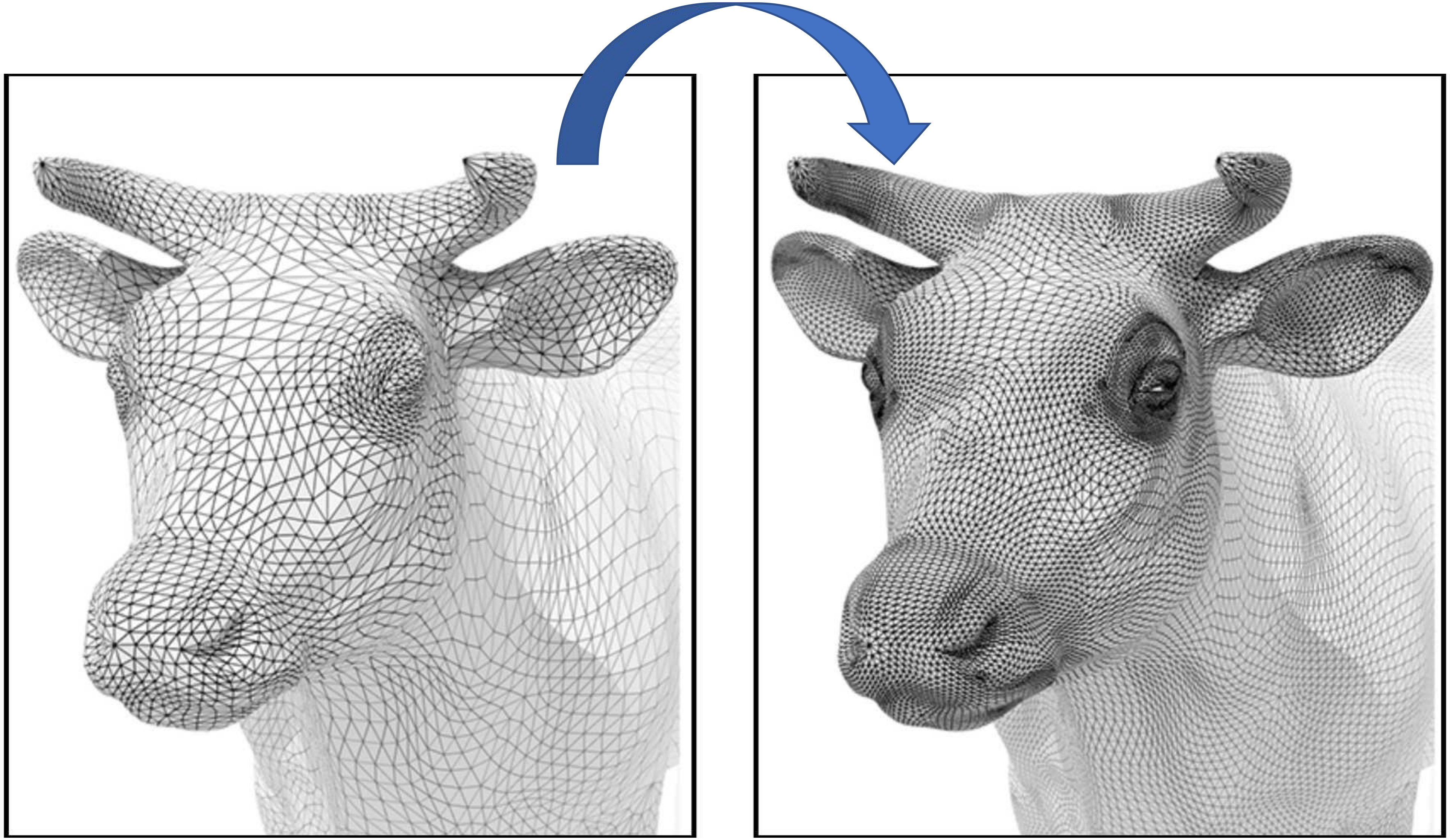
Geometry

Mesh Subdivision

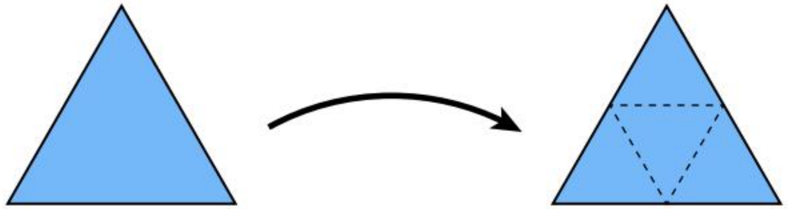
Mesh Subdivision

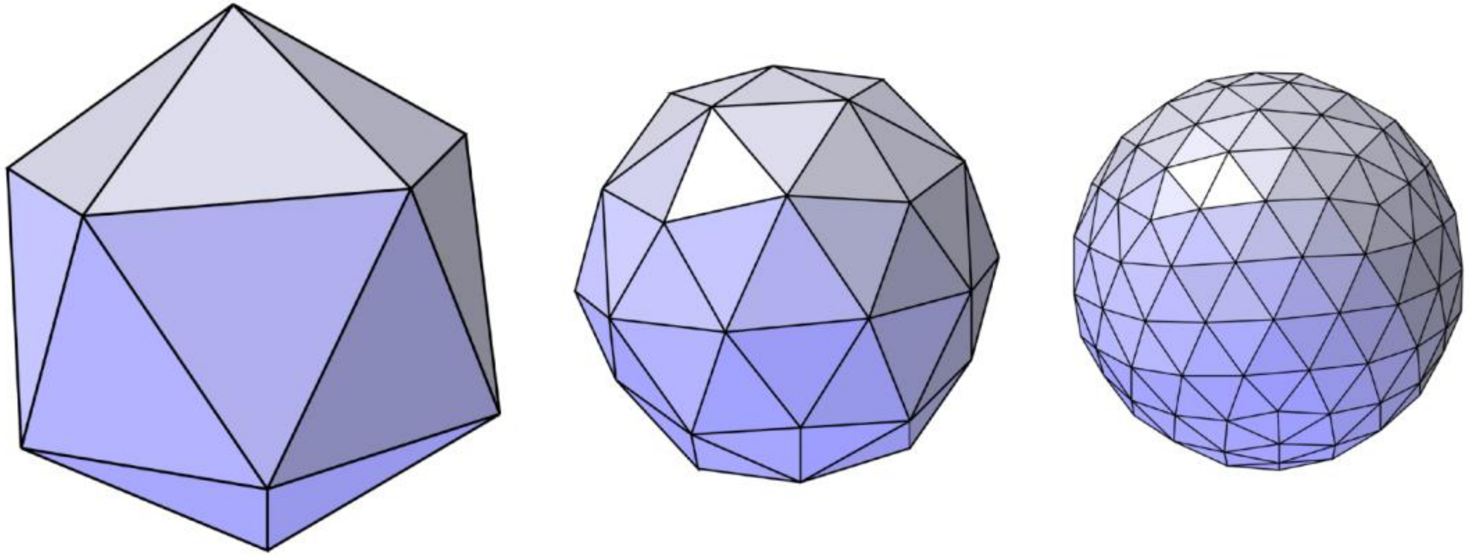


Loop Subdivision

Triangular Mesh Subdivision

- One-to-four split





