Data Visualization 2

Yasuhide Okamoto

Digest

- Huge 3D data
  - Easy to obtain, Hard to visualize

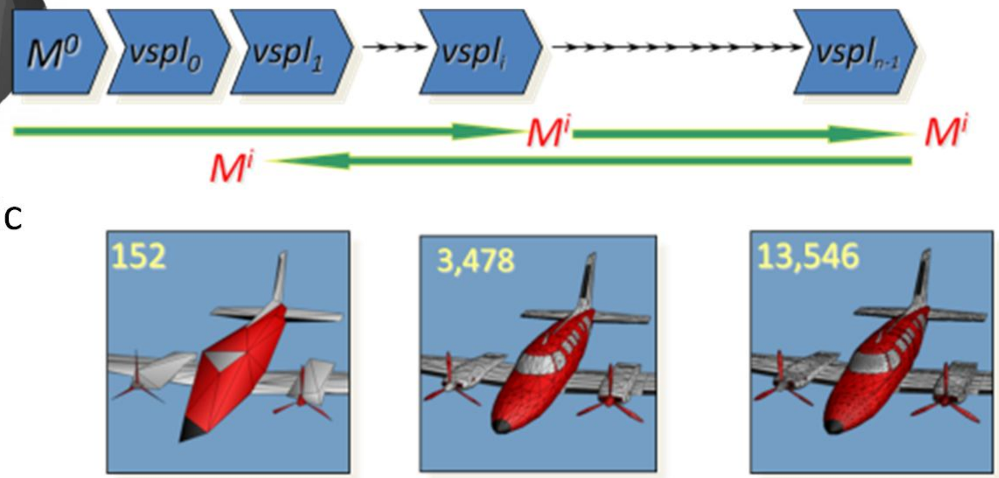
Good Sensors

Huge 3D Data

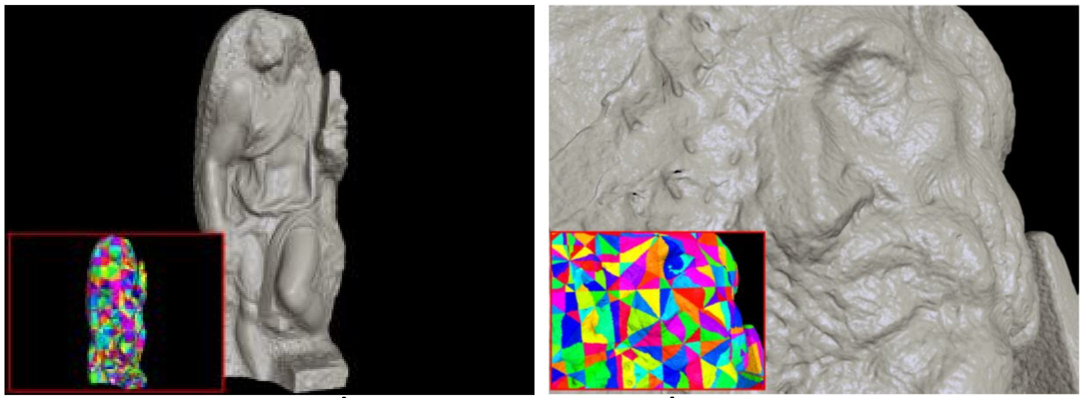
Too complex to process

Solutions

Simplification using Quadric Error Metric

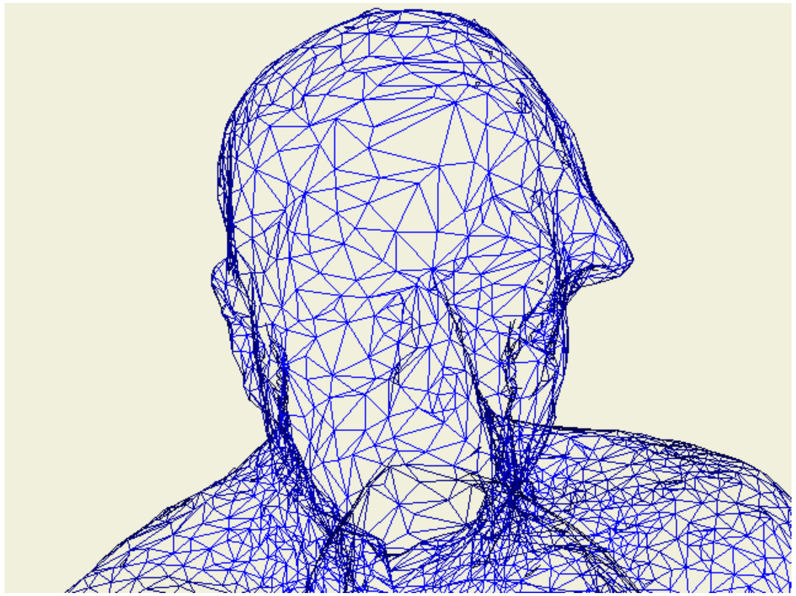


Progressive Meshes



Adaptive TetraPuzzles

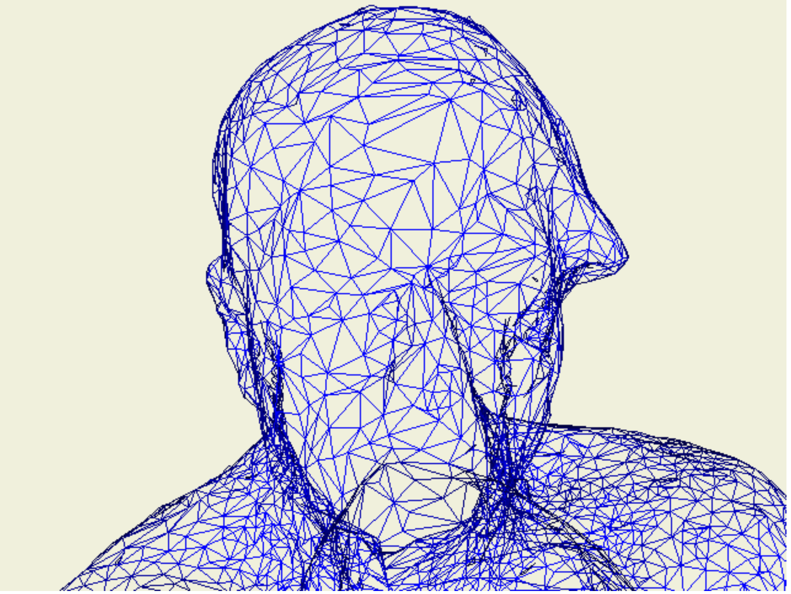
3D representation



Why we use triangles?

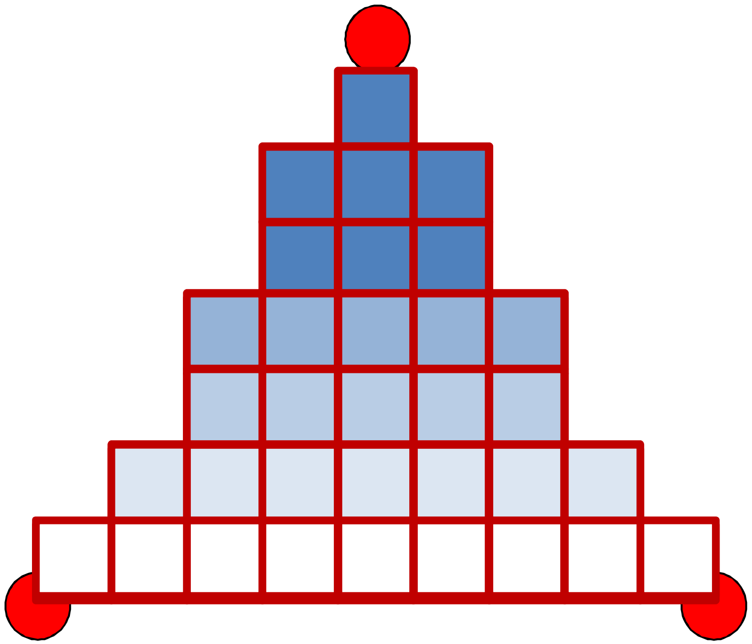
- Planar
- Convex
- Simplest polygon

- Possible to represent accurate surfaces

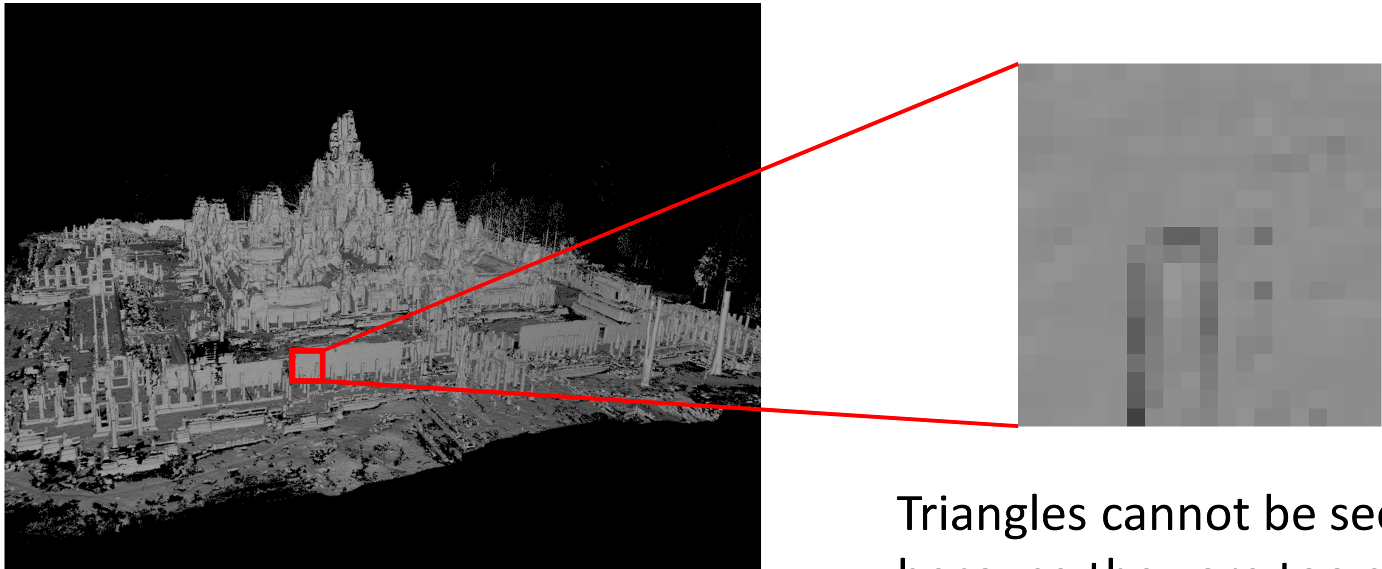


Rendering a triangle

1. Transformation of vertices
2. Rasterization
3. Texturing, Pixel shading



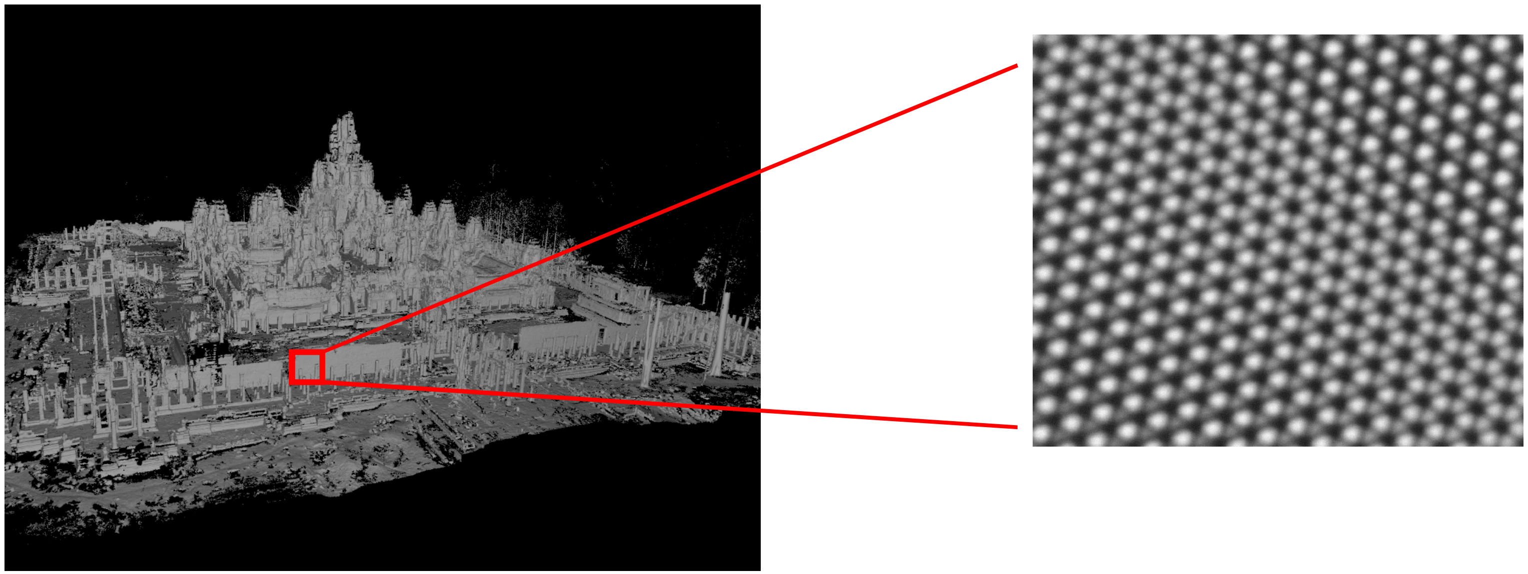
Triangle is the only solution?



