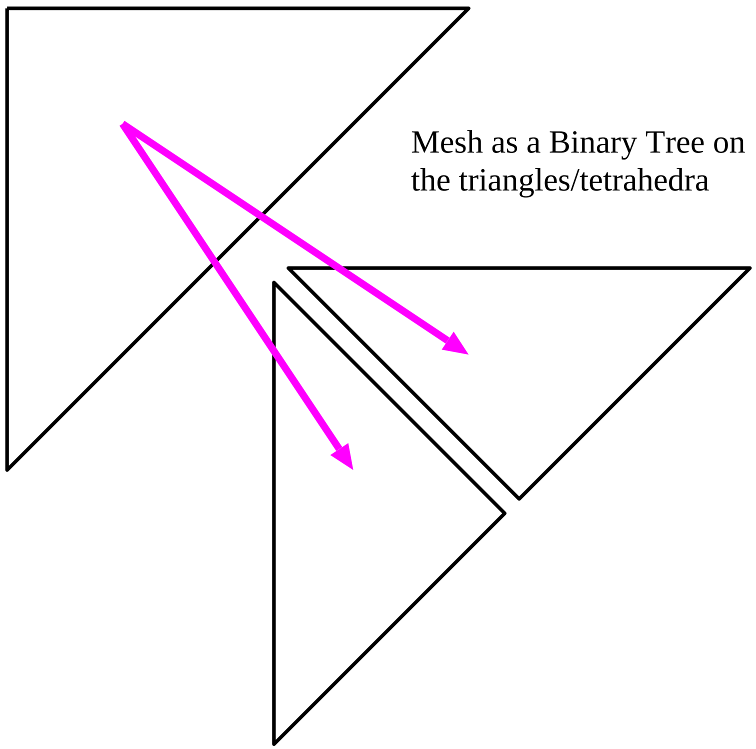
Merging a phase 0 diamond

Parent P1 can be merged
but P0 cannot be merged

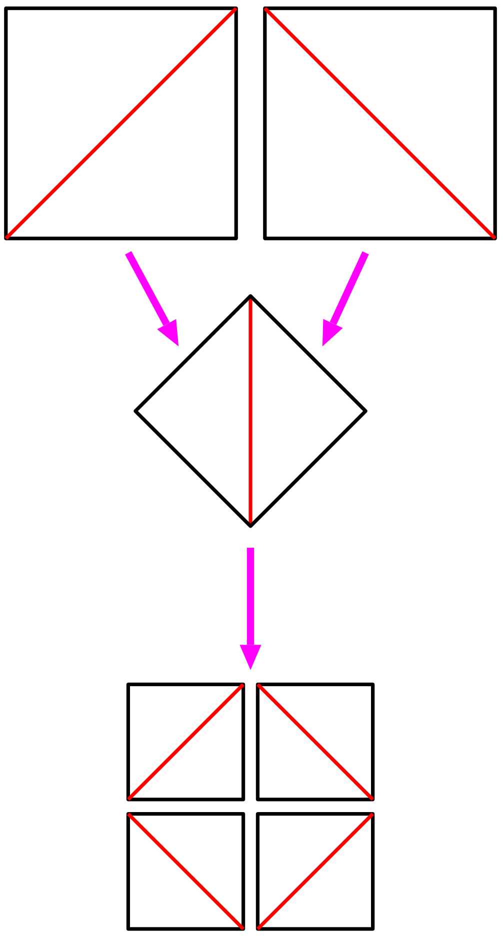


Mesh Structure

Mesh as a Binary Tree on
the triangles/tetrahedra



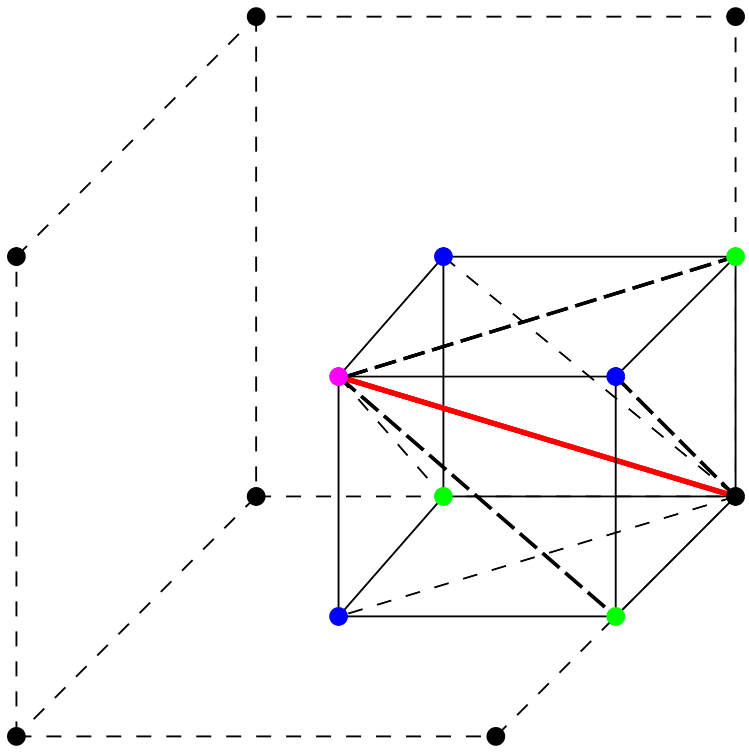
Mesh as a Directed Acyclic Graph
on the parent and child diamonds



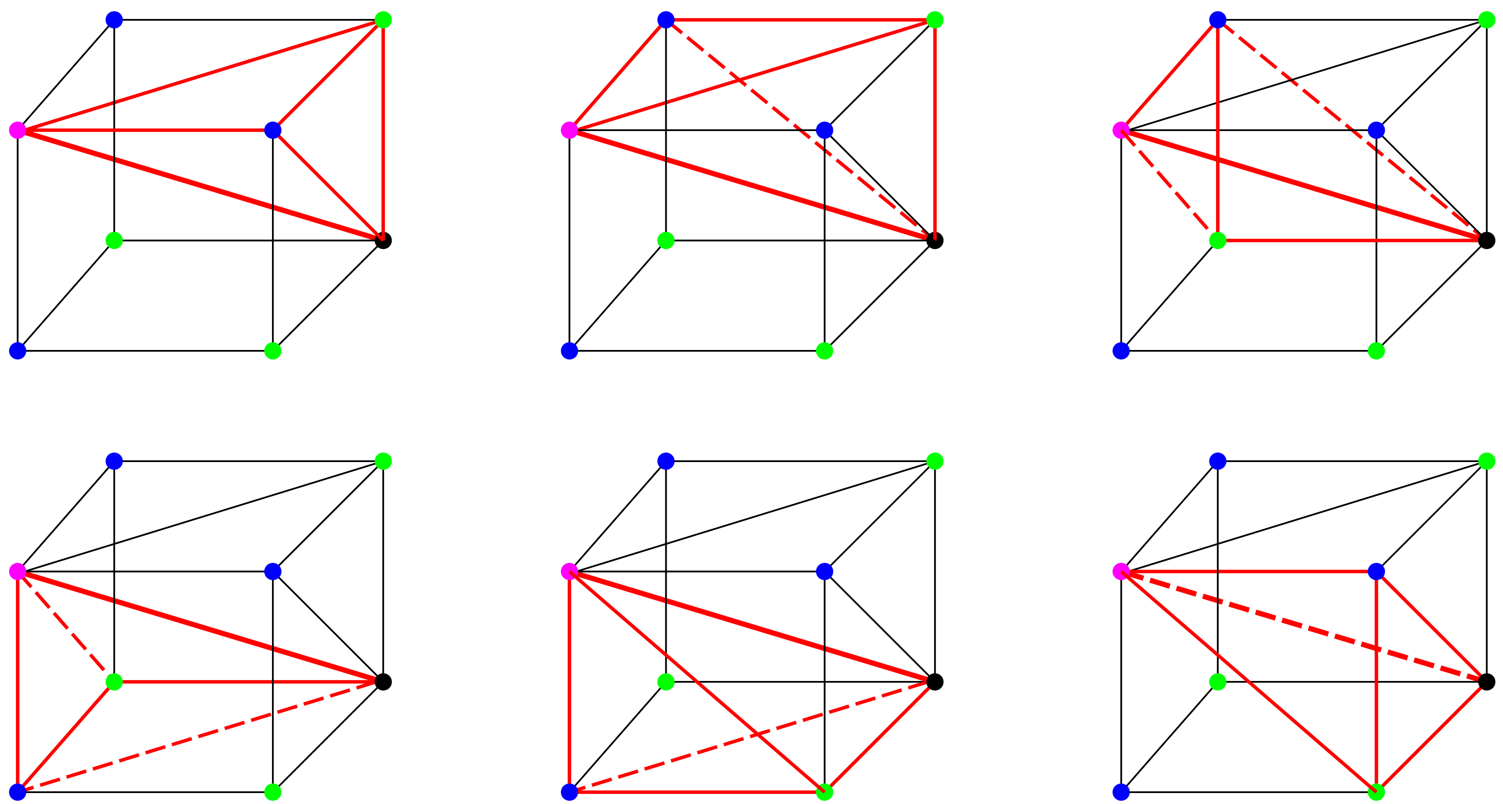
3D Mesh Refinement

- Tetrahedral Mesh Structure
  - Longest edge bisection
  - Crack-free adaptive refinement
- Multiresolution volumetric representation
  - Recursive Tetrahedral Meshes
    - Used to build multiresolution volumetric representation
  - Construct an approximate representation
    - Select different levels of detail from tetrahedral mesh
  - Visualize this approximate representation
    - Slicing, Isosurface extraction, Volume Rendering …