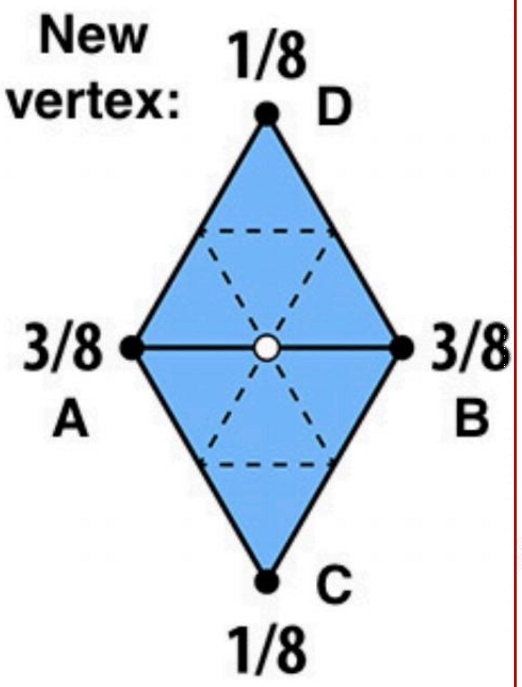
Simon Fuhrman

Loop Subdivision

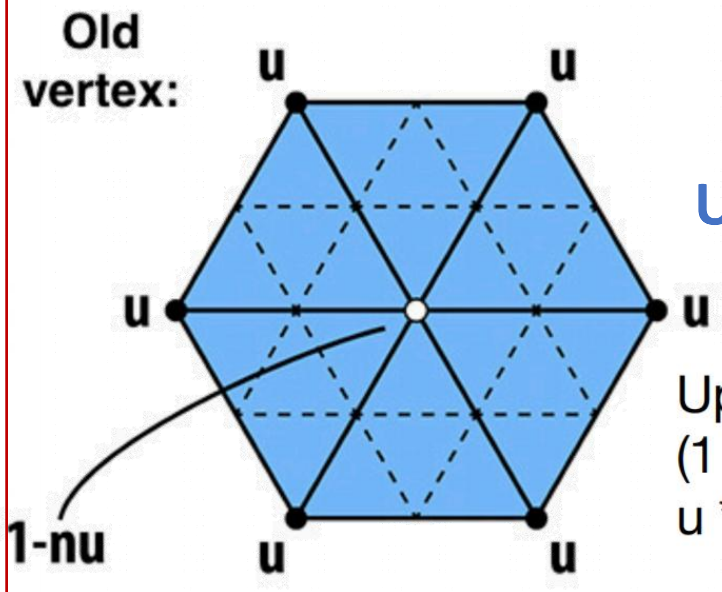
Triangular Mesh Subdivision

- One-to-four split
- Update vertex position

Update new vertex



Update to:
3/8 * (A + B) + 1/8 * (C + D)