Triangles cannot be seen
because they are too small…

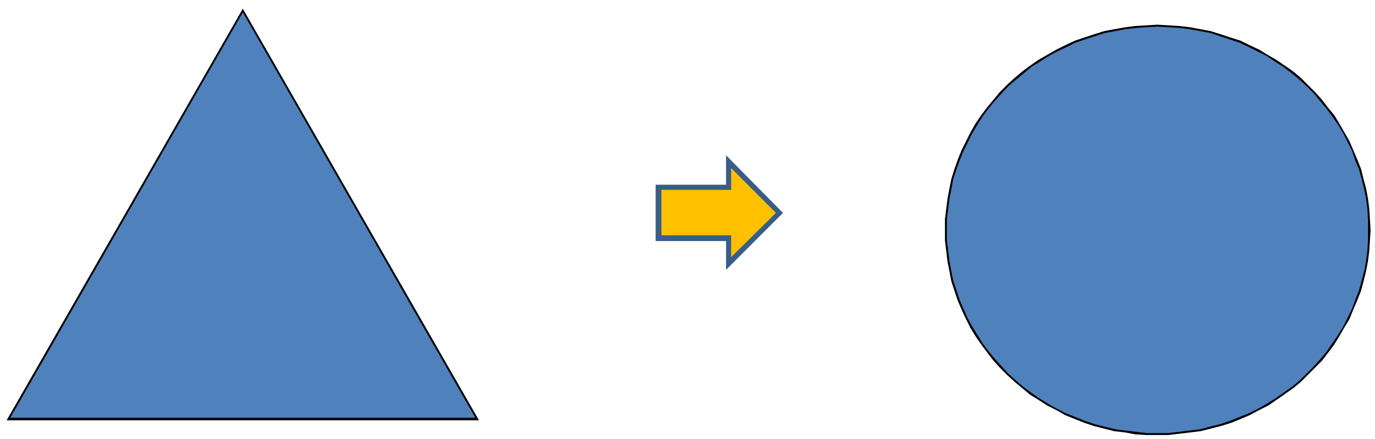
Another solution

- Everything is built up of atoms…



Another solution

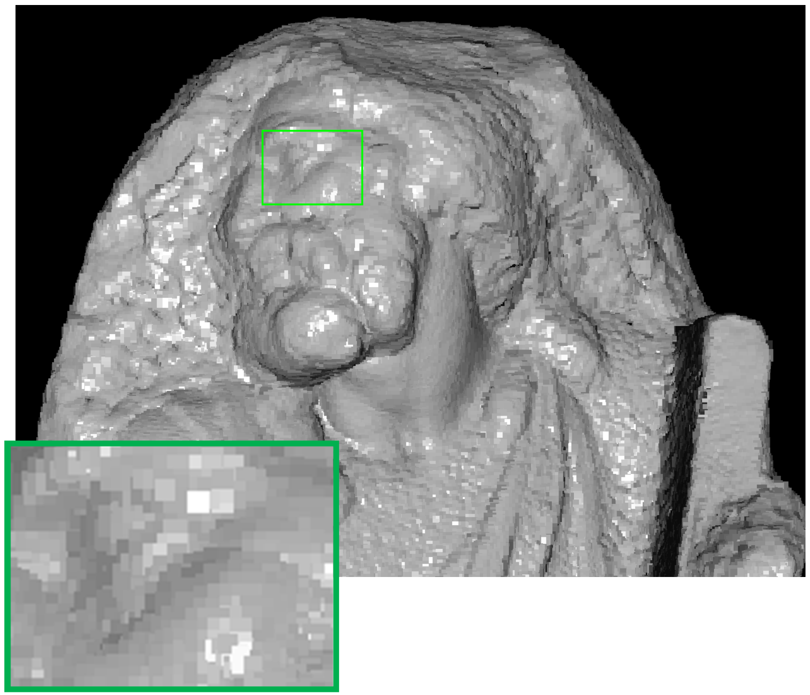
- Triangle -> Point



Another solution

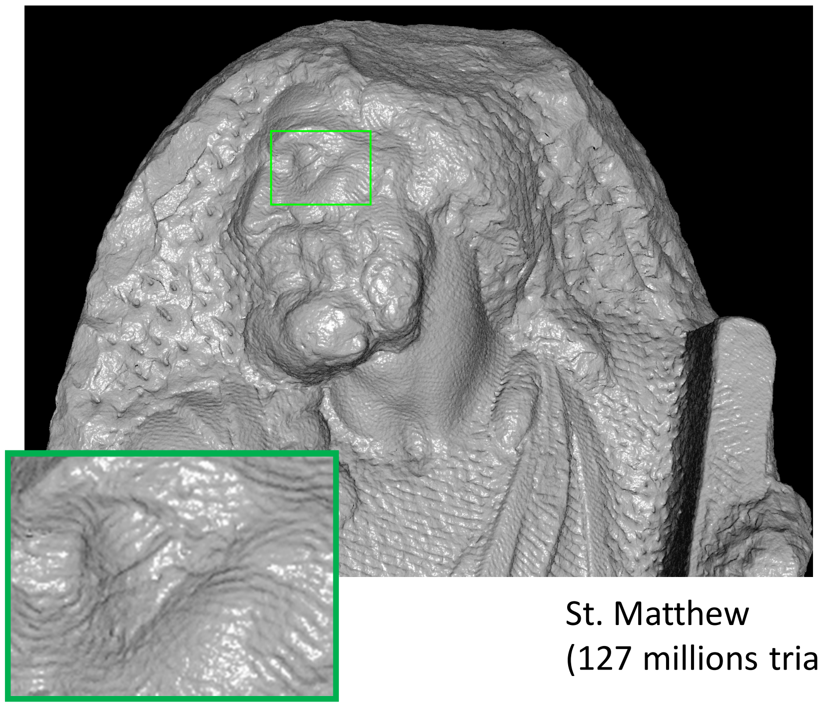
- Point based rendering

Points



8 frame/sec

Triangles

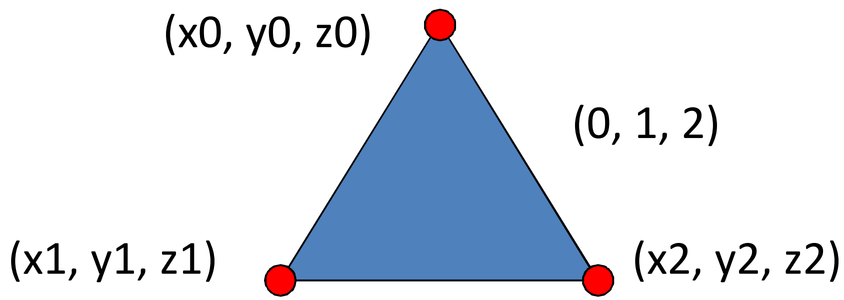


St. Matthew
(127 millions triangles)

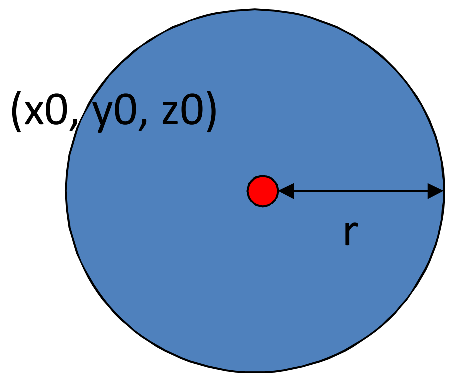
8 sec/frame

Merits of points

- Data simplicity
  - Triangles
    - 3 vertices
    - 1 set of indices

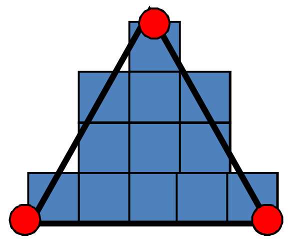


  - Points
    - 1 vertex
    - point size
    - (no connectivity)

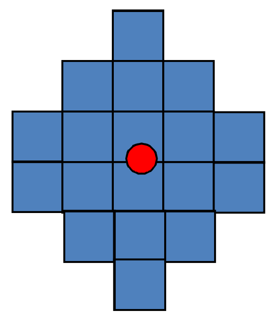


Merits of points

- Processing cost
  - Triangles
    - Projection of 3 vertices
    - Precise rasterization

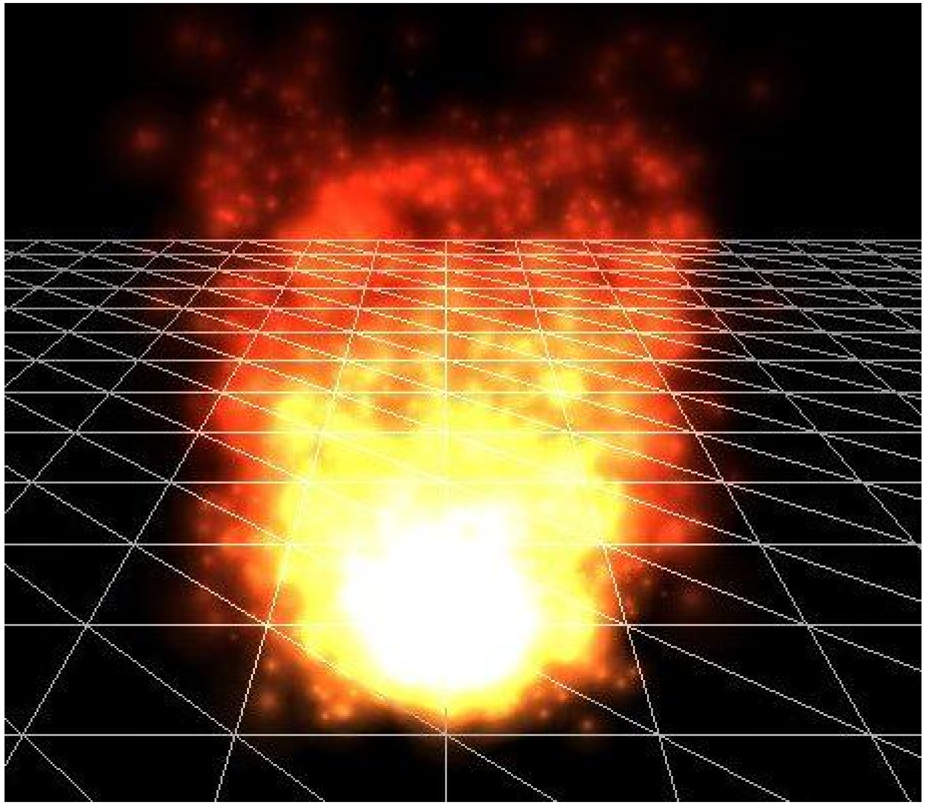
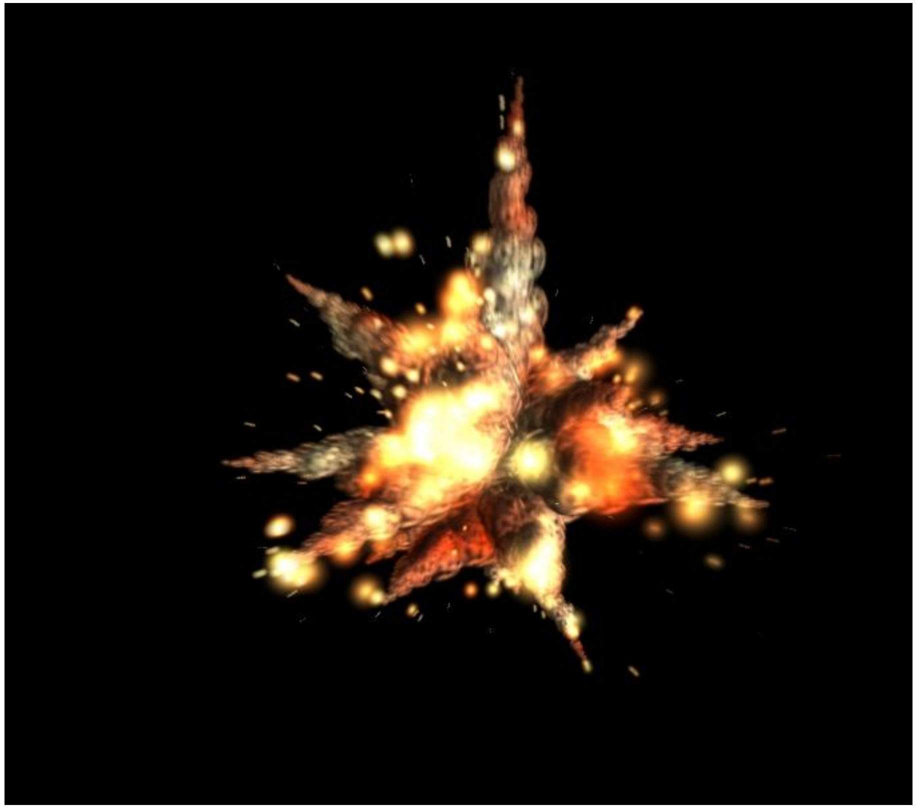