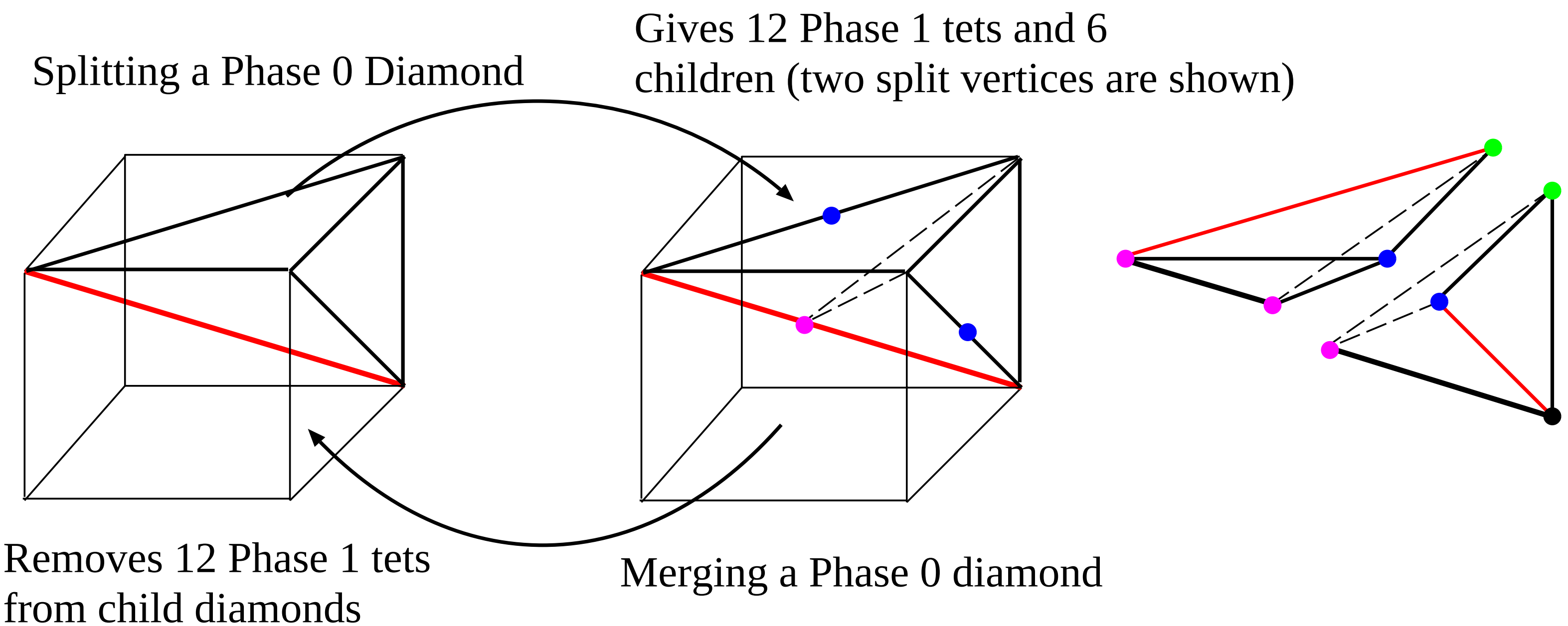
Initial Configuration



Initial Configuration



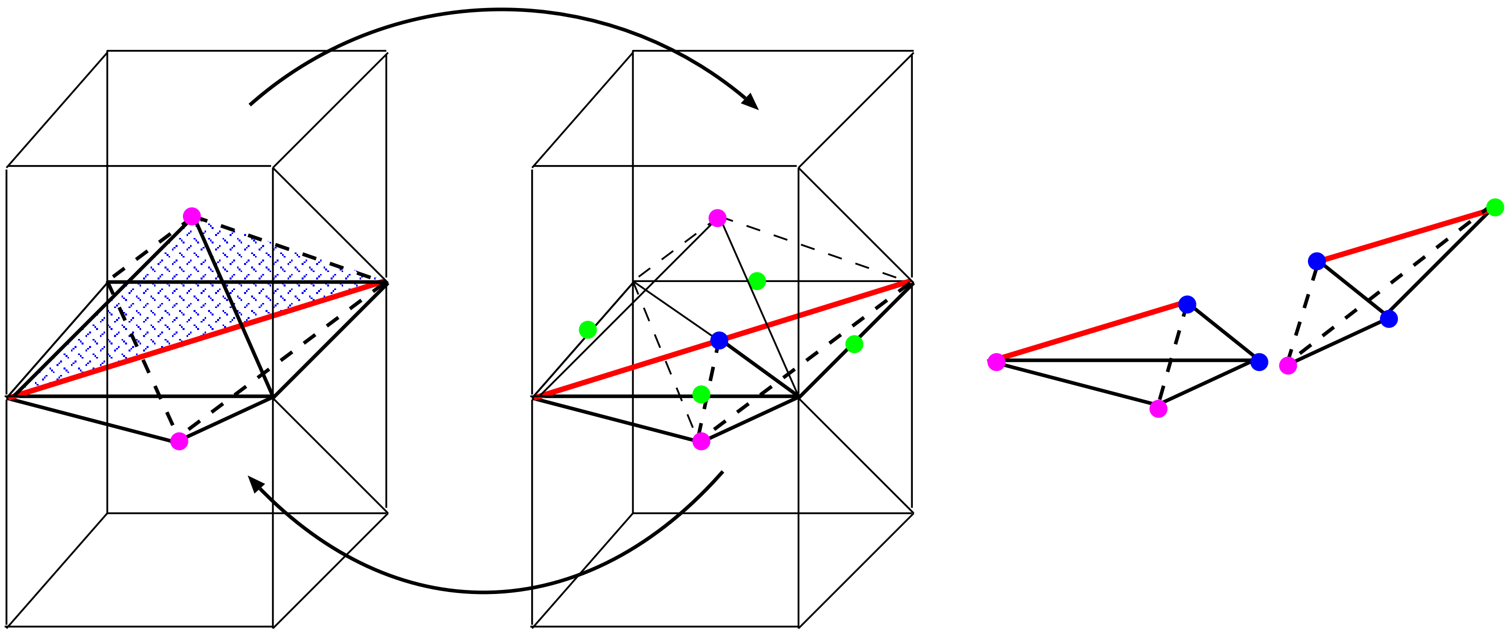
Phase 0 Split/Merge



Phase 1 Split/Merge

Splitting a Phase 1 Diamond

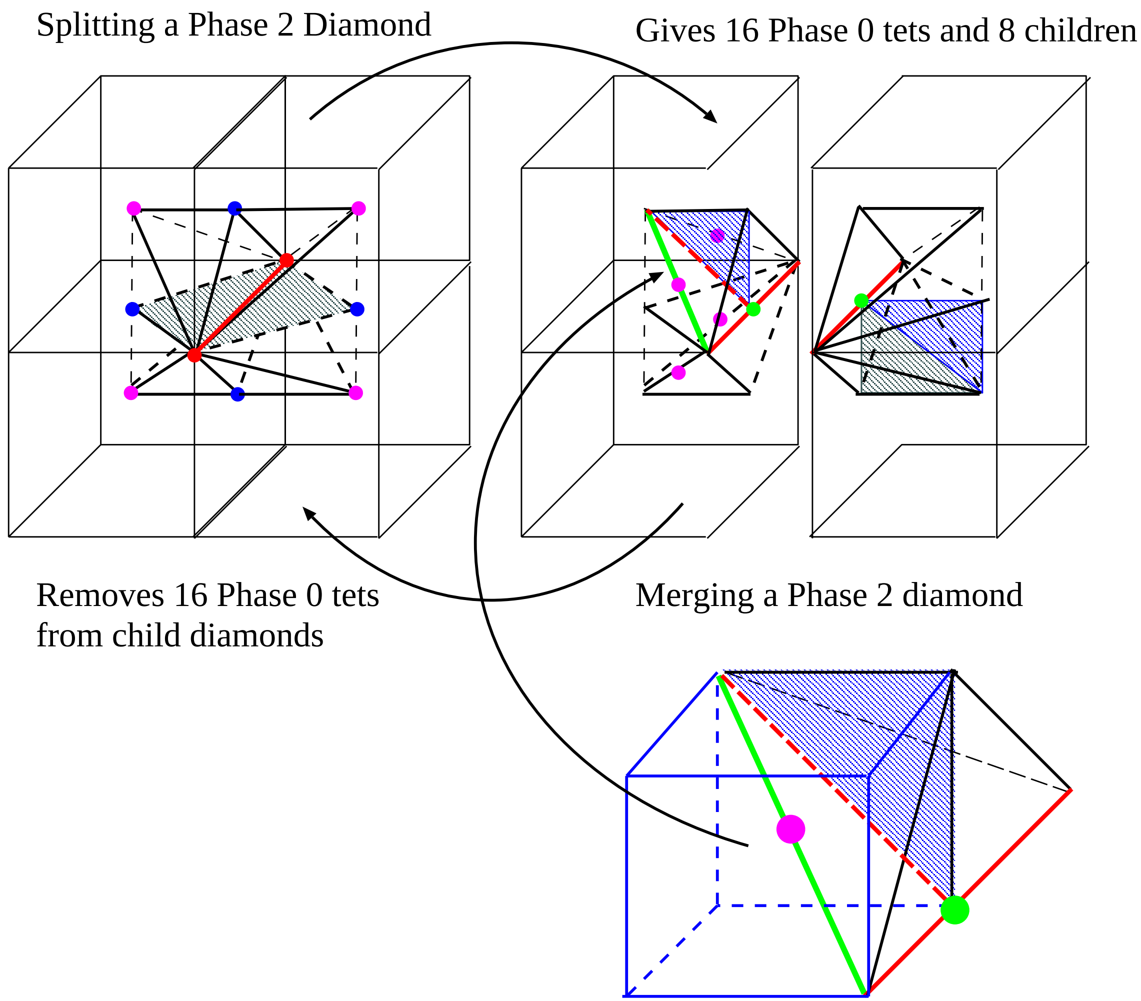
Gives 8 Phase 2 tets and 4 children



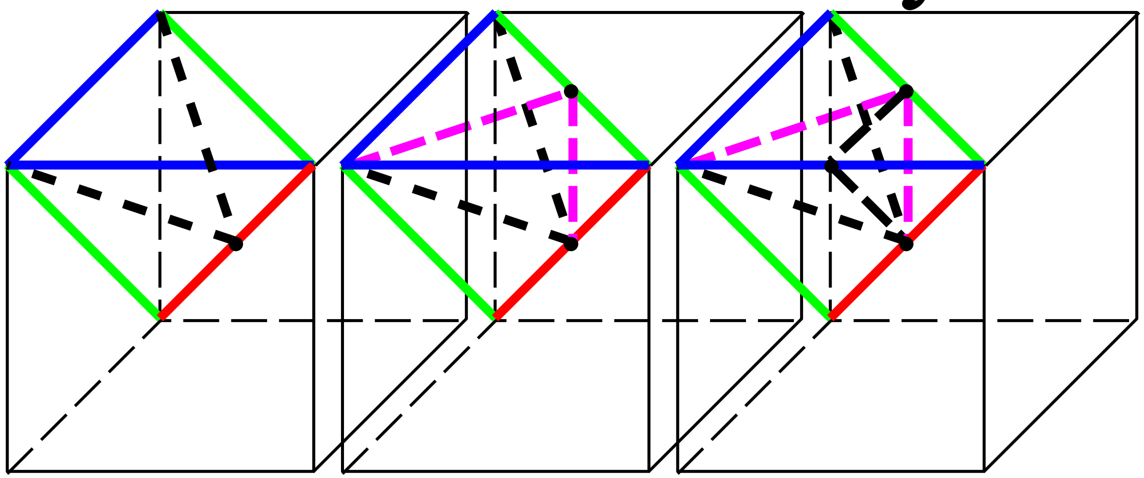
Removes 8 Phase 2 tets
from child diamonds

Merging a Phase 1 diamond

Phase 2 Split/Merge

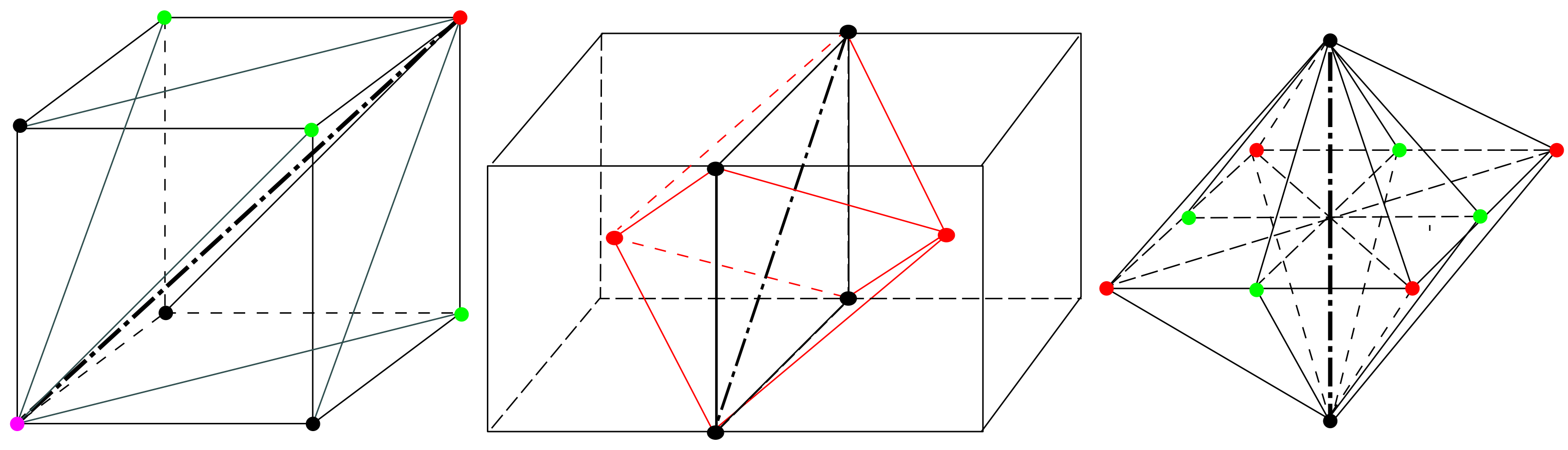


3D Refinement Summary



- Three phases of refinement
- Split cells, then faces, then edges of an octree
- Split a tet along the *split edge* at the *split vertex*
- Three refinements equivalent to one octree subdivision

Summary of Diamond Shapes



- Three type of diamonds around a split edge
- Refine all tetrahedra in a diamond at the same time

Error Based Refinement Strategy

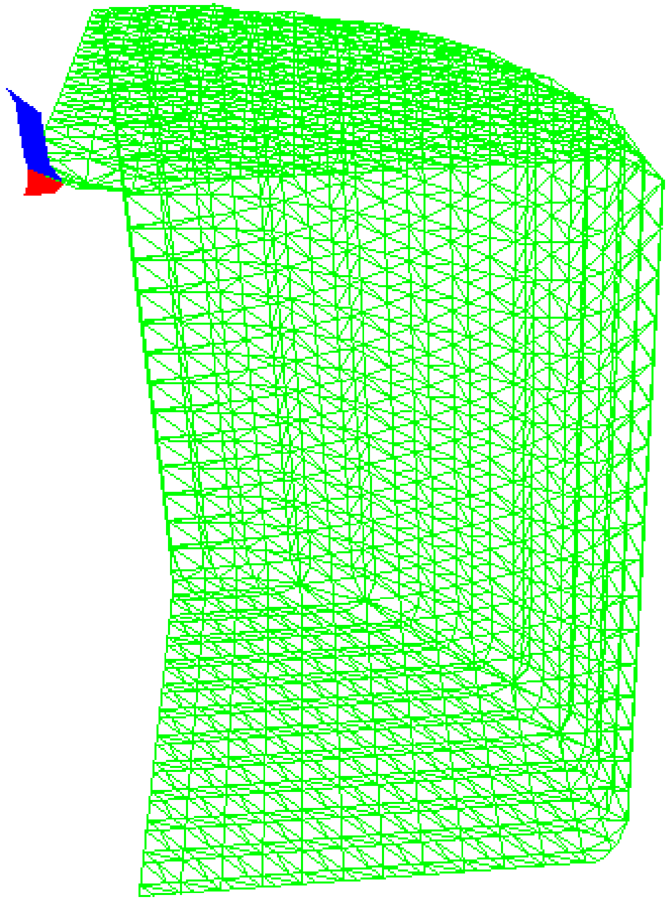
- Priority queues (split queue and merge queue)
  - Ordered by an error tolerance ***E***
- Error recomputation
  - Recompute error values for diamonds in split and merge queues
- Mesh Refinement
  - Split diamonds with error > ***E***
  - Merge diamonds with error < ***E***
- Stopping Criteria
  - Splitable diamonds have errors < E