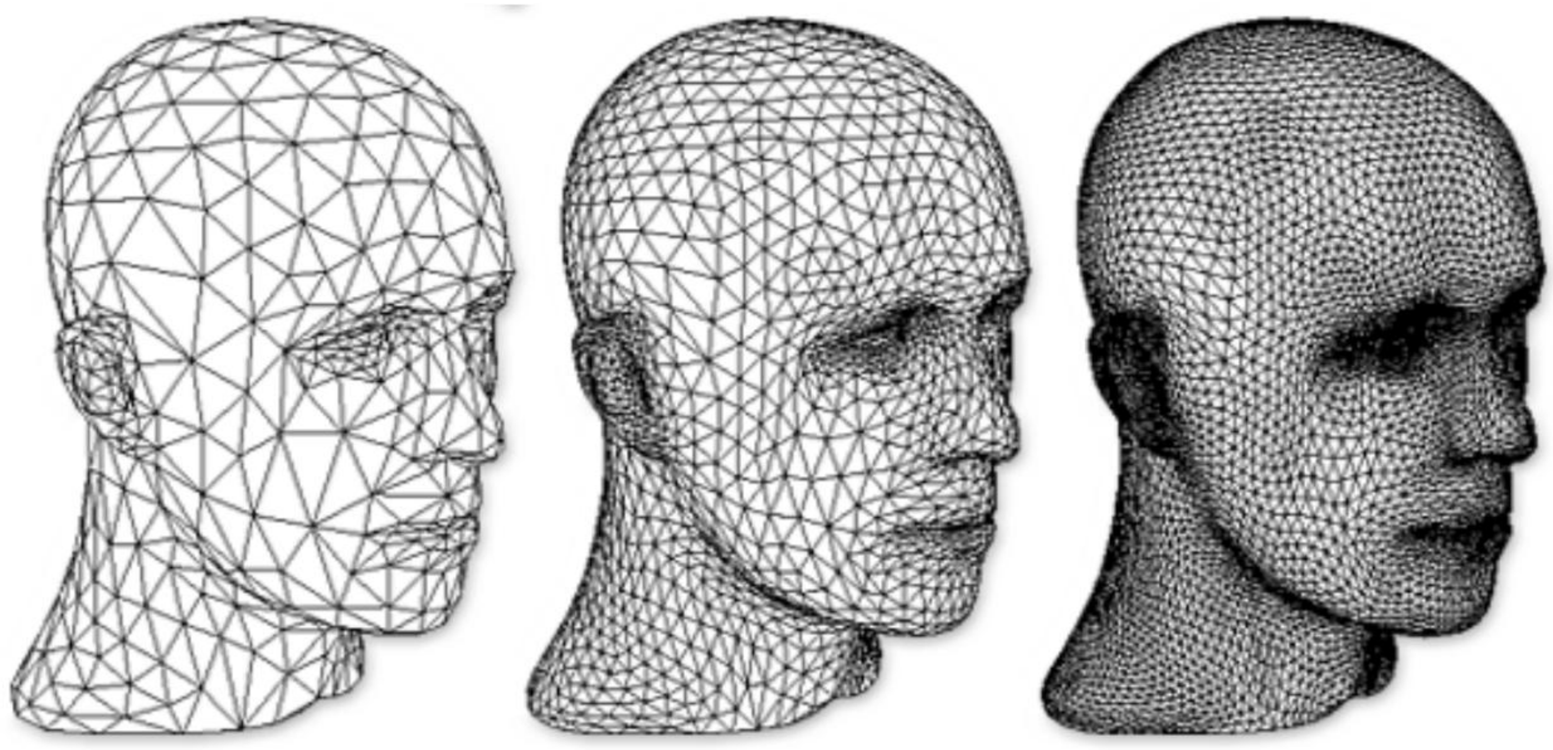
Update old vertex

Update to:
(1 - n*u) * original_position +
u * neighbor_position_sum