  - Points
    - Projection of 1 vertex
    - Simple rasterization



Applications of point rendering

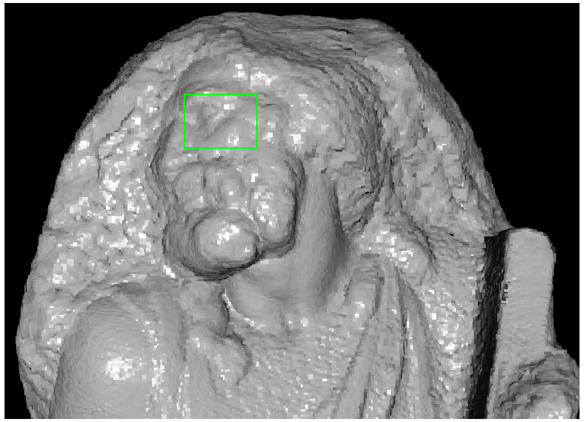
- Particle system

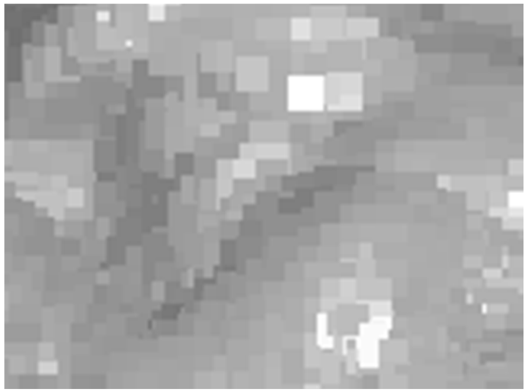



Demerit of point rendering

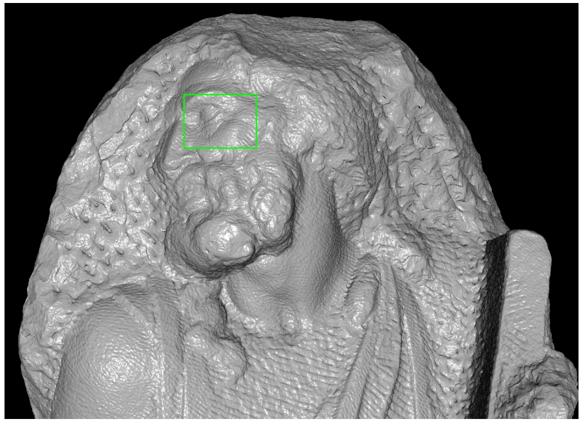
- Rendering quality

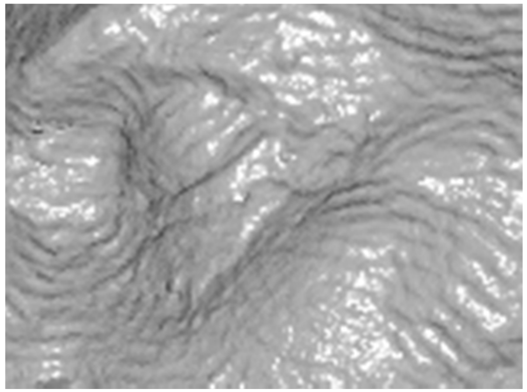
Points





Triangles

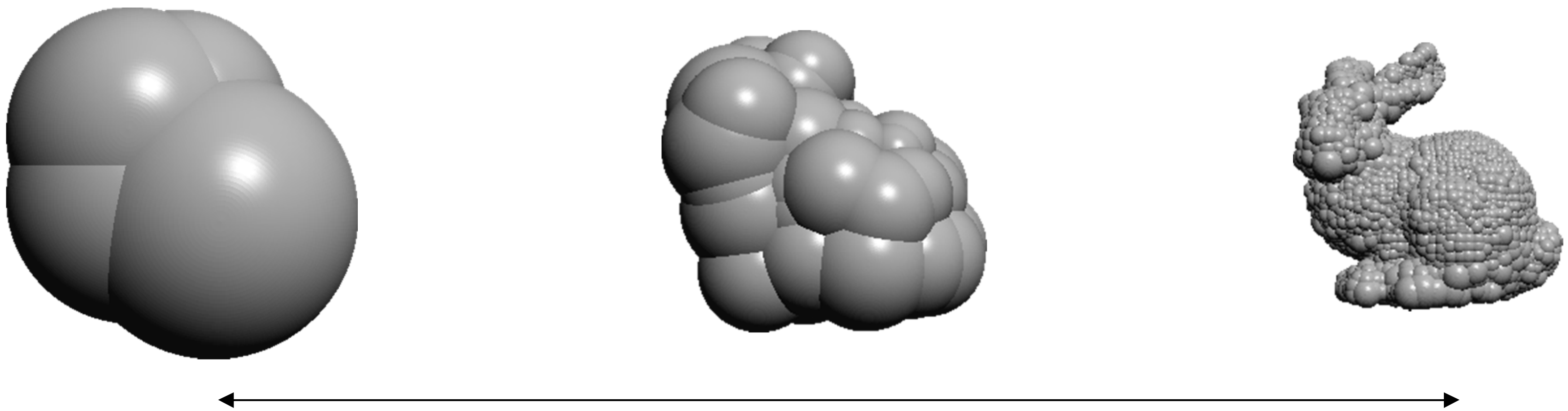




QSplat

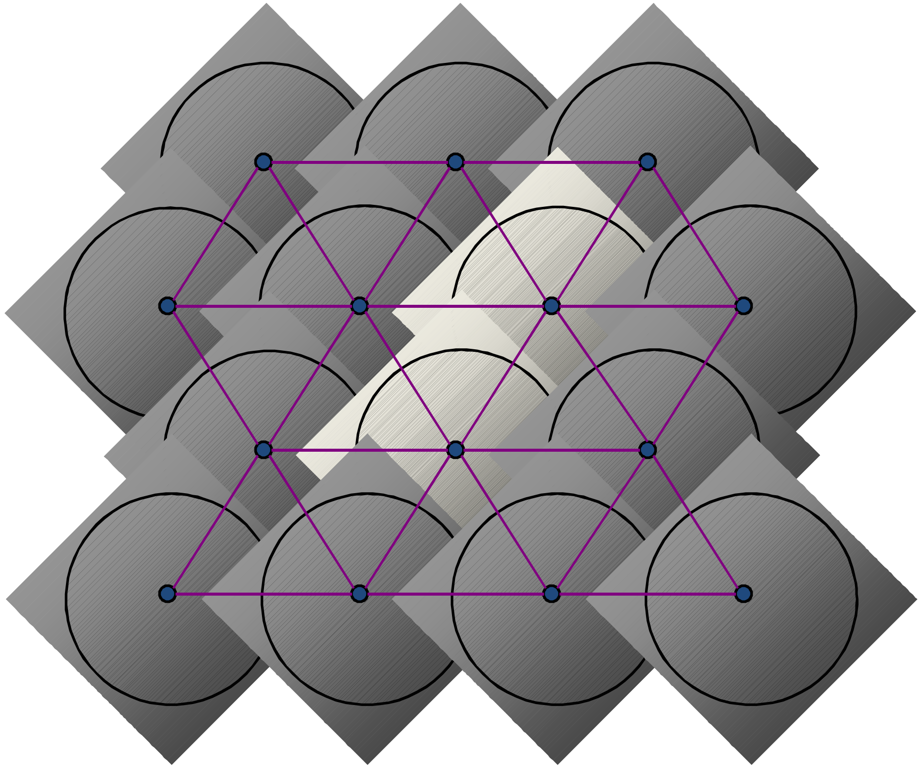
Szymon Rusinkiewicz, Marc Levoy
SIGGRAPH2000

- Point based rendering with LOD
  - Bounding sphere hierarchy



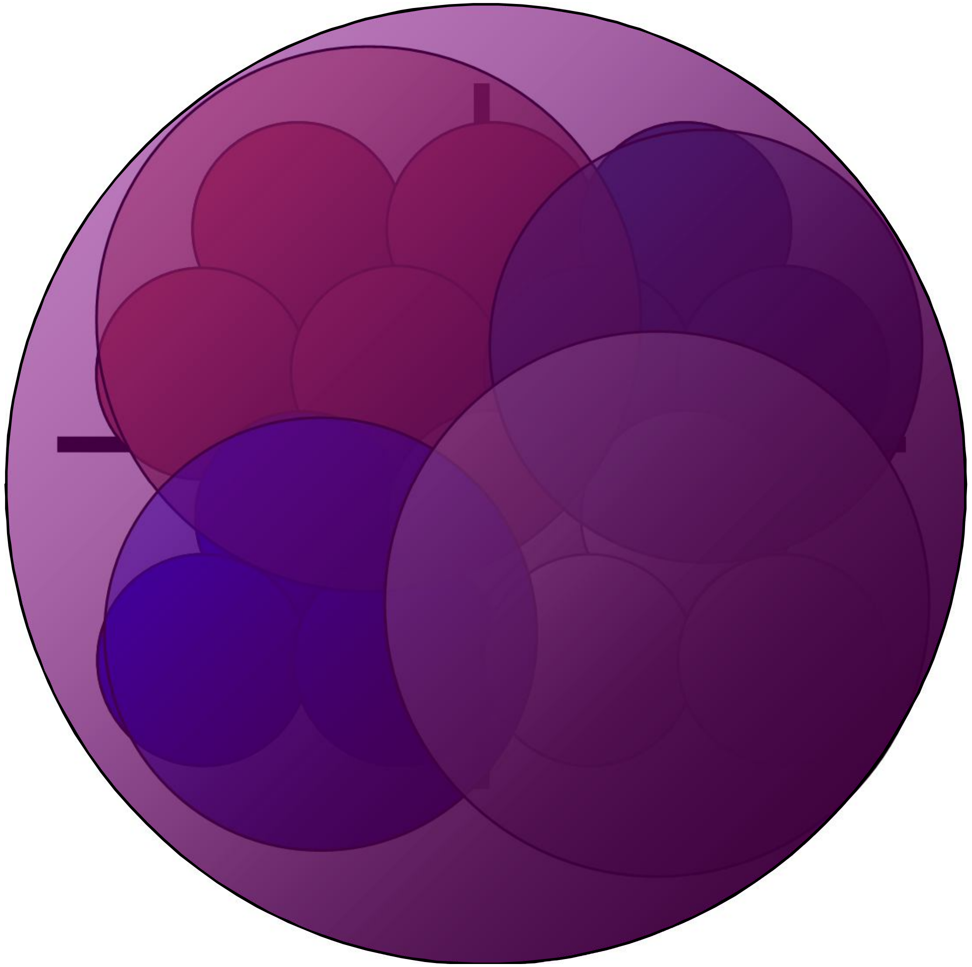
Data construction

- Input a triangle mesh
- Place a sphere at each triangle, large enough to touch neighbor spheres

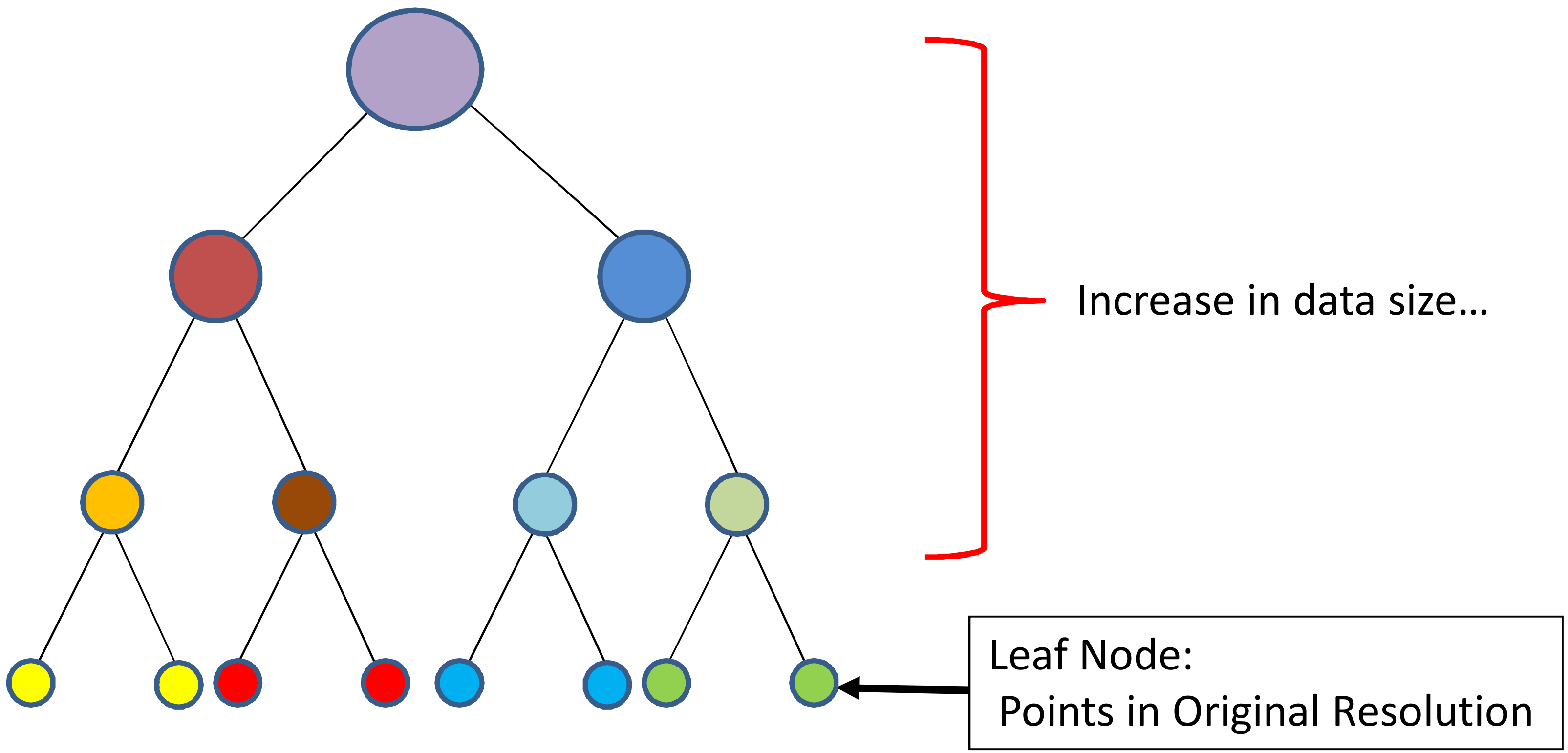


Creating a hierarchy

- Recursive splitting and merging

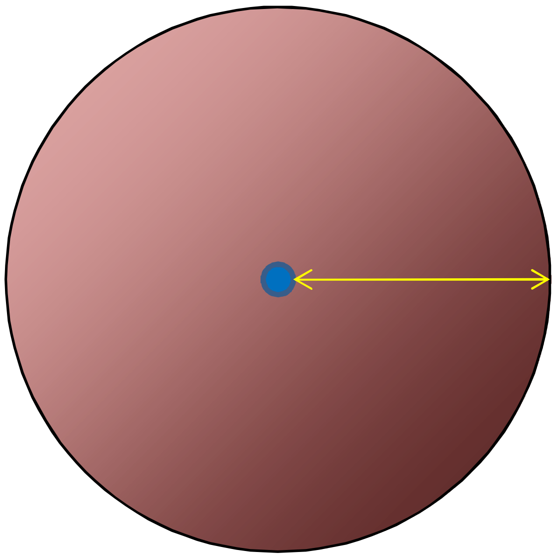


Point tree

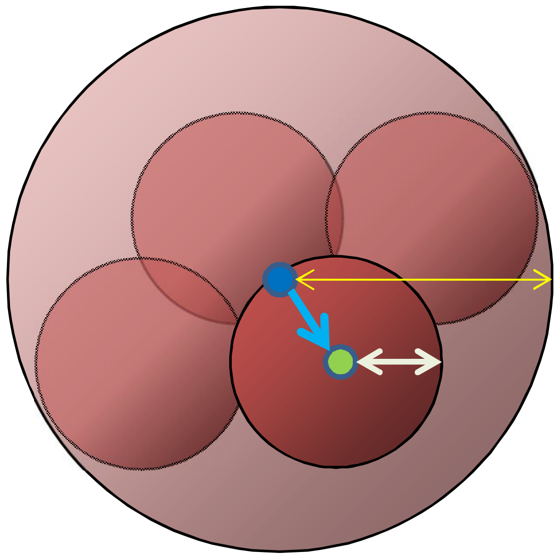

Increase in data size…
Leaf Node:
Points in Original Resolution

Hierarchical data compression

- Using relative values to the parent node
- Quantization



Parent Node:
Center (x_p, y_p, z_p), Radius: r_p



Child Node:
Center $(x_p+k_x*r, y_p+k_y*r, z_p+k_z*r)$, Radius: $r*k_r$

Node structure

Position and Radius	Tree Structure	Normal	Width of Cone of Normals	Color (Optional)
13 bits	3 bits	14 bits	2 bits	16 bits