  - Mergeable diamonds have errors > E
- Visualize leaf tetrahedra

Multiresolution Representation of Datasets with Material Interfaces

- What are material interfaces in datasets
  - Explicit surfaces of discontinuity
- How are material interfaces represented
  - Signed distance functions
- Discontinuous field representations
  - Representing fields with explicit discontinuities
- Multiresolution represention
  - Resampling of datasets on tetrahedral grid
  - Dual error metrics
    - Error in interface approximation
    - Error in reconstructed field
- Examples

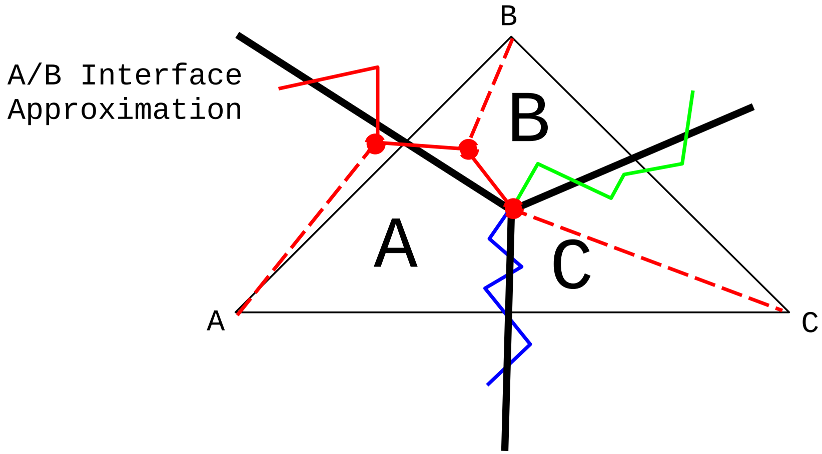
Material Interfaces

- Interfaces in Computational Simulation
  - Explicit discontinuities, physical boundaries
  - Discontinuous fields across interface
  - Extracted from volume fractions
- Example of a projectile impacting a block
  - Three interfaces
- Our approach
  - Adaptively resample the dataset on a tetrahedral mesh
  - Separate field representations on either side of a material interface

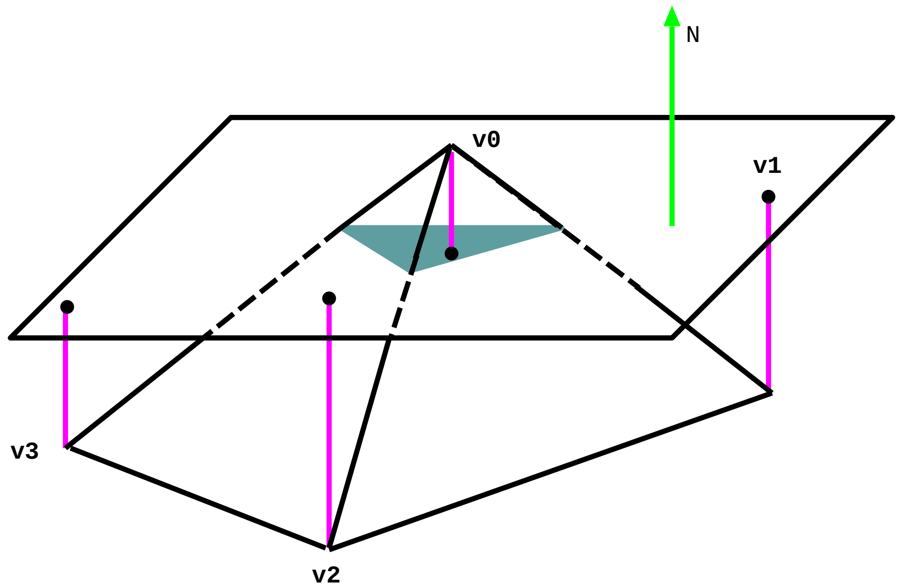


Representing Material Interfaces

- Interfaces given as triangle meshes
- Implicit representation
  - Zero set of signed distance function
  - One value per tet vertex per interface