Loop Subdivision

Triangular Mesh Subdivision

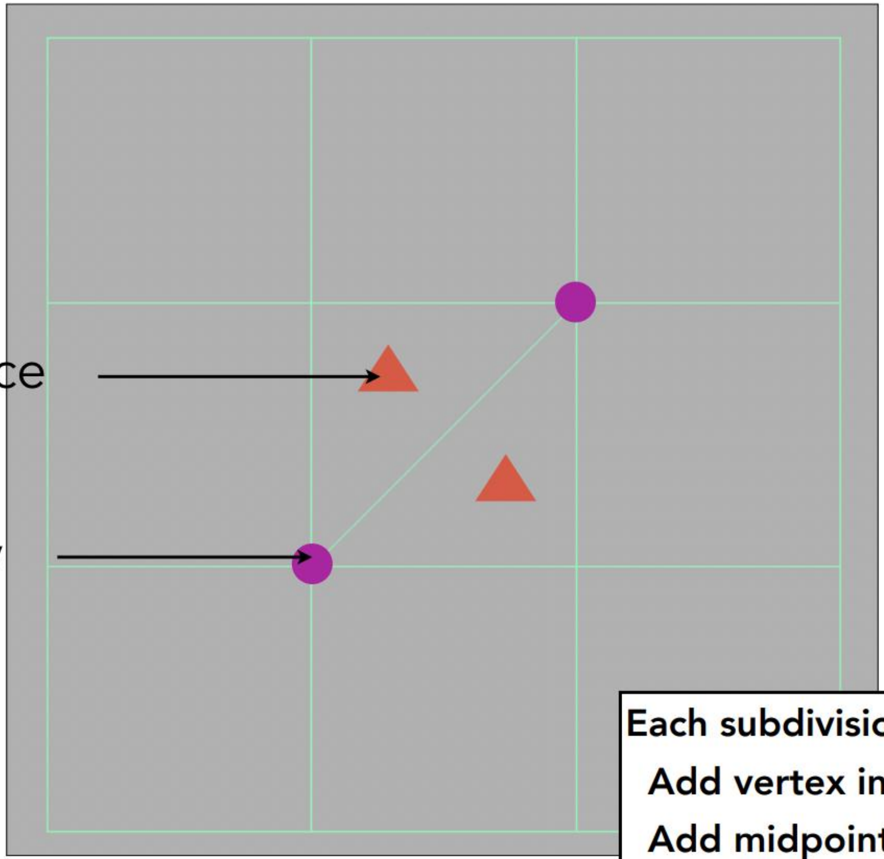
- One-to-four split
- Update vertex position