6 bytes

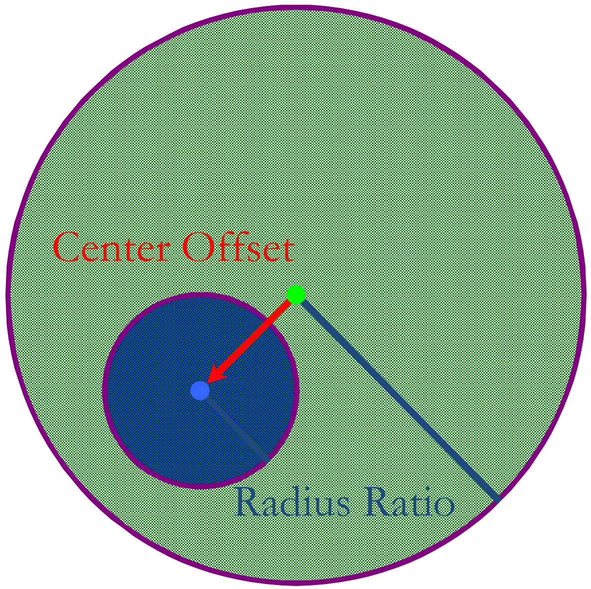
Position and radius

Position and Radius	Tree Structure	Normal	Width of Cone of Normals	Color (Optional)
13 bits	3 bits	14 bits	2 bits	16 bits

- Position and radius encoded relative to parent node

(x, y, z, r) are represented as $\frac{1}{13}$ to $\frac{13}{13}$

only 7621 combinations are valid, not 13^4



Tree structure

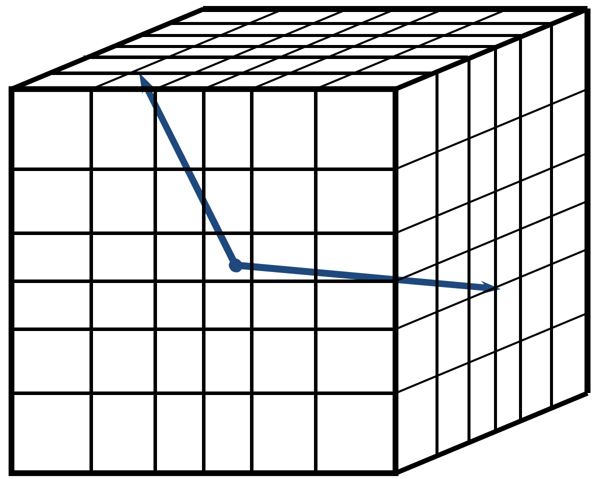
Position and Radius	Tree Structure	Normal	Width of Cone of Normals	Color (Optional)
13 bits	3 bits	14 bits	2 bits	16 bits

- Number of children (0, 2, 3, or 4) – 2 bits
- Presence of grandchildren – 1 bit

Normal

Position and Radius	Tree Structure	Normal	Width of Cone of Normals	Color (Optional)
13 bits	3 bits	14 bits	2 bits	16 bits

- Normal quantized to grid on faces of a cube

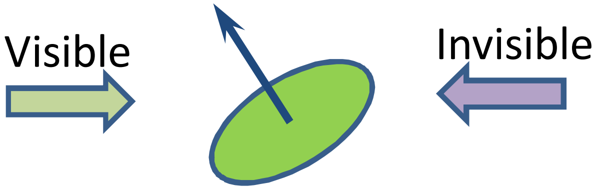


$52 \times 52 \times 6$

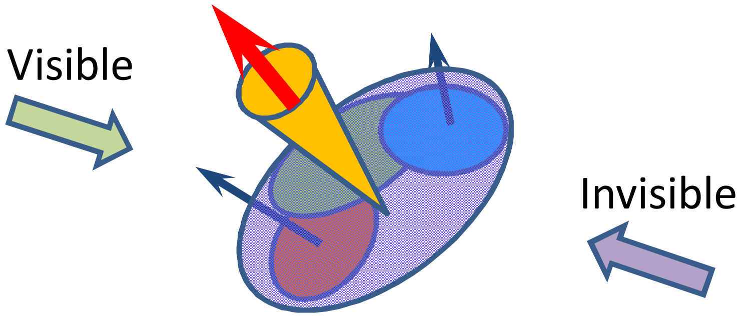
Normal cone

Position and Radius	Tree Structure	Normal	Width of Cone of Normals	Color (Optional)
13 bits	3 bits	14 bits	2 bits	16 bits

- Normal cone: range of visible directions



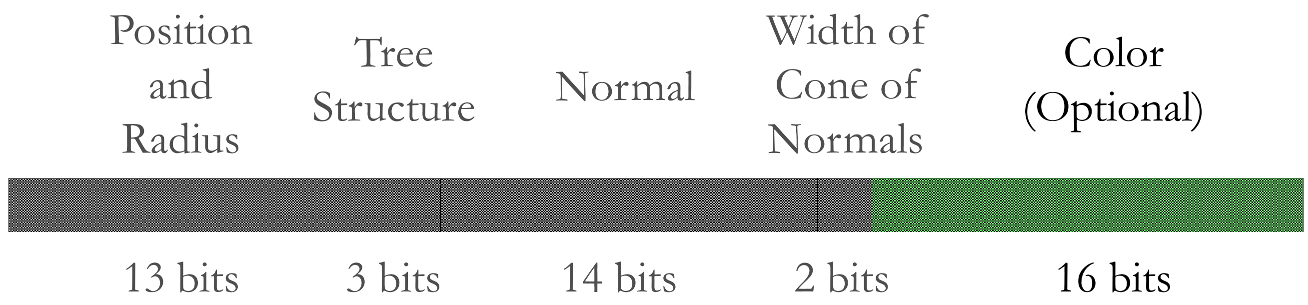
Original Point
(leaf node)



Inner node's point

- Quantizing cone's width to $\frac{1}{16}, \frac{4}{16}, \frac{9}{16}, \frac{16}{16}$

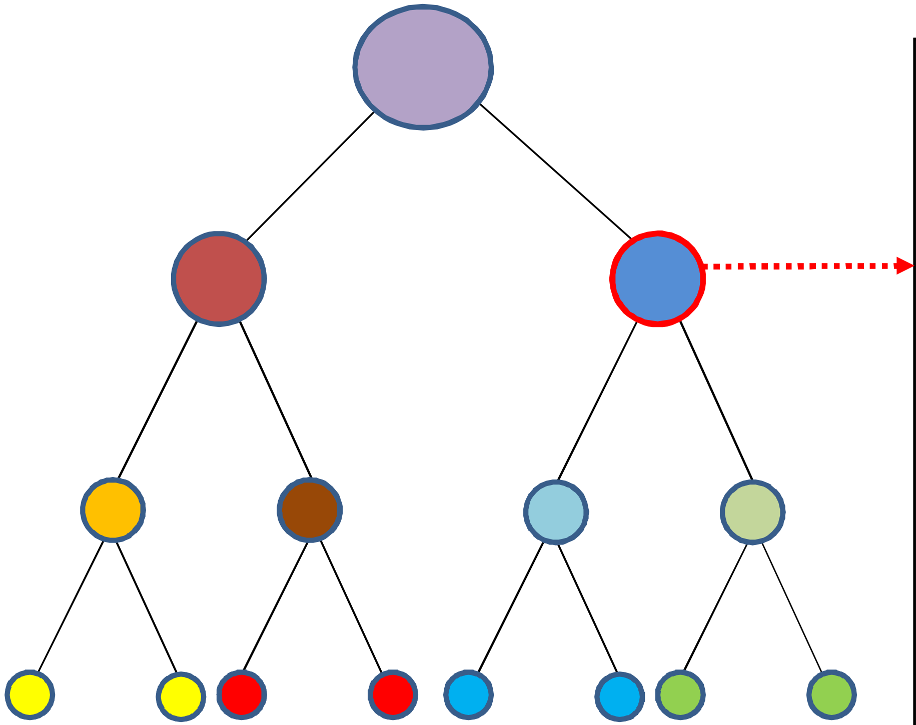
Color



- Per-vertex color is quantized 5-6-5 (R-G-B)

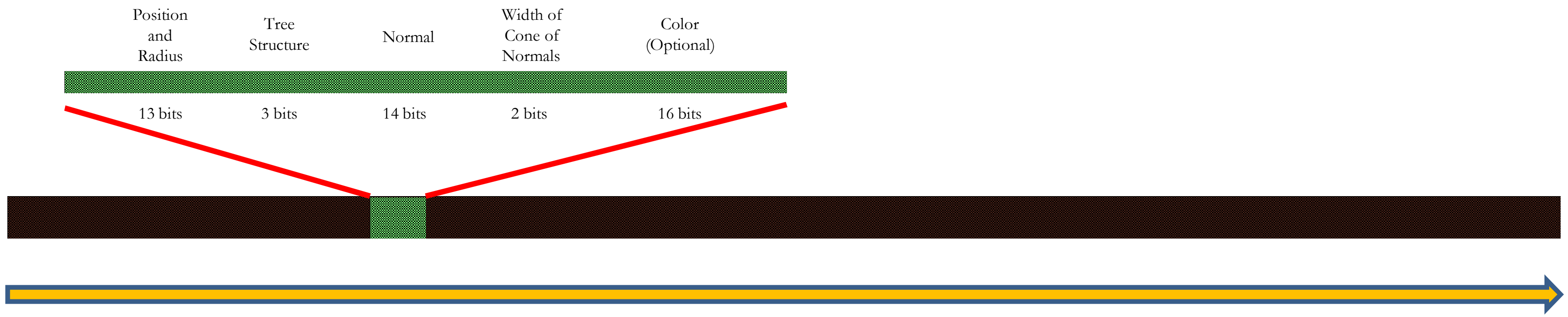
Rendering algorithm

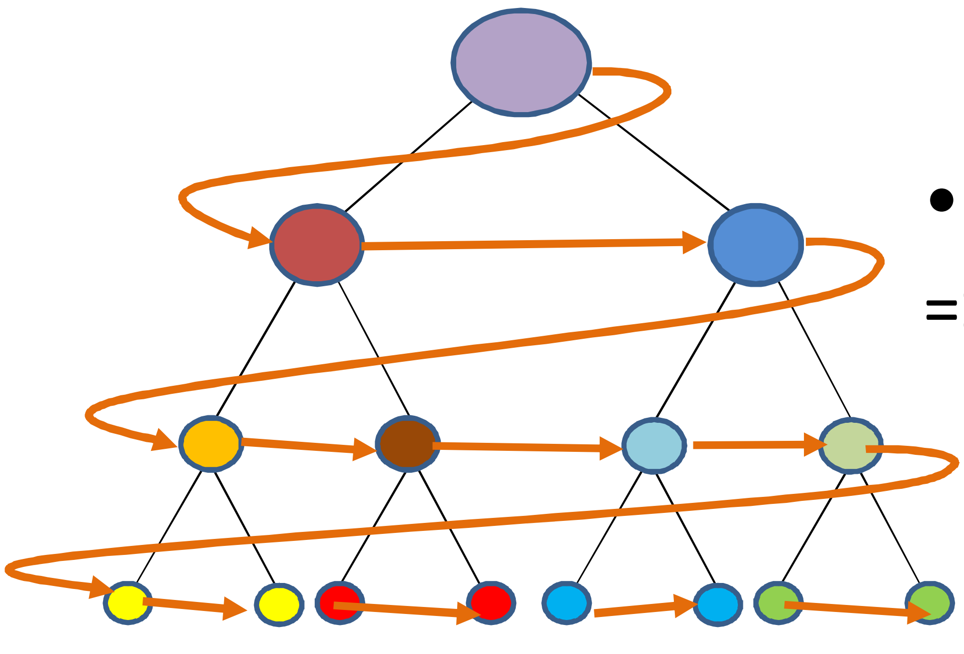
- Traverse hierarchy recursively



```
if (node not visible)
        Skip this subtree
else if (leaf node)
        Draw points
else if (size on screen < threshold)
        Draw points
else
        Traverse children
```