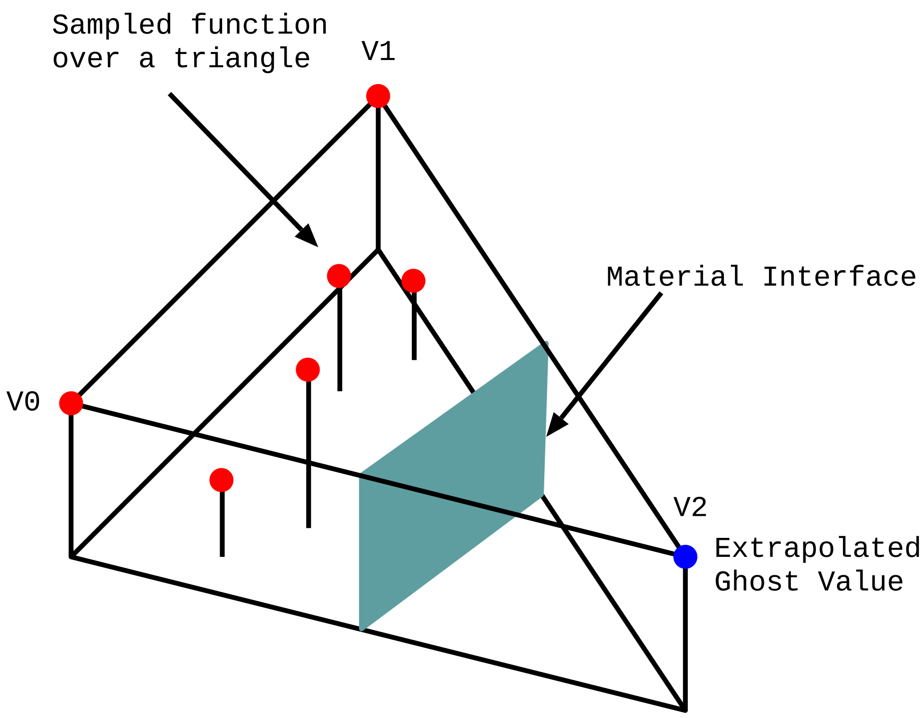
Triangle with three interfaces



Approximation through a tetrahedron

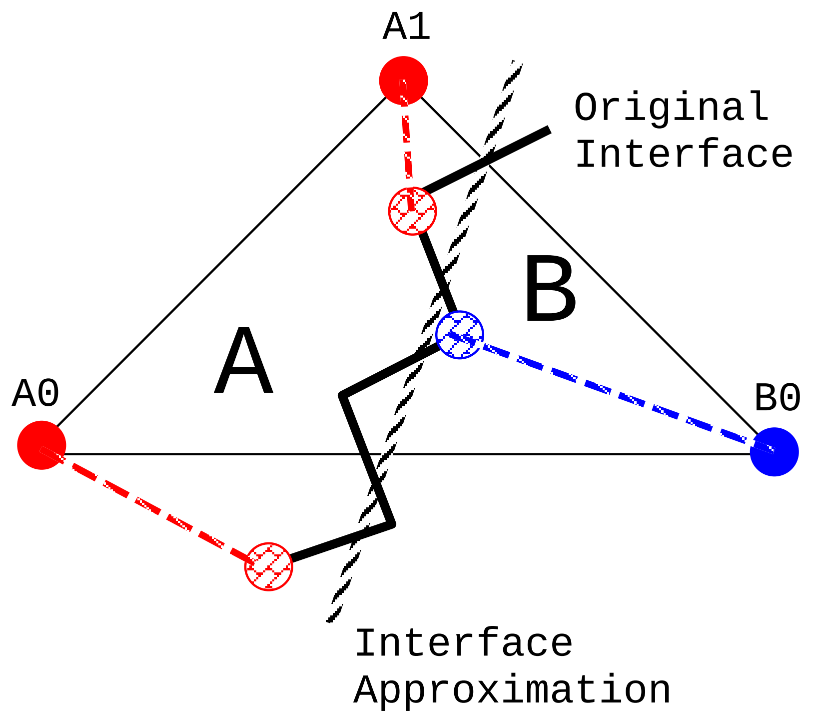
Discontinuous Field Representations

- Tetrahedra with interfaces
  - Poor approximation of field values with linear interpolation
- Our solution
  - Separate field representations for each material
- Ghost Values
  - Need extra field values at vertices
  - Extrapolate these values across interface boundaries



Computing Ghost Values

- Purpose of ghost values
  - Linear interpolation requires values at each of the vertices
  - Each vertex needs a field value for each material
- Ghost Values
  - Extrapolated from known values
  - One ghost value per material per vertex



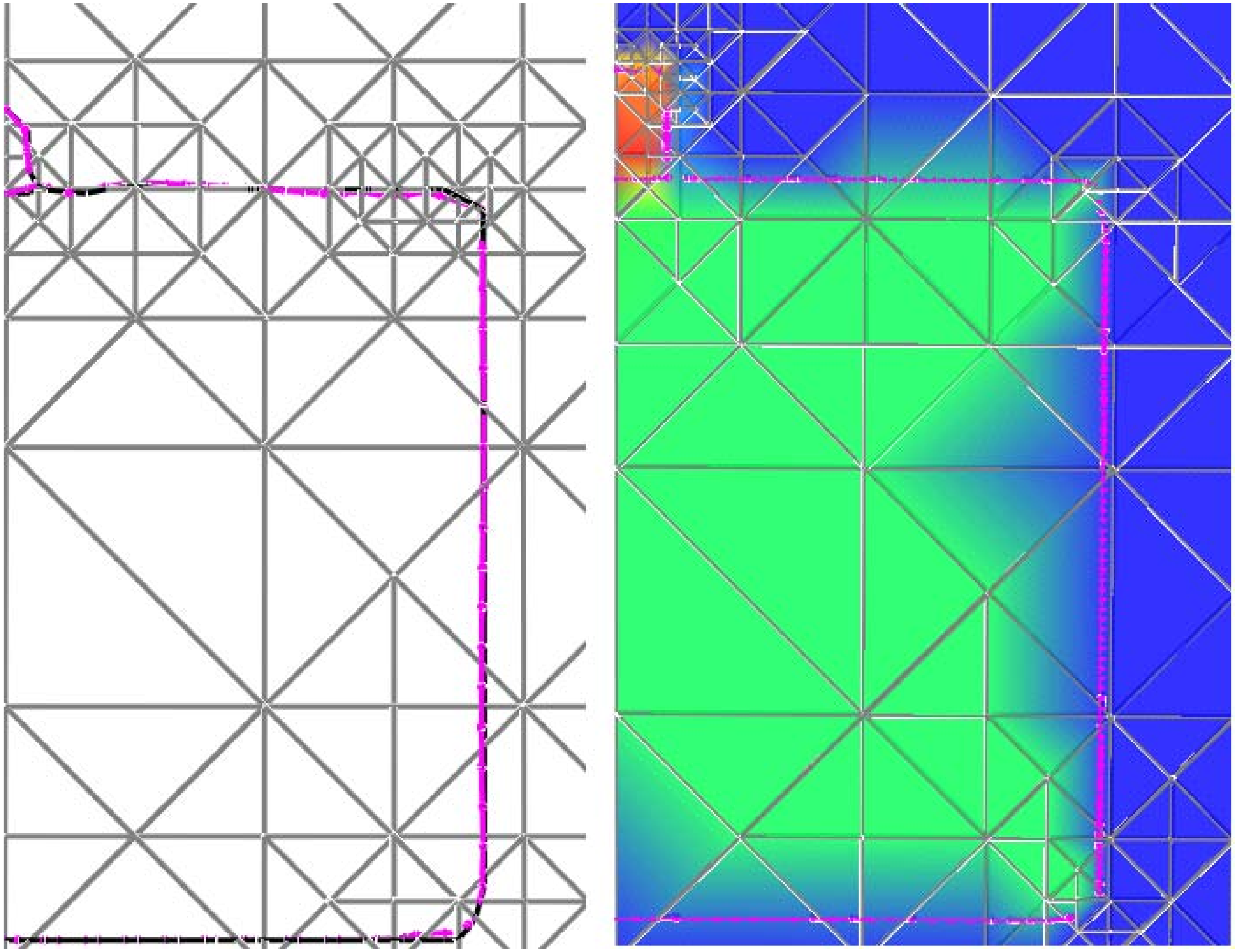
Ghost values for material B needed at vertices A0 and A1

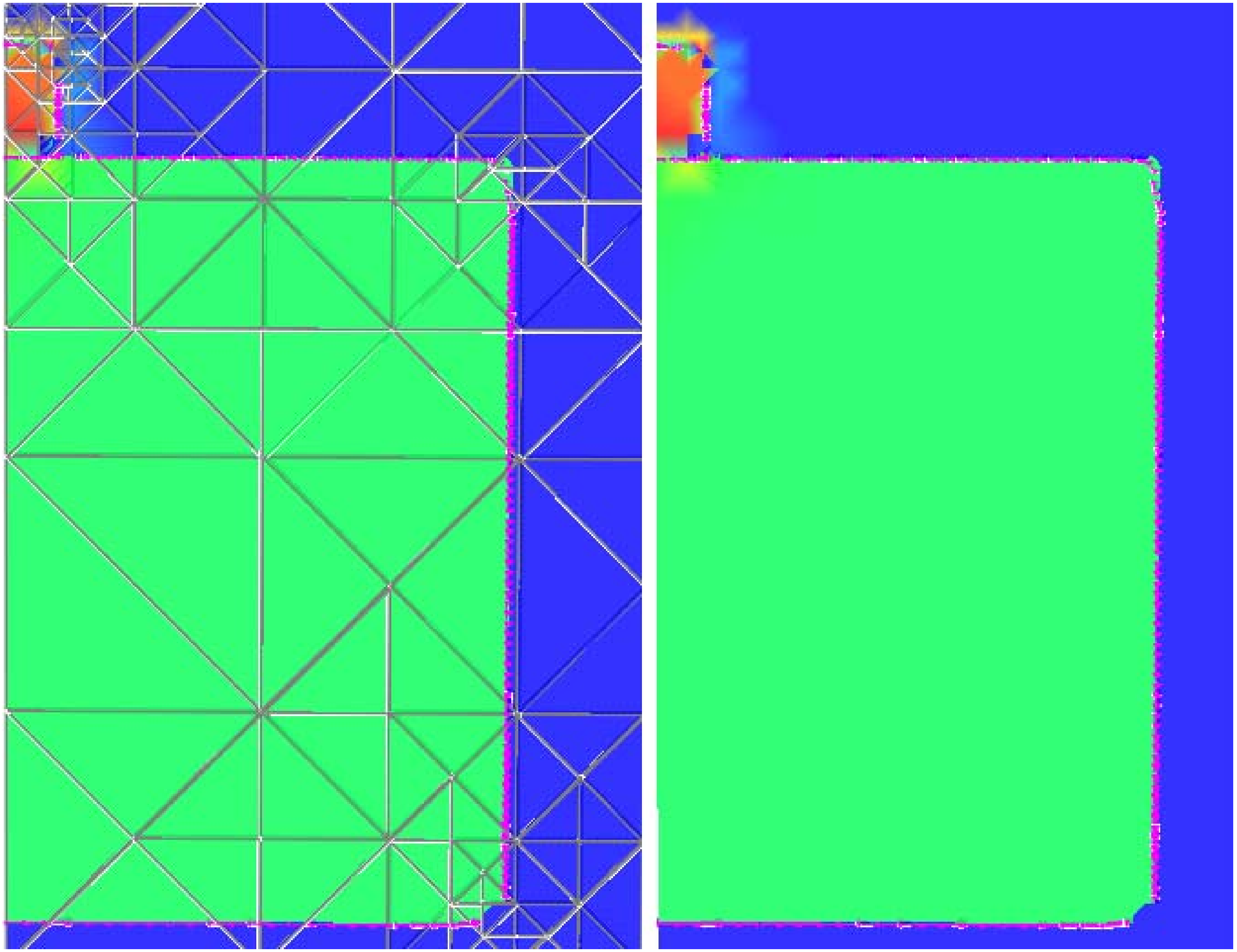
Building A Multiresolution Represention

- Single algorithm for a large number of dataset types
  - Rectilinear, Curvilinear, Adaptively refined
- Adaptive resampling
  - Sample on the vertices of a tetrahedral mesh
  - Field representations over tetrahedra
    - Explicit representation of material interfaces
    - Separate field representations for each material
  - Error metrics
    - Interface approximation error and field approximation error

Results

- Simulation of projectile impacting a solid block
  - Density, pressure, per-material densities, volume fractions
  - Interfaces reconstructed from volume fractions
  - Cell centered data
- Three material interfaces
  - Block/Empty Space, Projectile/Empty Space, Projectile/Block
- Numbers
  - Initial size 32x32x52 for 53248 nodes
  - 3200 Tetrahedra with interface error = 0.15
  - 12K Tetrahedra with field error ~ 0.007





View-dependent Extraction of Isosurfaces

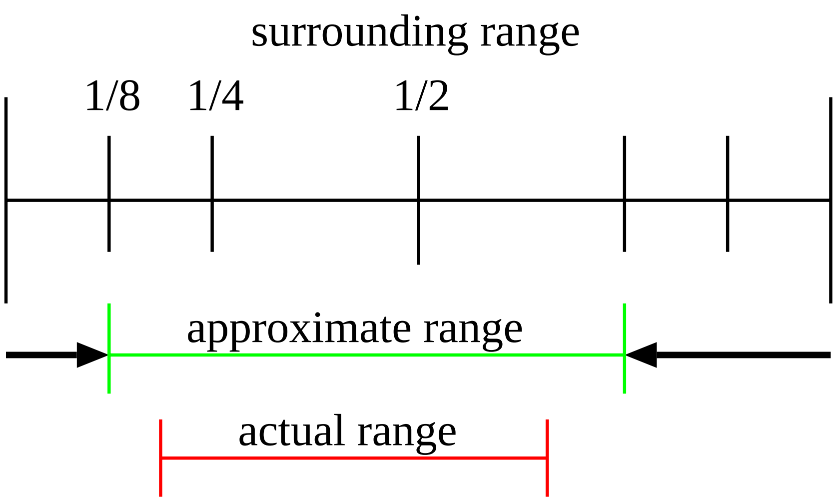
- Algorithm Overview
    - Split/Merge refinement
- Preprocessing
    - Min/Max values
    - Gradients
- Error Metrics
    - Field error, isosurface error, view-dependent error
- Data layout scheme
- Results

Algorithm Overview

- Given a view-dependent error ***E***
- Error recomputation
  - Mark diamonds without the isosurface as *empty*
  - Mark diamonds outside of the view-frustum as *invisible*
  - *Invisible* and *empty* diamonds have an *error* of 0
- Mesh Refinement
  - Merge invisible and empty diamonds
  - Delay isosurface extraction for invisible diamonds
- Stopping Criteria
  - View-dependent error satisfied or max triangle count reached
  - Time for processing current frame has expired
- Extract isosurface from leaf tetrahedra

Precomputed Data Values

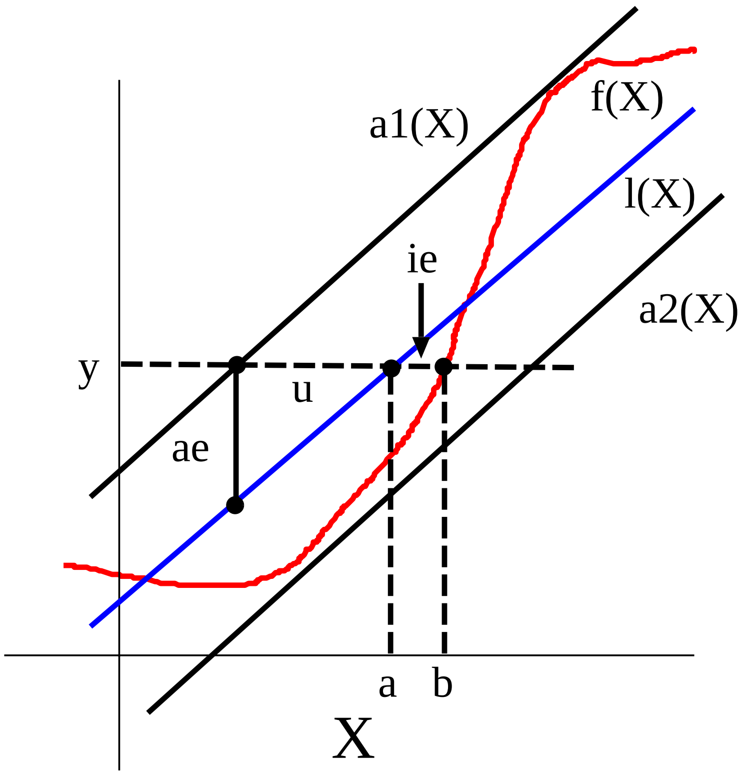
- Assumptions
  - Dataset is 2^n x 2^n x 2^n
  - Periodic boundary conditions
- Per diamond information (32 bits)
  - Data values (8 bits)
  - Gradient value (14 bits)
  - Error Value (6 bits)
  - Min/Max Values (4 bits)
- Error encoding
  - Logarithmic scale
  - Per octree level error codes



- Min/Max encoding
  - Relative to surrounding diamond S
  - 2 bits encode offset from min/max of S

Error Metrics

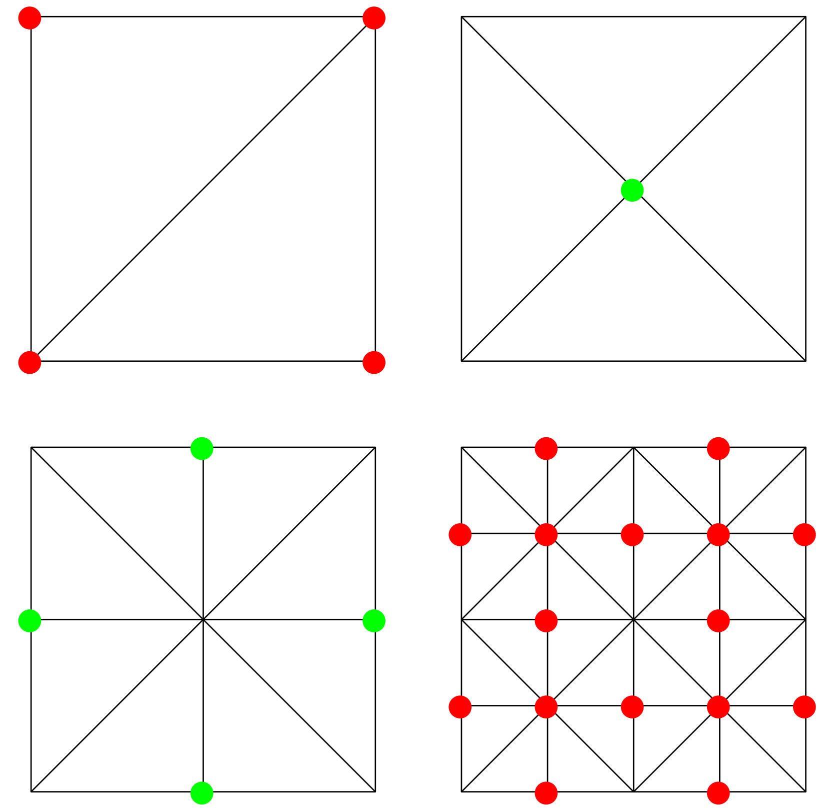
- Approximation error (ae)
  - Different between linear field in tetrahedra and actual data values
- Isosurface error (ie)
  - Deviation of approximated isosurface from actual isosurface
  - Clamped at the size of a tetrahedron
- View-dependent error
  - Projection of isosurface error onto viewscreen
  - Gaze directed error