Data alignment





- Breadth-First Order

=> Good for memory coherency

Results of data construction

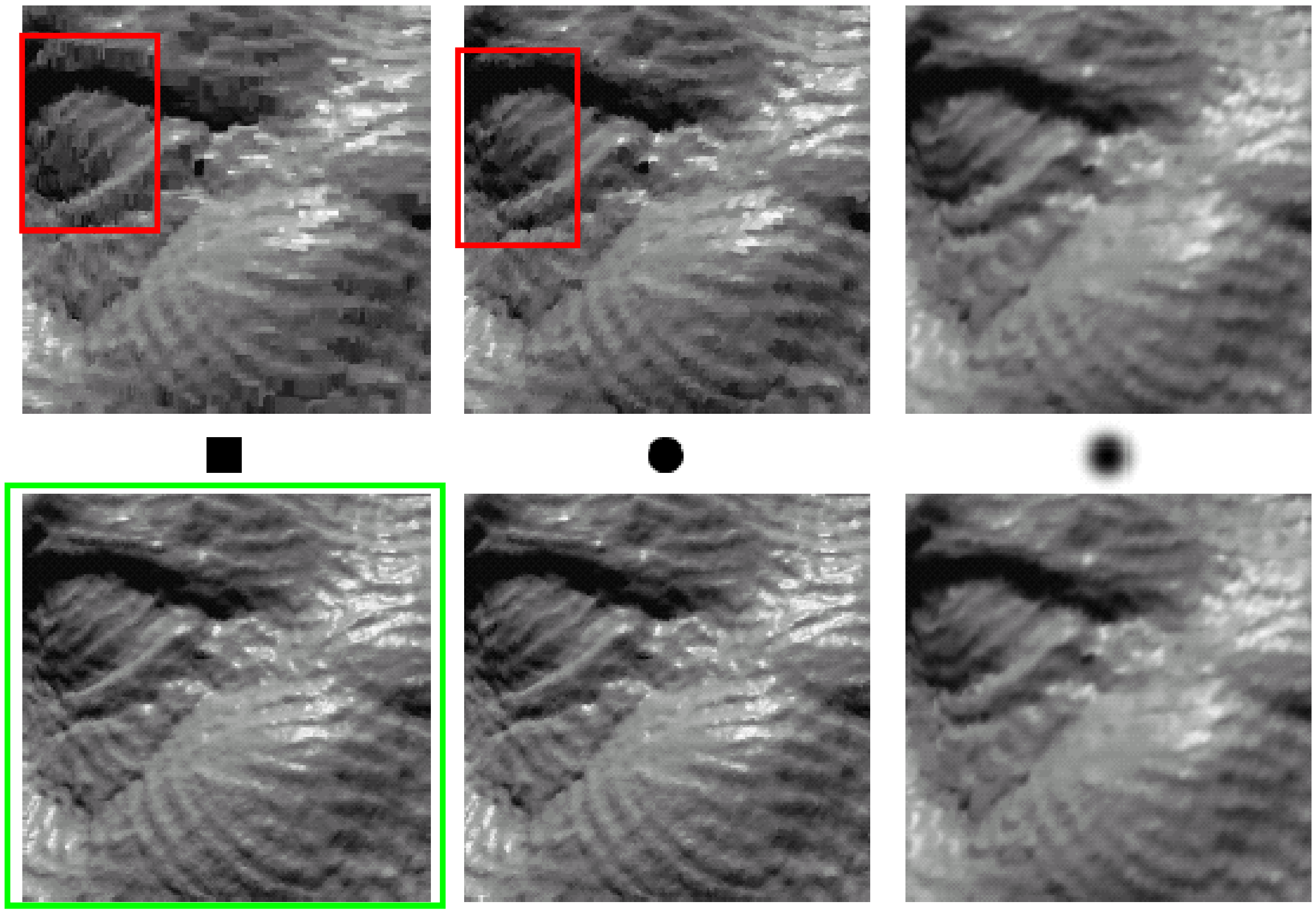
Model	Input points	Interior Nodes	Data construction (min)	Output size (MB)
David's head	2,000,651	974,114	0.7	18
David (2mm)	4,251,890	2,068,752	1.4	27
St. Matthew	127,072,827	50,285,122	59	761

Comparing splat shapes

aliasing

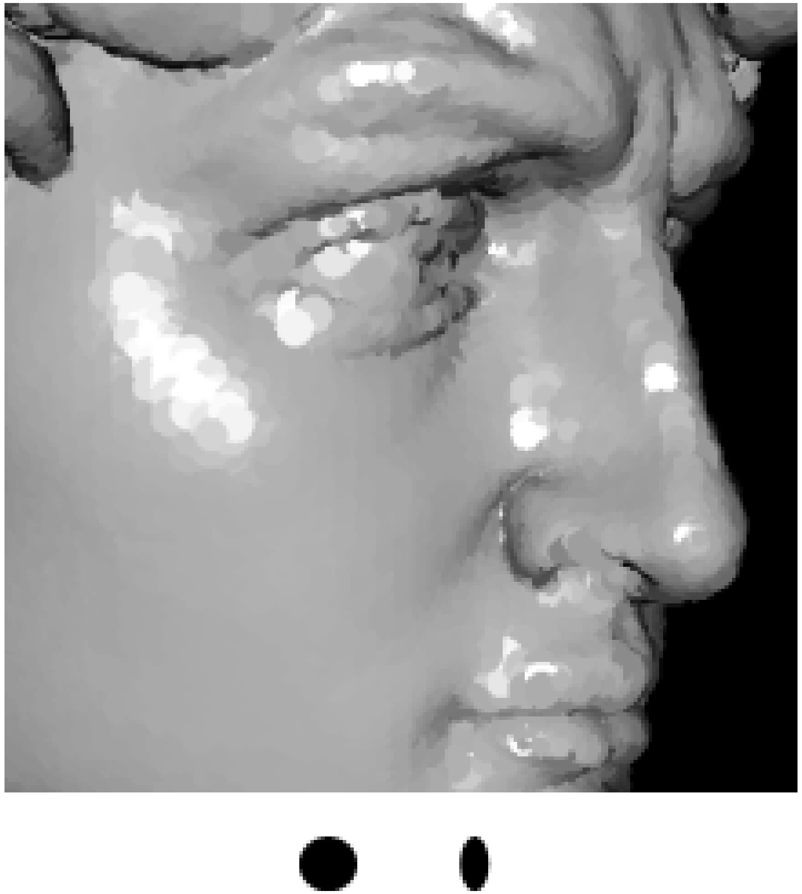
Same recursion level

Same rendering time



Highest quality

Comparing splat shapes



Demonstration video

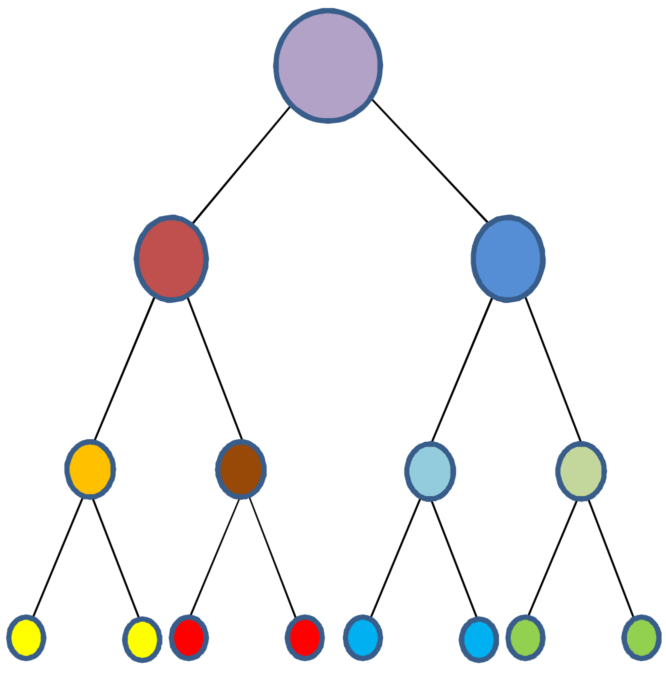
Summary of QSplat

- Hierarchical representation using points

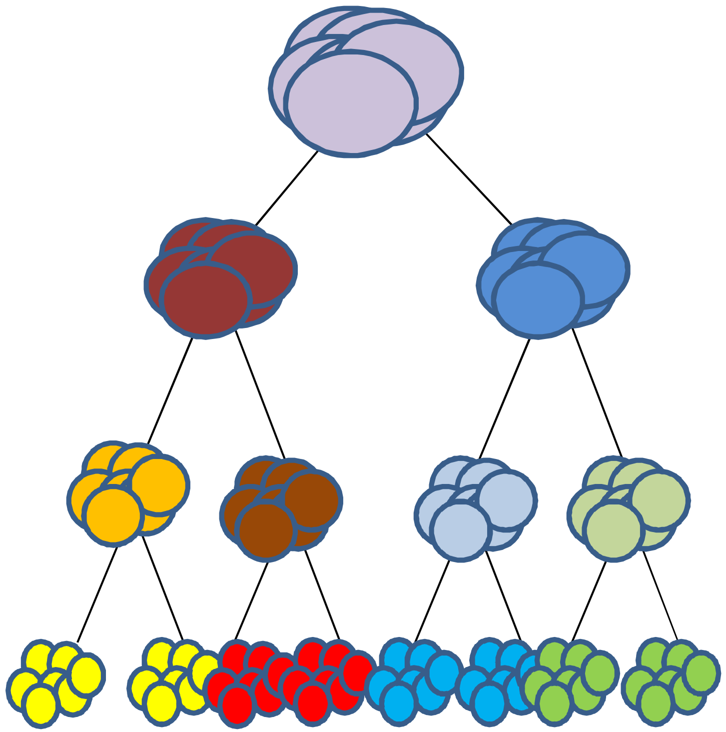
- High speed and high quality rendering for complex models
- Fast preprocessing
- High compression rate

Similar works

- Layered Point Clouds



QSplat --- Point based Tree
Tree Depth: log(n)



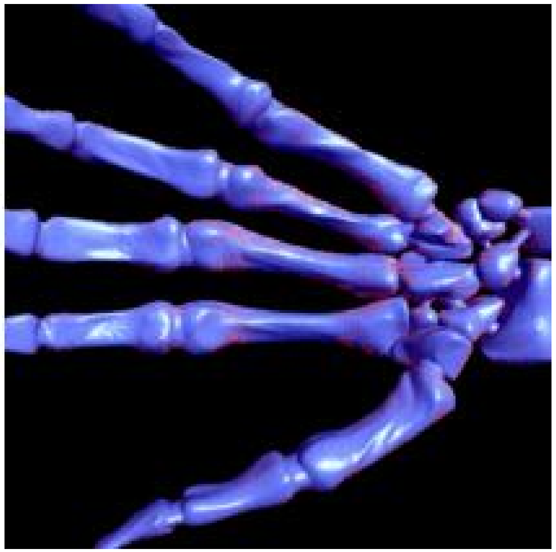
Layered Point Clouds --- Point Cloud based Tree
Tree Depth: log(n/k)