ie <= ae / gradient magnitude

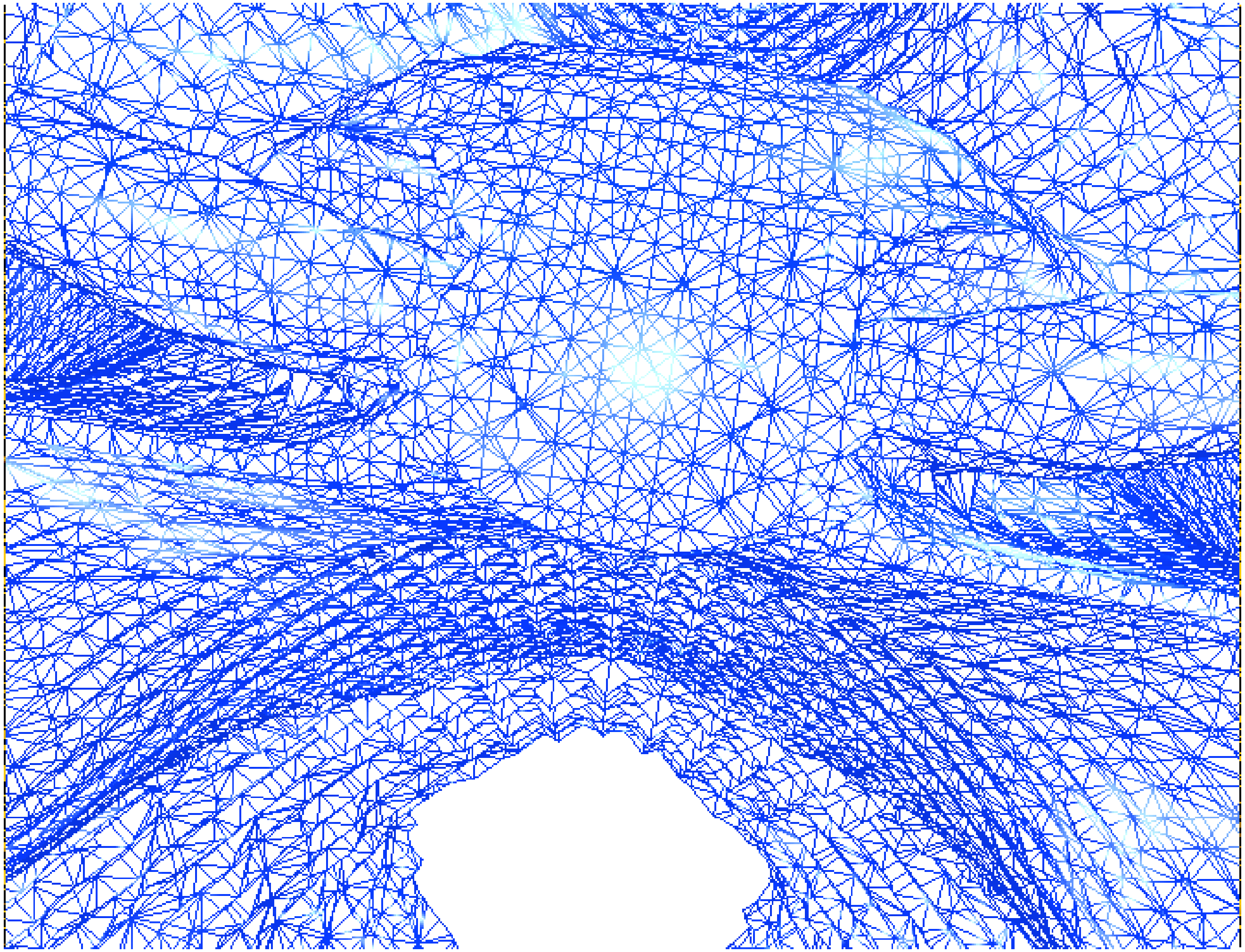
Memory Layout Scheme

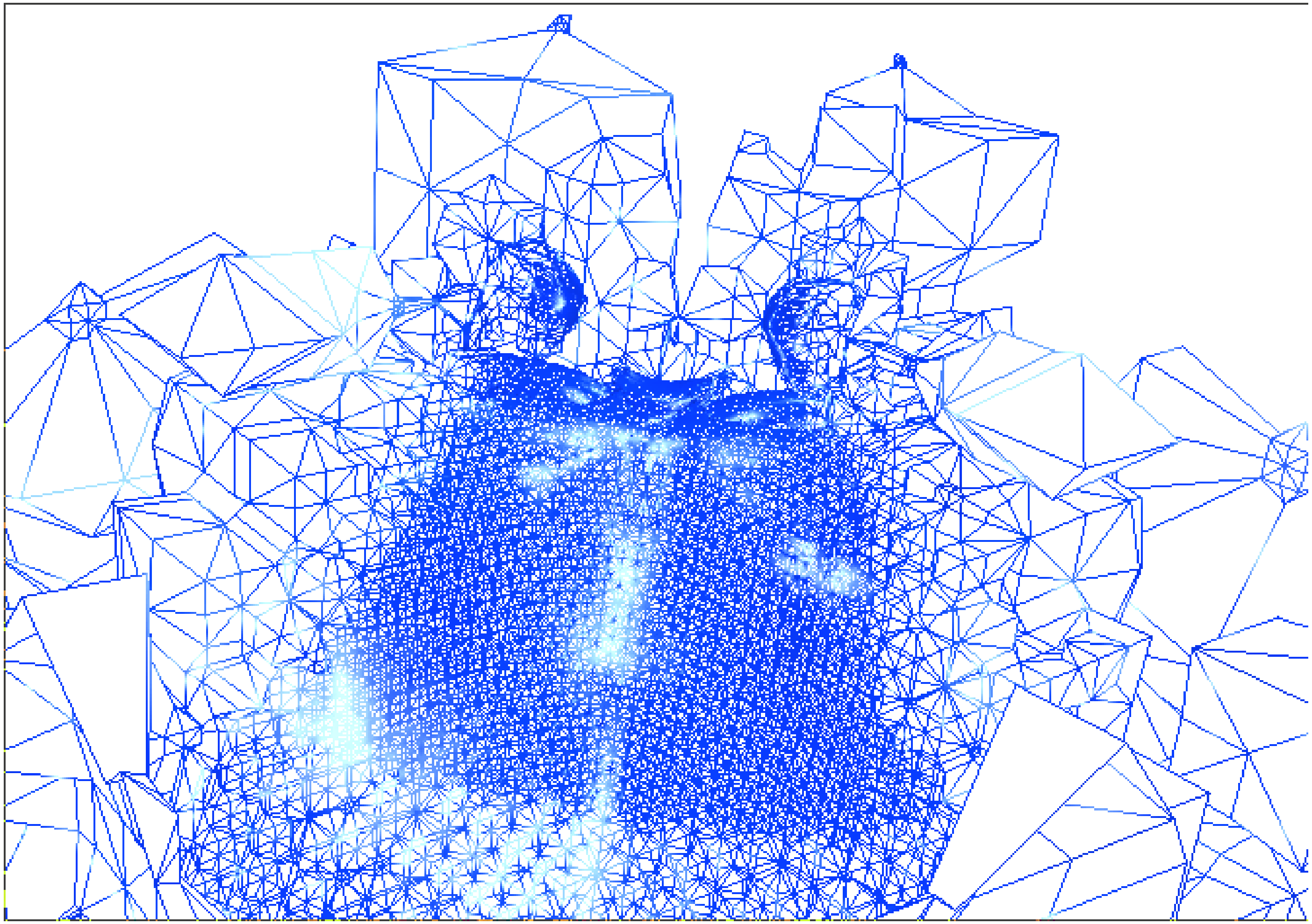
- Memory performance is essential
  - Out-of-core storage scheme
  - Stream in data as it is needed
  - Must be cache coherent
- Data layout follows mesh refinement
  - Good coherence
  - Good disk performance
  - 2D follows quadtree
  - 3D follows octree