Similar works

- Layered Point Clouds

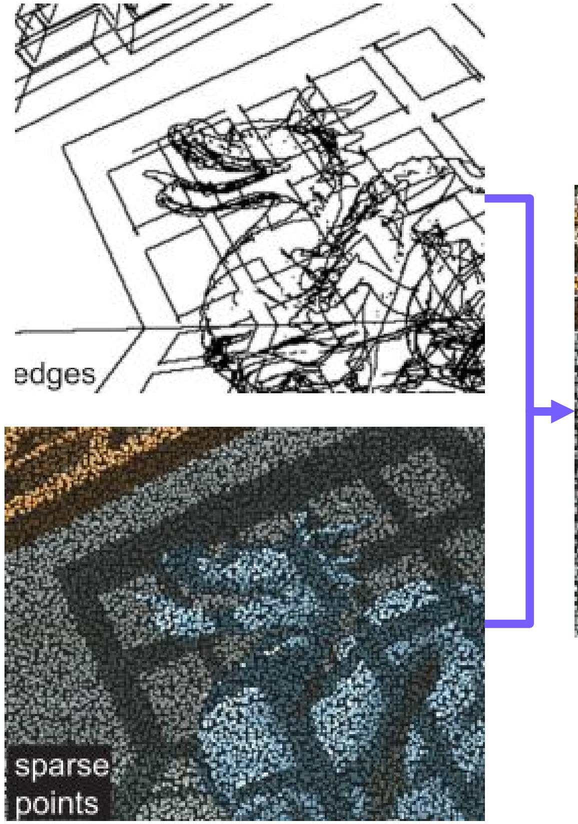
Similar works

- POP
  - Hybrid method using Points and Polygons



Similar works

- Combining Edges and Points

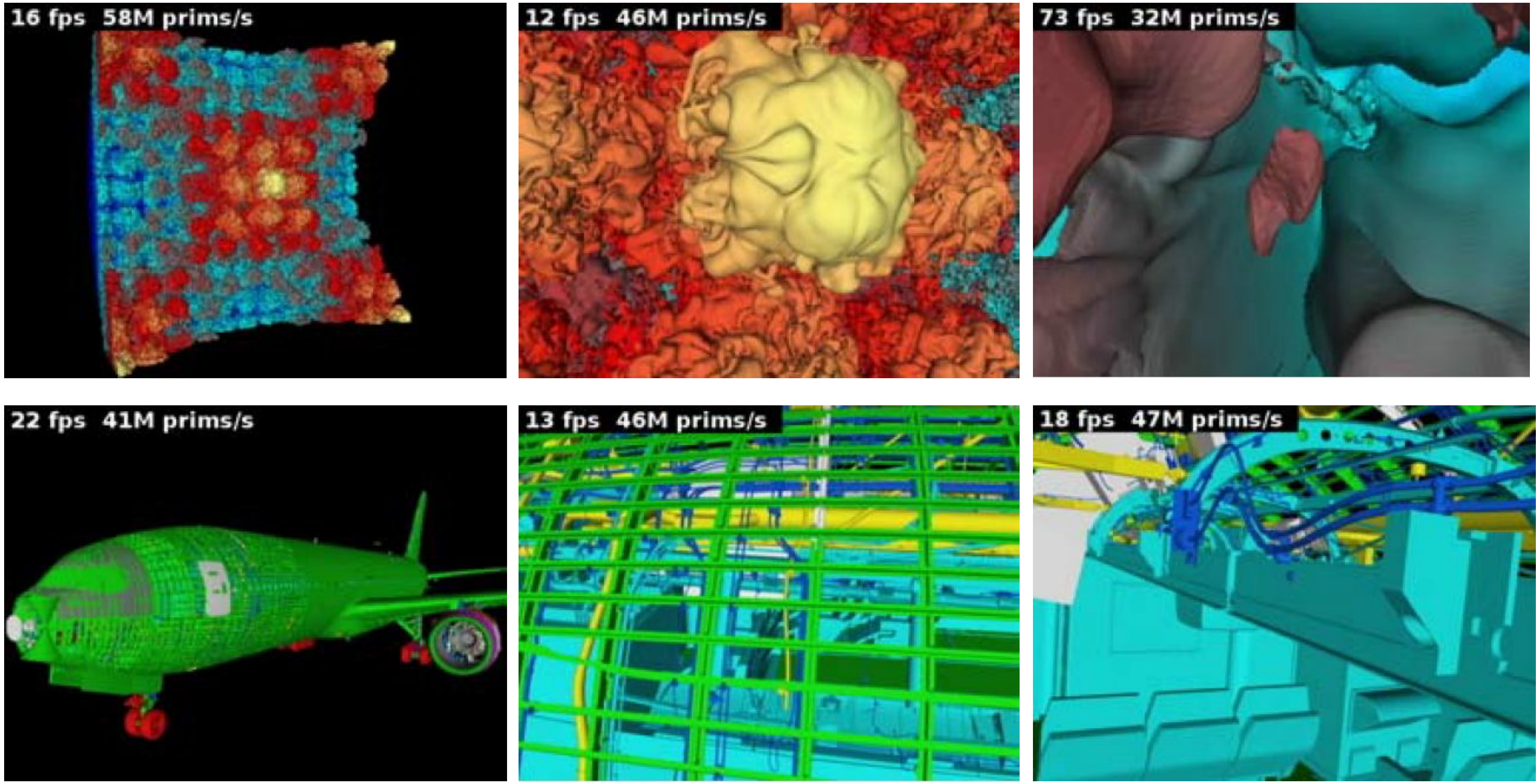


Edge & Point image

Final image

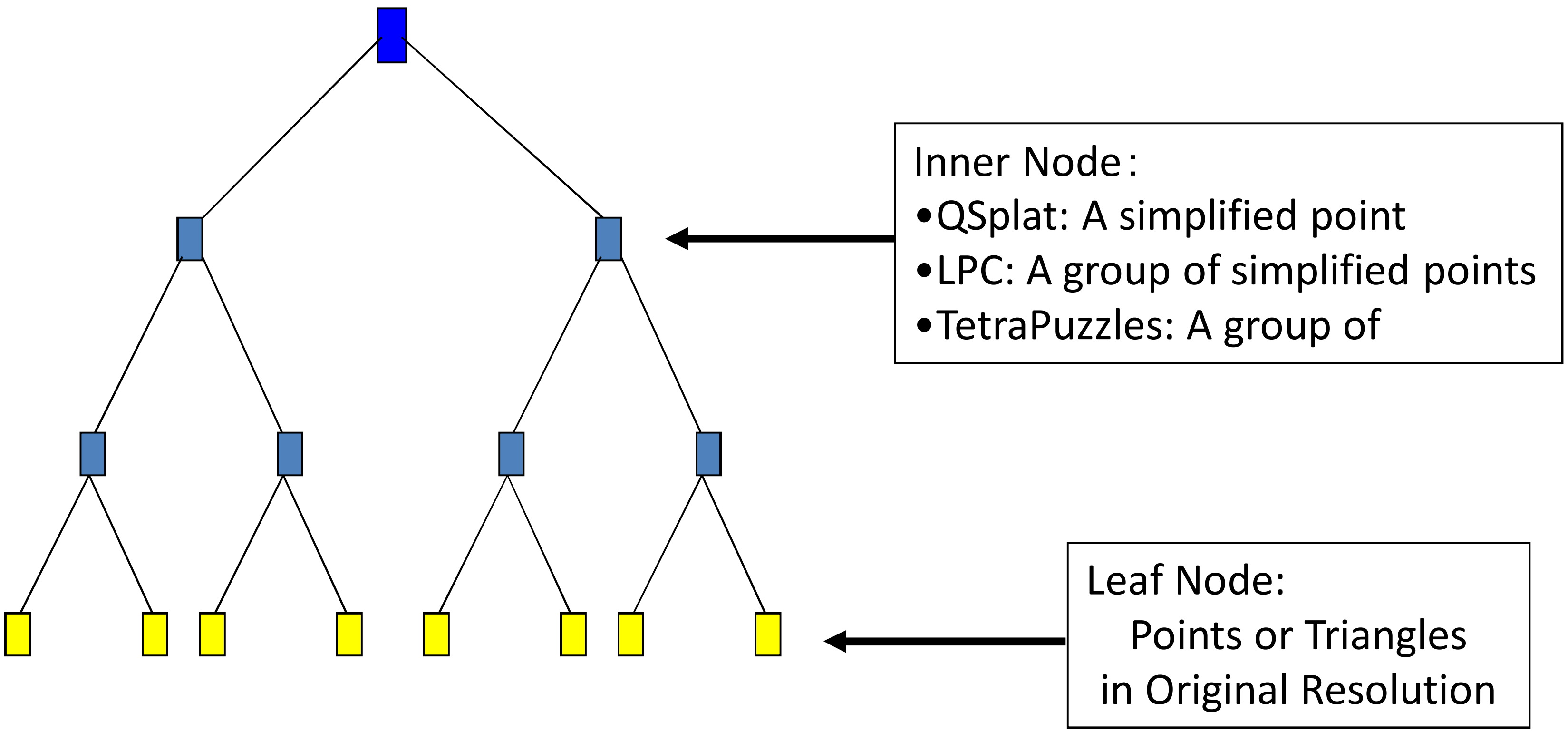
Far Voxels

Enrico Gobbetti, Fabio Marton
SIGGRAPH2005



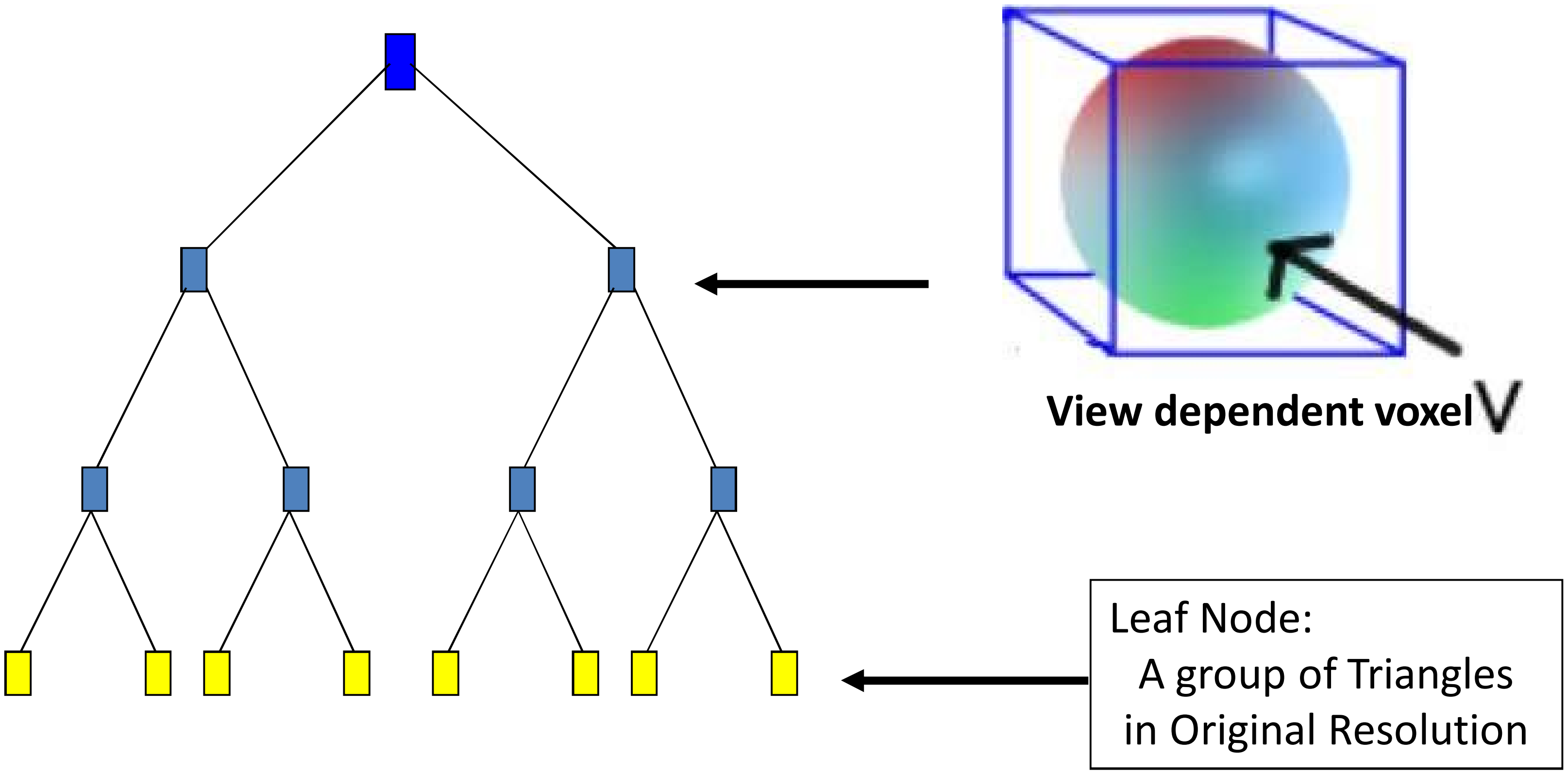
Tree structure

- Recursive Split
- Make inner nodes in bottom up order



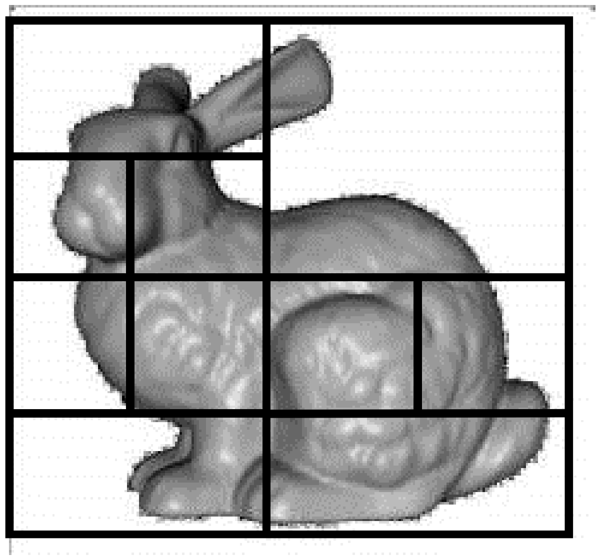
Tree structure of Far Voxels

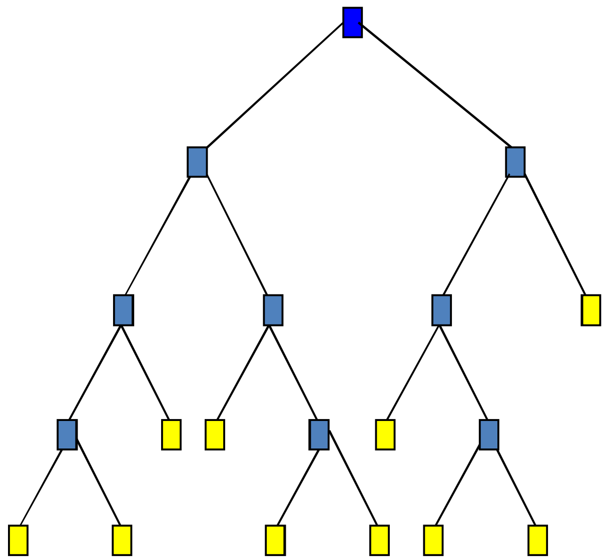
- View Dependent Voxels for inner nodes



Construction of tree structure

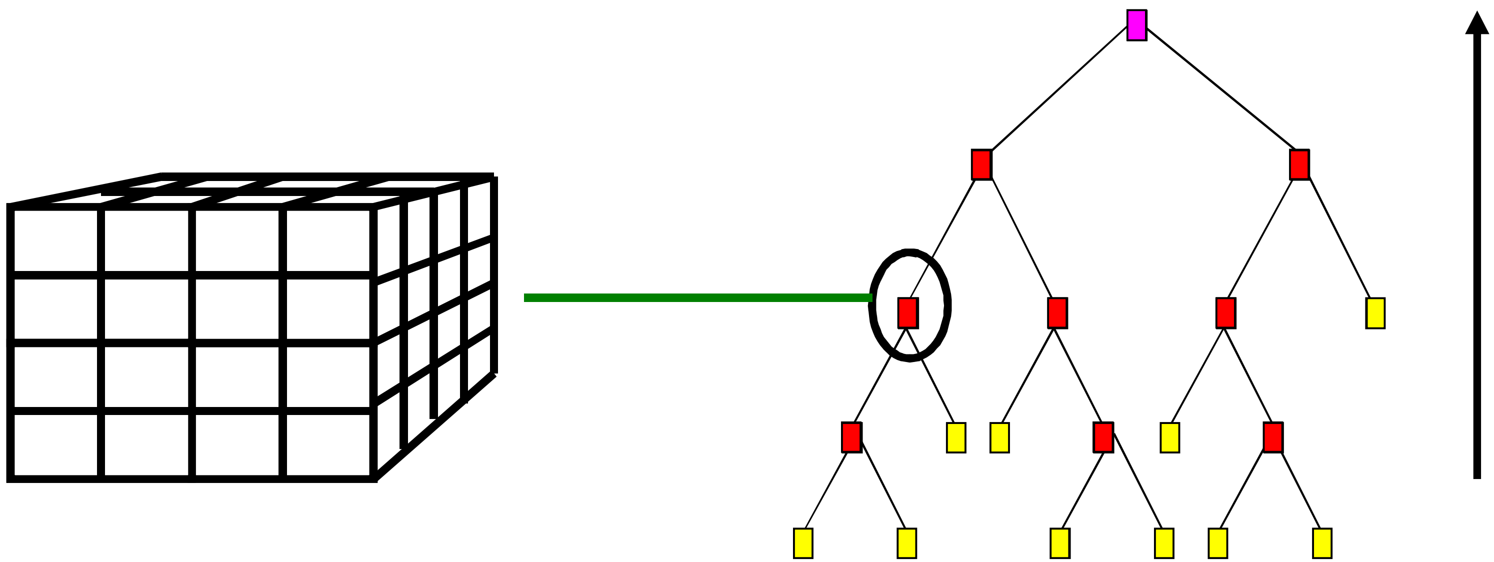
- Recursive split of bounding boxes along the longest axis