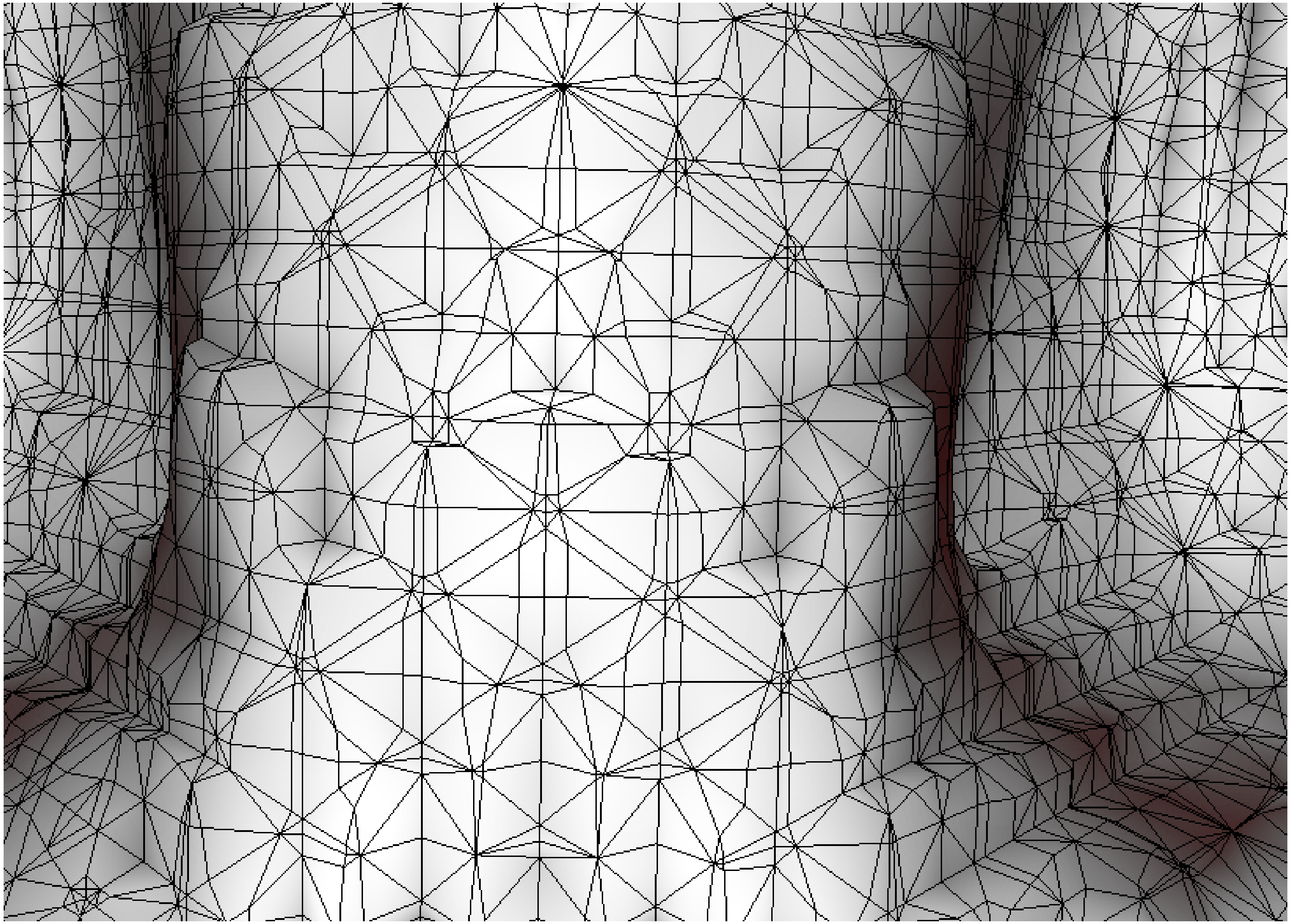


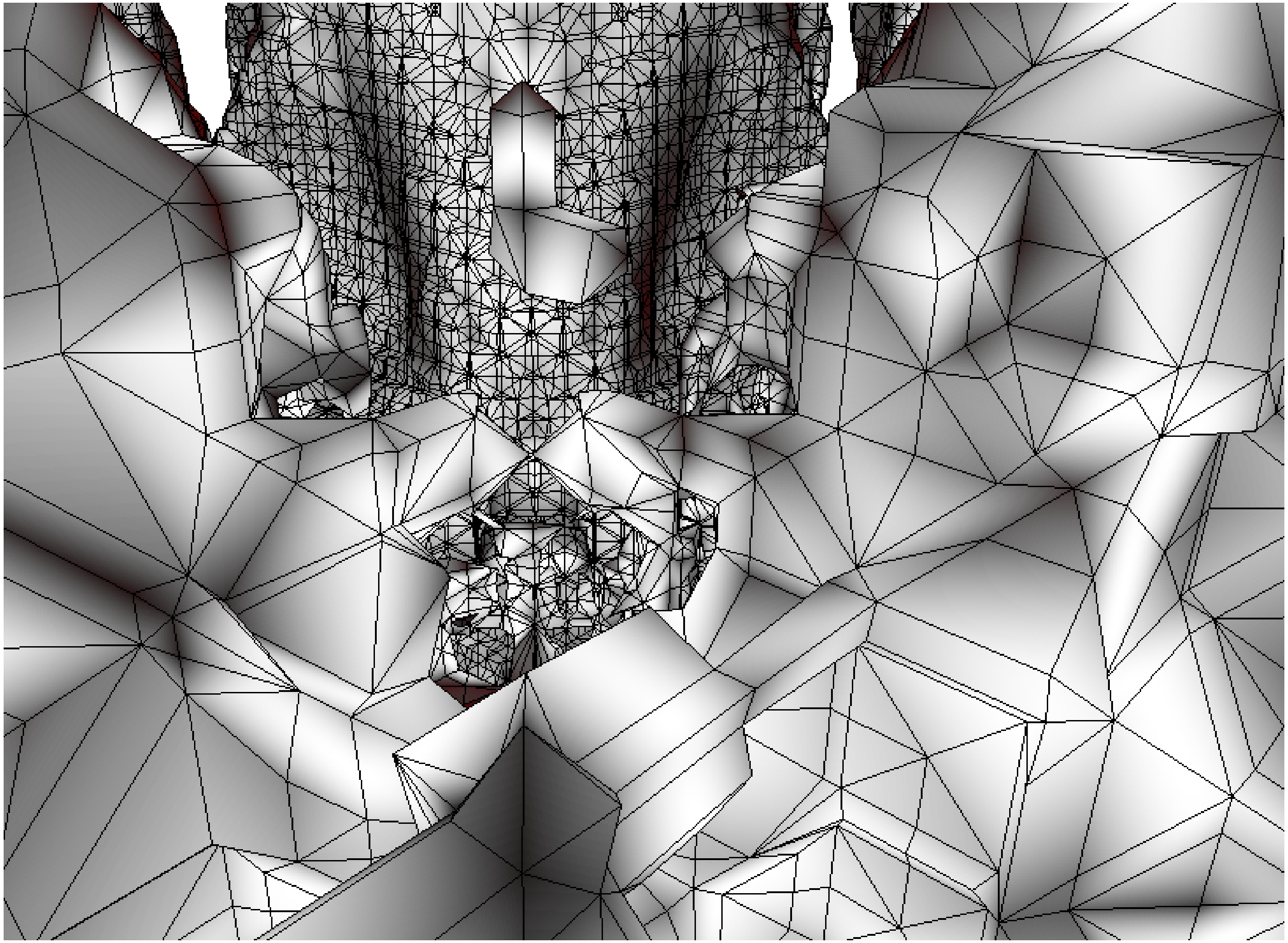
Results

- Richtmyer-Meshkov instability simulation
  - 2Kx2Kx2K x 270 timesteps
  - 8GB per dataset
- Test datasets
  - 512^3 chunks (138MB)
  - Preprocessed data (552MB)
- Performance
  - SGI Onyx with Infinite Reality Graphics
  - Culling/Priority 700K – 1.2e^6 updates per second
  - Drawing 1000K triangles per second
  - Split/Merge 1000 – 4000 per second
  - 250K – 350K triangles per second









Future Work

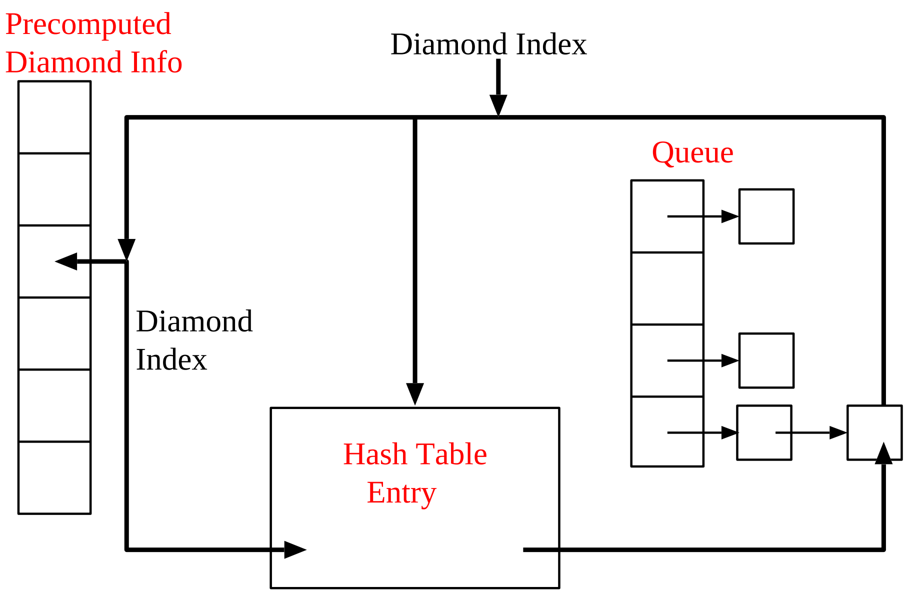
- Time varying data
  - Encoding for frame-to-frame changes
  - Encoding of changes in min/max, errors etc
- Parallelization
  - Threading drawing and mesh refinement
  - Bricking datasets into chunks
- Alternative Rendering Techniques
  - View-dependent volume rendering
  - Hybrid point, isosurface, and volume rendering
  - Shading techniques for isosurfaces
- Higher Order Field Representations
  - Cliffs and discontinuities (similar to material interfaces)
  - Quadratic tetrahedra

Mesh Encoding

- Assume power of 2 grids
  - Vertex represented as (i,j,k) index
- Split edge encoding
  - Split edge (64,64,0) ➡ (0,0,64) with split vertex (32,32,32)
    - Encoding of split edge is (-1,-1,1)
    - Decode by scaling the encoding to diamond's level (level 5)
- Encoding Parents, Children, Tetrahedra
  - Same method as split edge
  - Offsets relative to the diamond's split vertex
  - Stored in a lookup table
- Arbitrary adaptive grids
  - Start indices at level 32 (i.e. 2^32 cubed virtual grid)

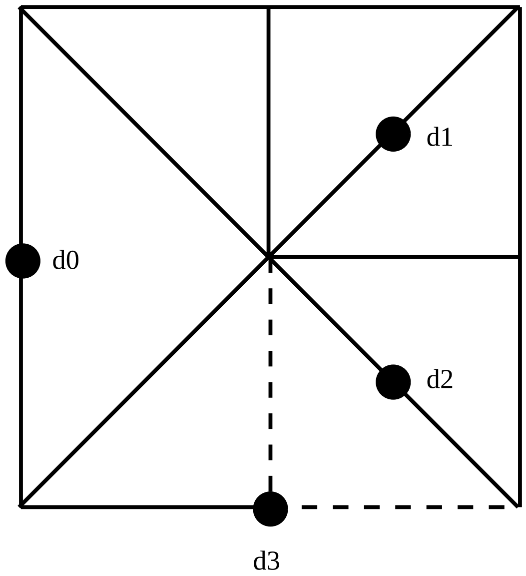
Data Structures

- Split/Merge queues
  - Hash tables on view-dependent error
  - Chaining for collisions
- Queue hash table
  - Maps (i,j,k) indices to queue entries
  - Necessary to lookup parents and children
- Preprocessed data
  - Paged using mmap