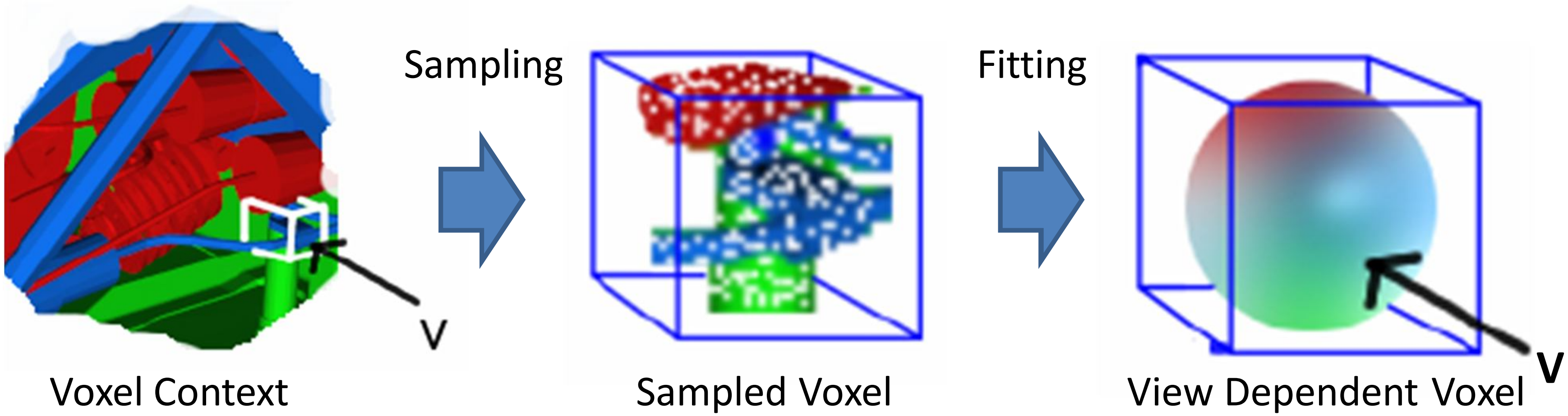
Define inner nodes

- Bottom-up order construction
- Assign a voxel grid from the bounding box to each inner node



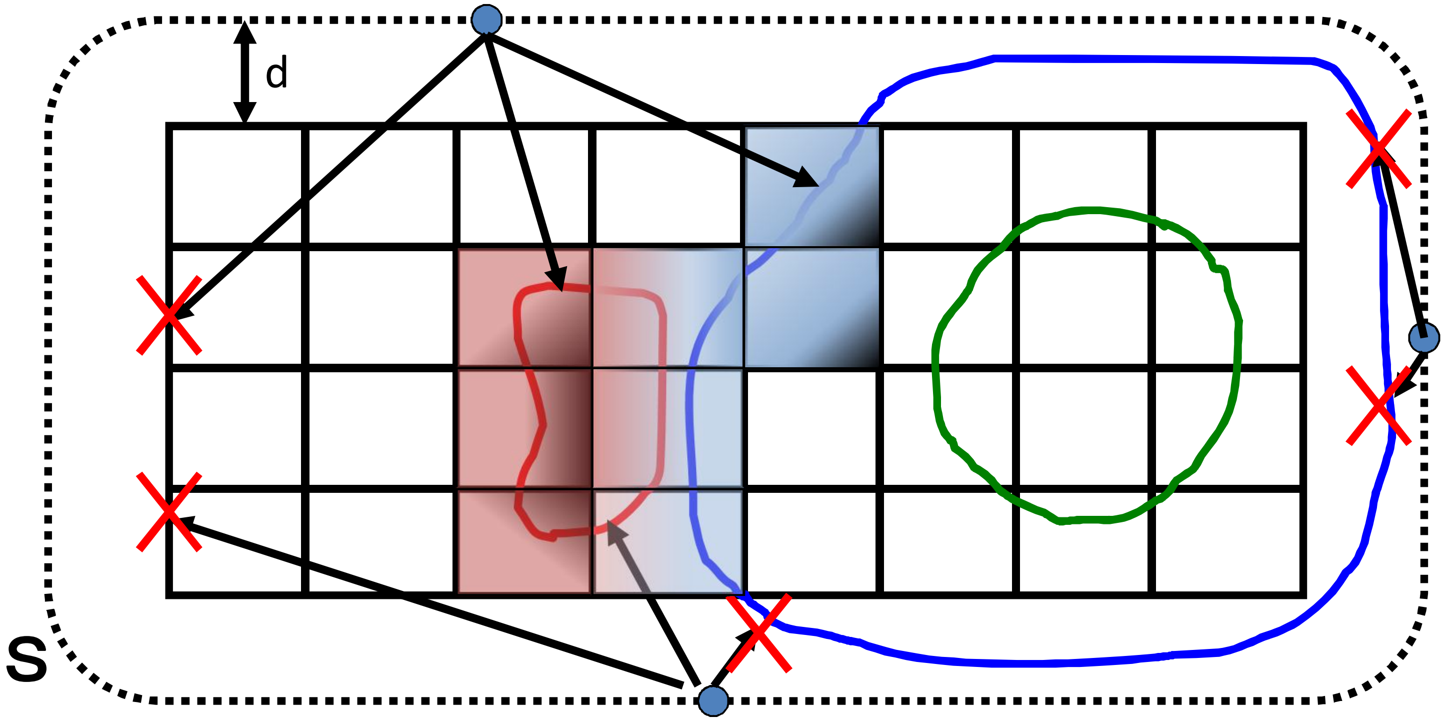
Sampling

- Define "View-Dependent Voxel" for each grid

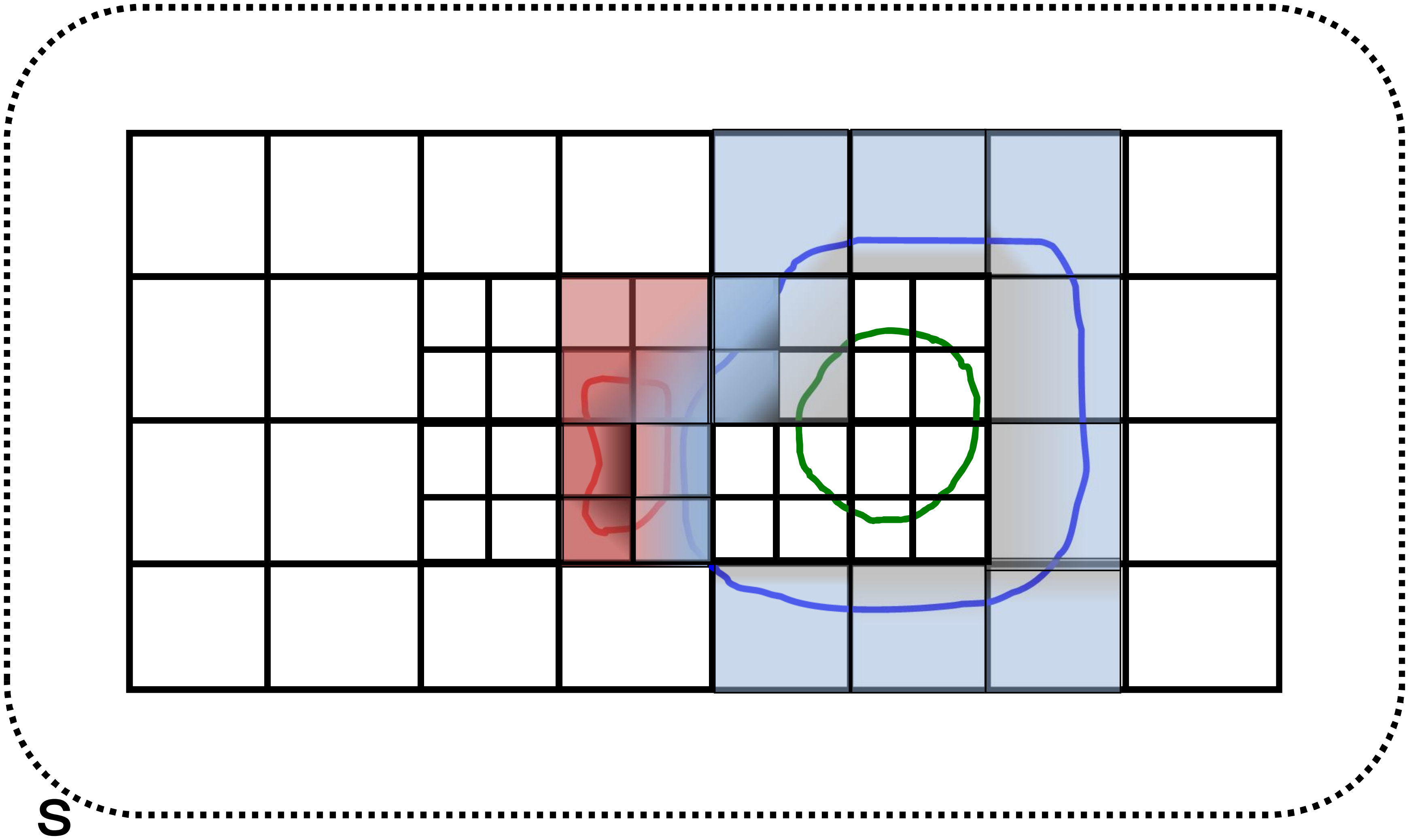


C(v) = (r, g, b)

Sampling by ray casting

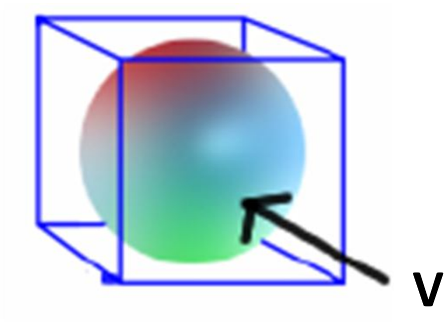


Recursive sampling



Simpler representation

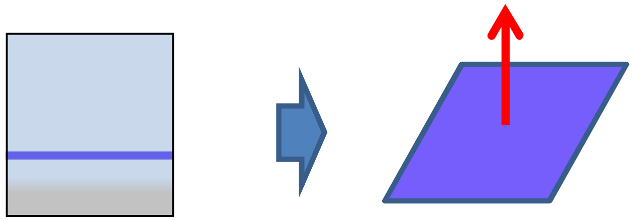
- The representation of View-Dependent Voxel is still complex…



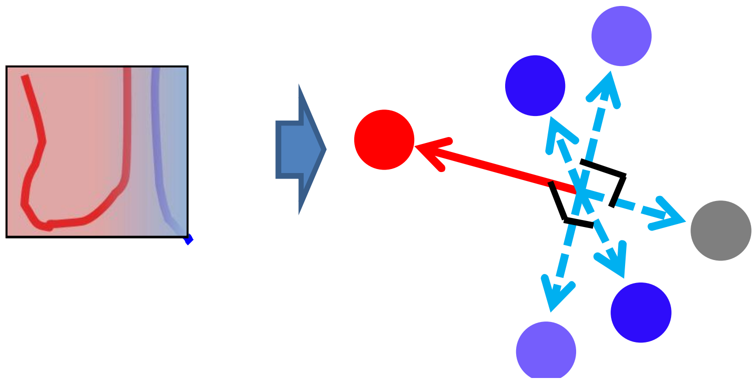
- Classify VDVs to simple Parametric Shaders
  - K1: Flat Shader
  - K2: Smooth Shader

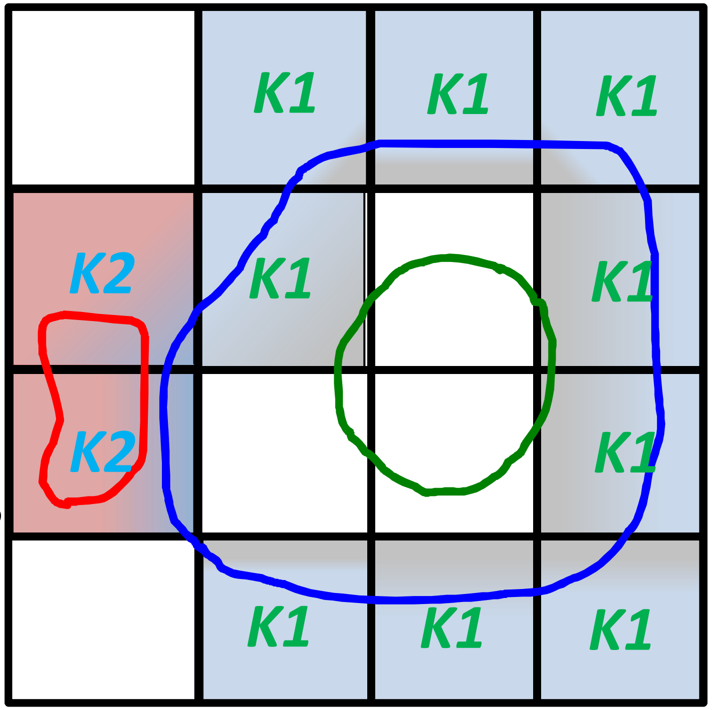
Parametric shaders

- K1: Flat Shader
  – 1 Normal & 2 Colors

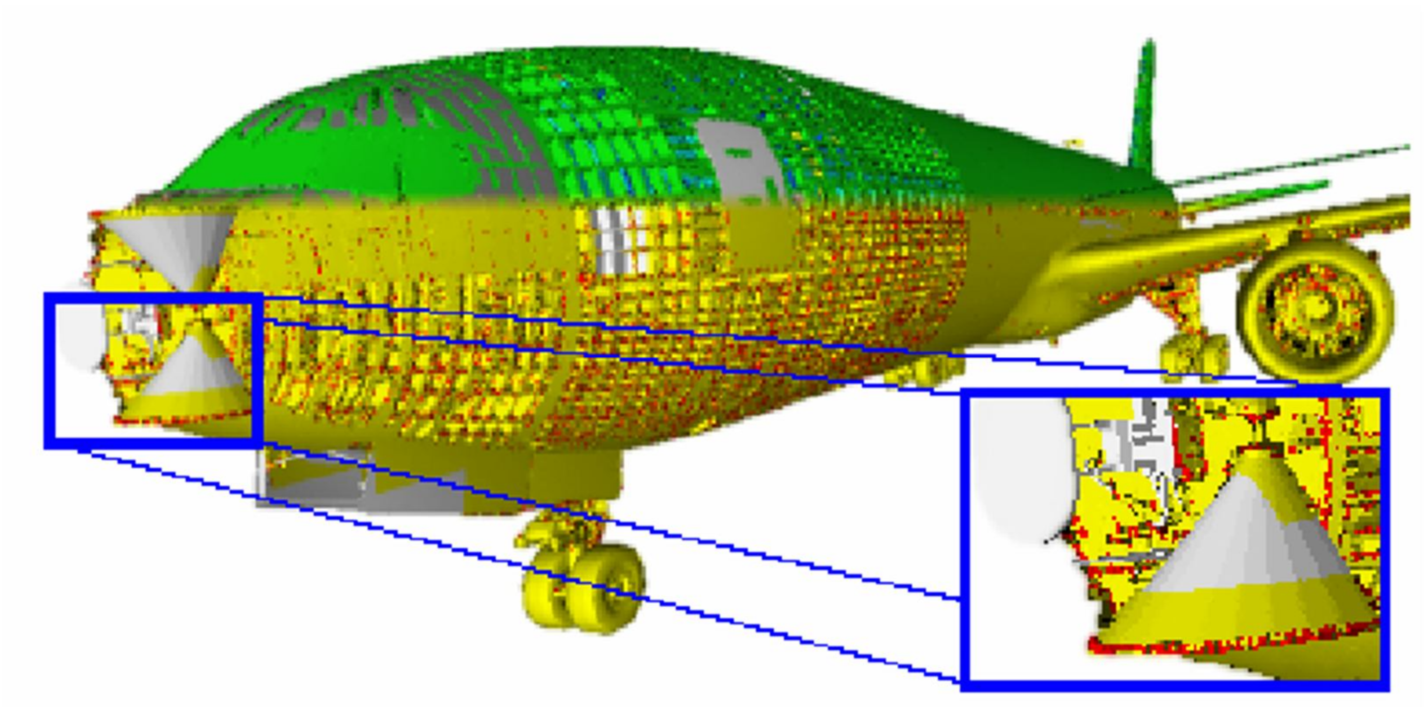


- K2: Smooth Shader
  – 1 Primary Normal & 6 Colors



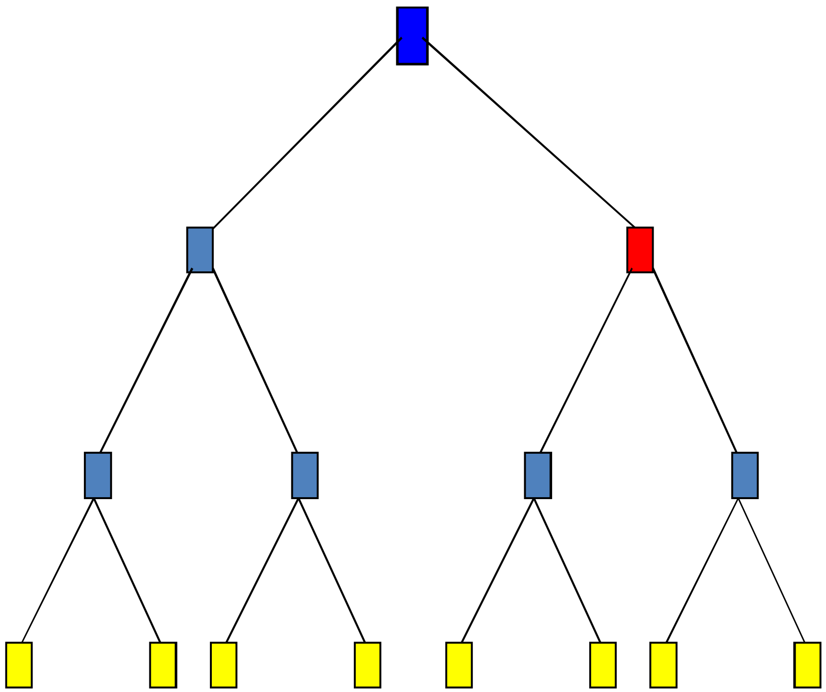


Parametric shaders



Yellow : K1 (Flat Shader)
Red : K2 (Smooth Shader)
White : Leaf (Original Triangles)

Rendering process



if (node not visible)

Skip this subtree

else if (leaf node)

Draw triangles

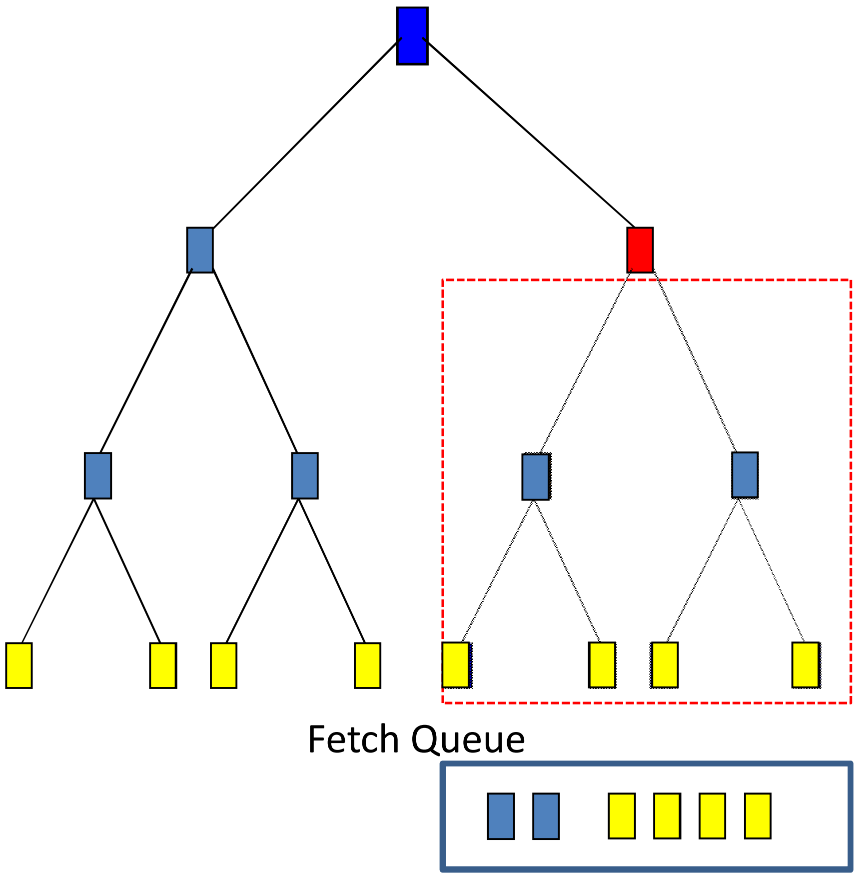
else if (size on screen < threshold)

Draw VDVs or triangles

else

Traverse children

Asynchronous I/O & rendering



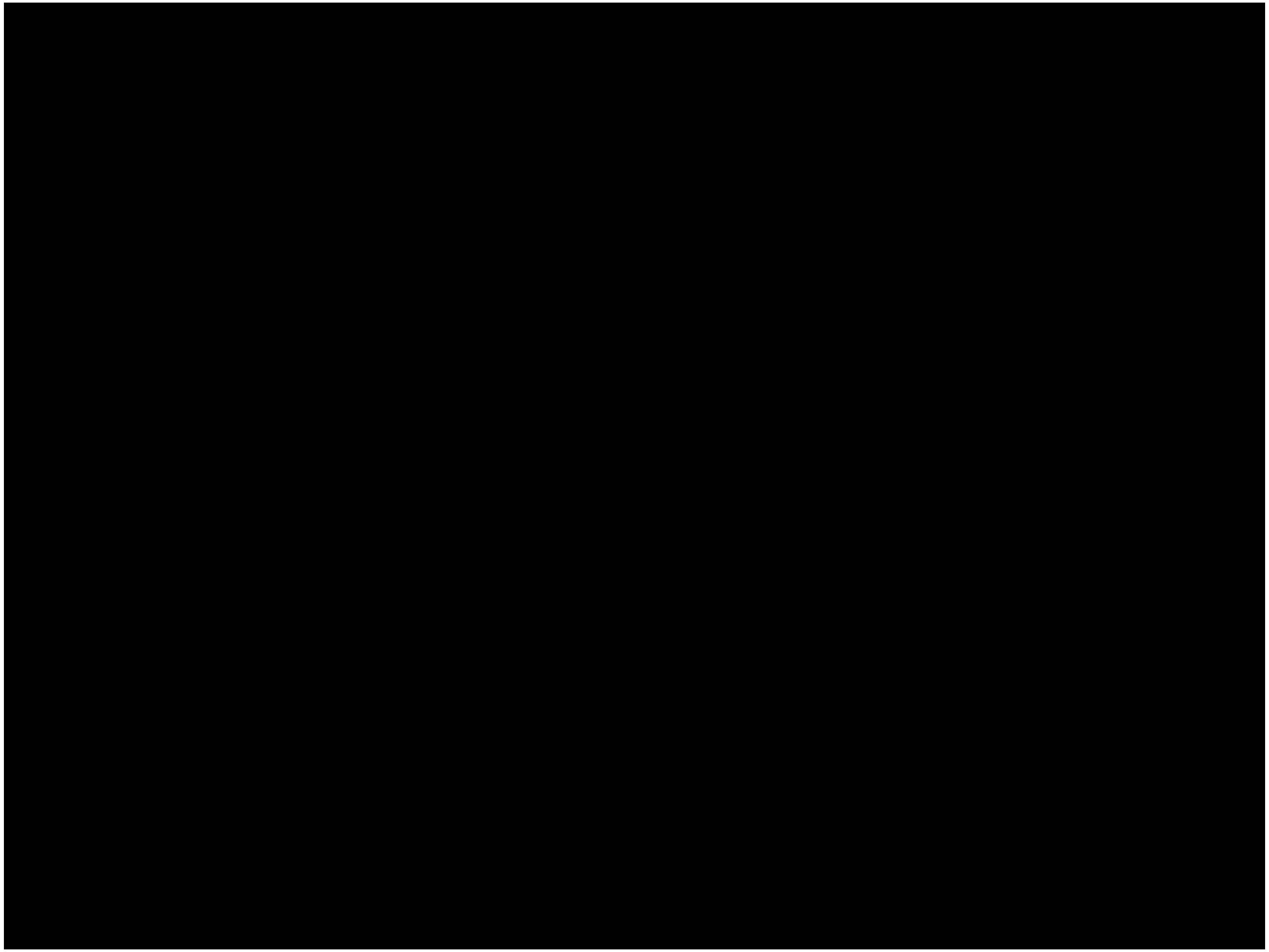
Rendering Process

if (node not visible)
    Skip this subtree
else if (leaf node)
    Draw triangles
else if (size on screen < threshold)
    Draw VDVs or triangles
else if (children are not loaded on memory)
    Draw VDVs or triangles
    Register children to Fetch Queue
else
    Traverse children

I/O Process

while (Nodes exist in Fetch Queue)
    Load Nodes on memory

Demonstration movie



Results

Model	Input Data		Output Data		Frames/sec	
	Triangle	Data Size	Construction time	Data Size	Min	Avg
St. Mathew	372M	14.5GB	14592 sec	10.6GB	9	45
Boeing 777	350M	13.7GB	16461 sec	14.9GB	8	44
Isosurface	472M	18.4GB	23751 sec	16.1GB	7	34

Summary of Far Voxels

- Using Triangles and View-Dependent Voxels
- Simple sampling and classifications for VDV
  - Recursive sampling by ray casting
  - Classification to Flat & Smooth shaders
- Real-time rendering
  - Simple voxel shading
  - Asynchronous I/O processing

Summary

- Data visualization methods for huge 3D data
  - Triangle(Polygon) based methods
    - Mesh Simplification
    - Progressive Meshes
    - Adaptive TetraPuzzles
  - Point & Another Primitive based Rendering
    - QSplat
    - Far Voxels

Papers

- “Surface Simplification Using Quadric Error Metrics”, Michael Garland, Paul S. Heckbert, SIGGRAPH’97
- “Progressive Meshes”, Hugues Hoppe, SIGGRAPH’96
- “Adaptive TetraPuzzles – Efficient Out-of-Core Construction and Visualization of Gigantic Polygonal Models”, Paolo Cignoni et al., SIGGRAPH2004
- “QSplat: A Multiresolution Point Rendering System for Large Meshes”, Szymon Rusinkiewicz, Marc Levoy, SIGGRAPH2000
- “Far Voxels – A Multiresolution Framework for Interactive Rendering of Huge Complex 3D Models on Commodity Graphics Platforms”, Enrico Gobbetti, Fabio Marton, SIGGRAPH2005

Additional Papers

- "Layered Point Clouds", Enrico Gobbetti, Fabio Marton, Eurographics Symposium on Point-Based Graphics 2004
- "POP: A Hybrid Point and Polygon Rendering System for Large Data", Baoquan Chen, Minh Xuan Nguyen, Conference on Visualization 2001
- "Combining Edges and Points for Interactive High-Quality Rendering", Kavita Bala et al., SIGGRAPH2003

Announcement

Before

12/17	Data Processing(1)	(Prof. Oishi)
1/7	Data Processing(2)	(Prof. Oishi)
1/14	Patch-based Object Recognition(1)	(Prof. Kagesawa)
1/21	Patch-based Object Recognition(2)	(Prof. Kagesawa)

After

12/17	Patch-based Object Recognition(1)	(Prof. Kagesawa)
1/7	Data Processing(1)	(Prof. Oishi)
1/14	Data Processing(2)	(Prof. Oishi)
1/21	Patch-based Object Recognition(2)	(Prof. Kagesawa)