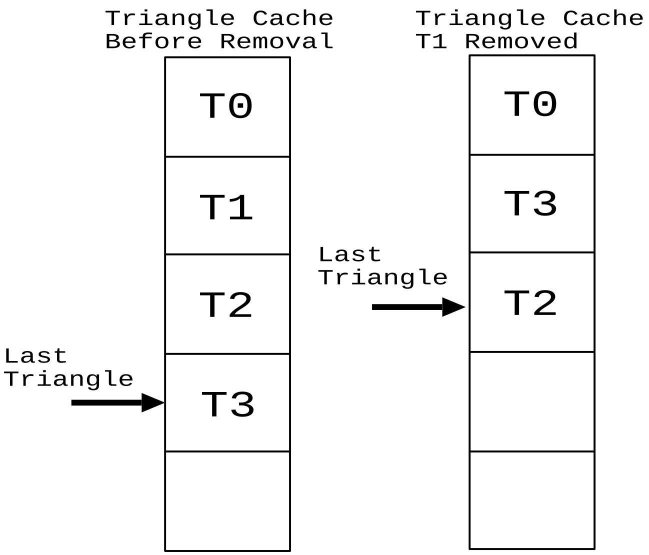


Data Structures

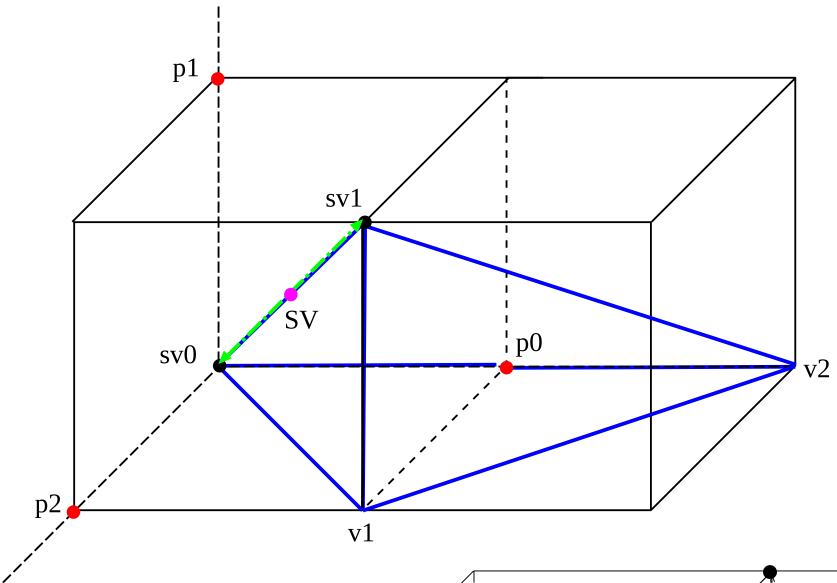
- Tetrahedron Flags
  - Flags indicate which of a diamonds tetrahedra are in the current mesh

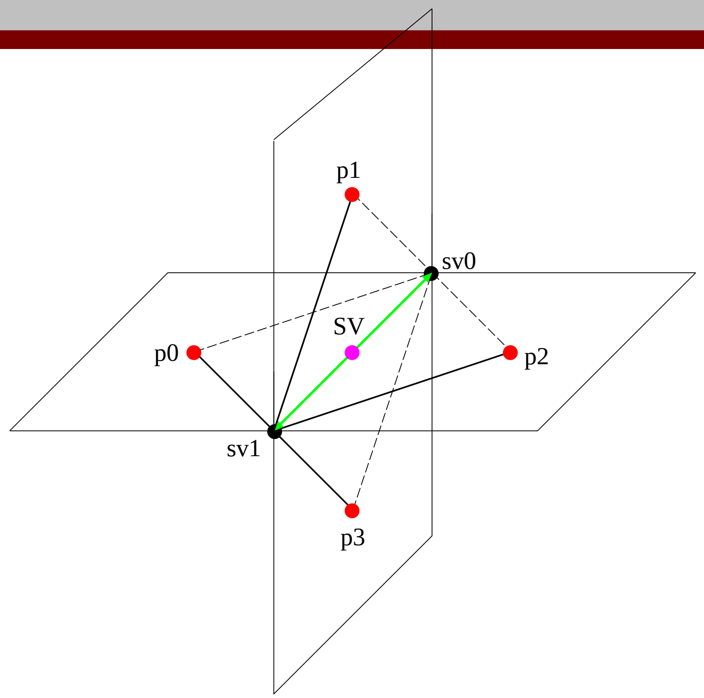


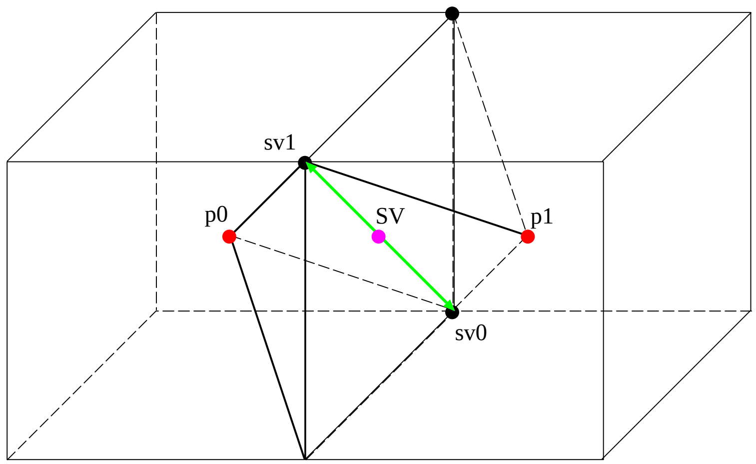
- Triangle Caching
  - Improves performance
  - Geometry cached in array
    - Arrays for tris and quads



Parent Diamonds

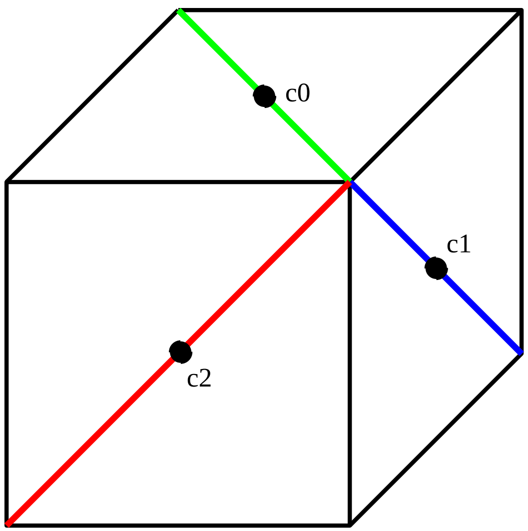


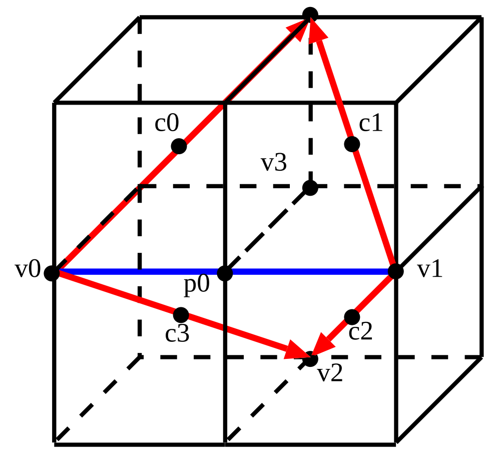


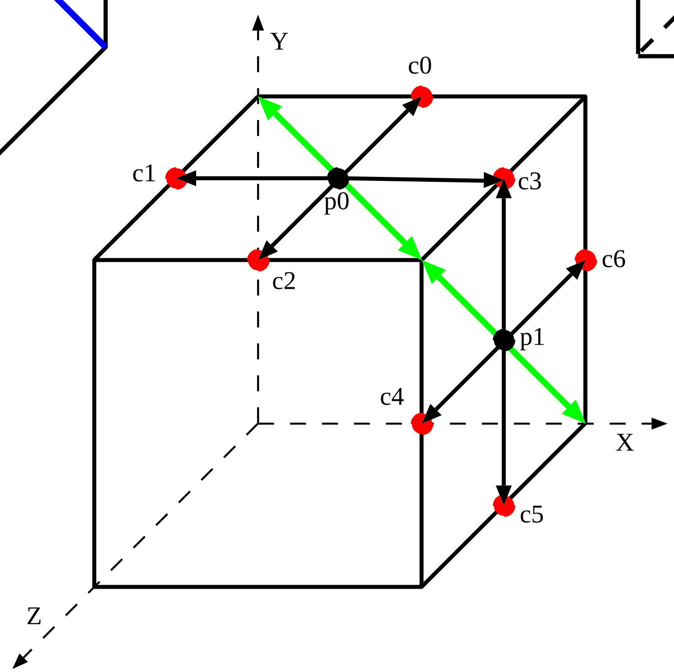


A diamond *D's* parents are the diamonds that must be split to create *D's* tetrahedra

Child Diamonds







A diamond *D's* children are the diamonds created when *D* is split.

Split/Merge Refinement

- Adaptive refinement strategy
  - Priority queues (split queue and merge queue)
  - *Current mesh* is the diamonds in the split queue and their tetrahedra
- Splitting a diamond *D*
  - Recursively split *D's* parents to create all of *D's* tetrahedra
  - Split all of *D's* tets
  - *D* is now on the merge queue
- Merging a diamond *D*
  - Only possible if all of *D's* children are not split
  - Unsplit *D's* tets
  - *D* is now on the split queue
  - Check to see if *D's* parents can be put on the merge queue