Catmull-Clark Subdivision (General Mesh)

Non-quad face

Extraordinary vertex (奇异点) (degree != 4)

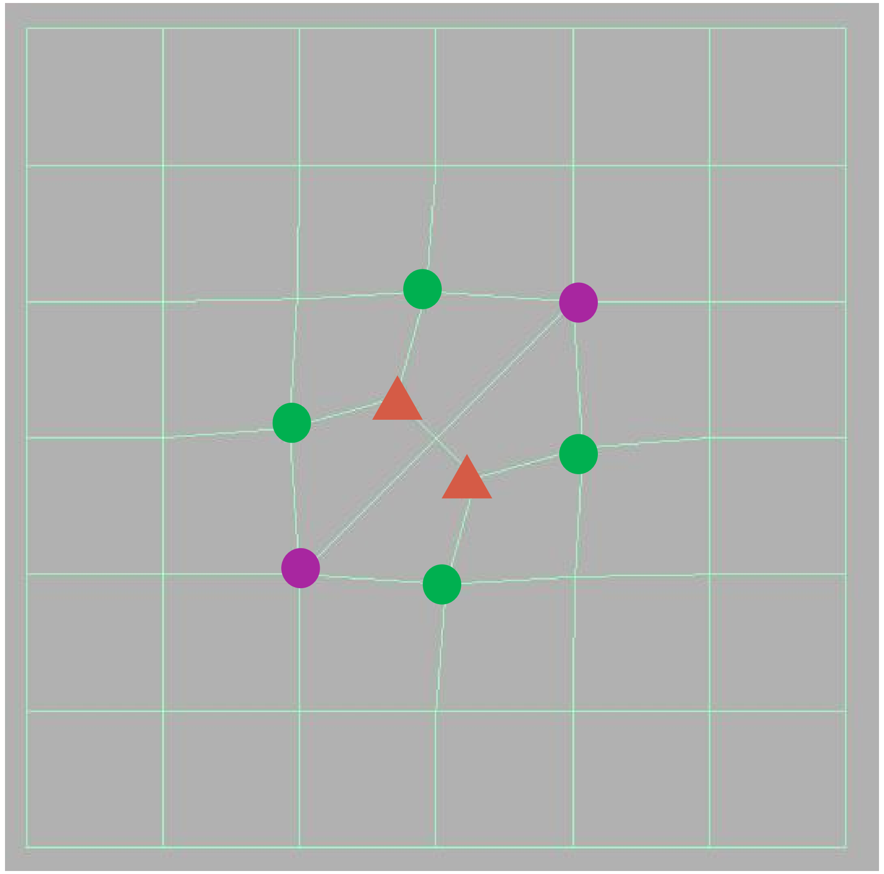


Each subdivision step:
- Add vertex in each face
- Add midpoint on each edge
- Connect all new vertices

New extraodinary vertices

Old extraodinary vertices

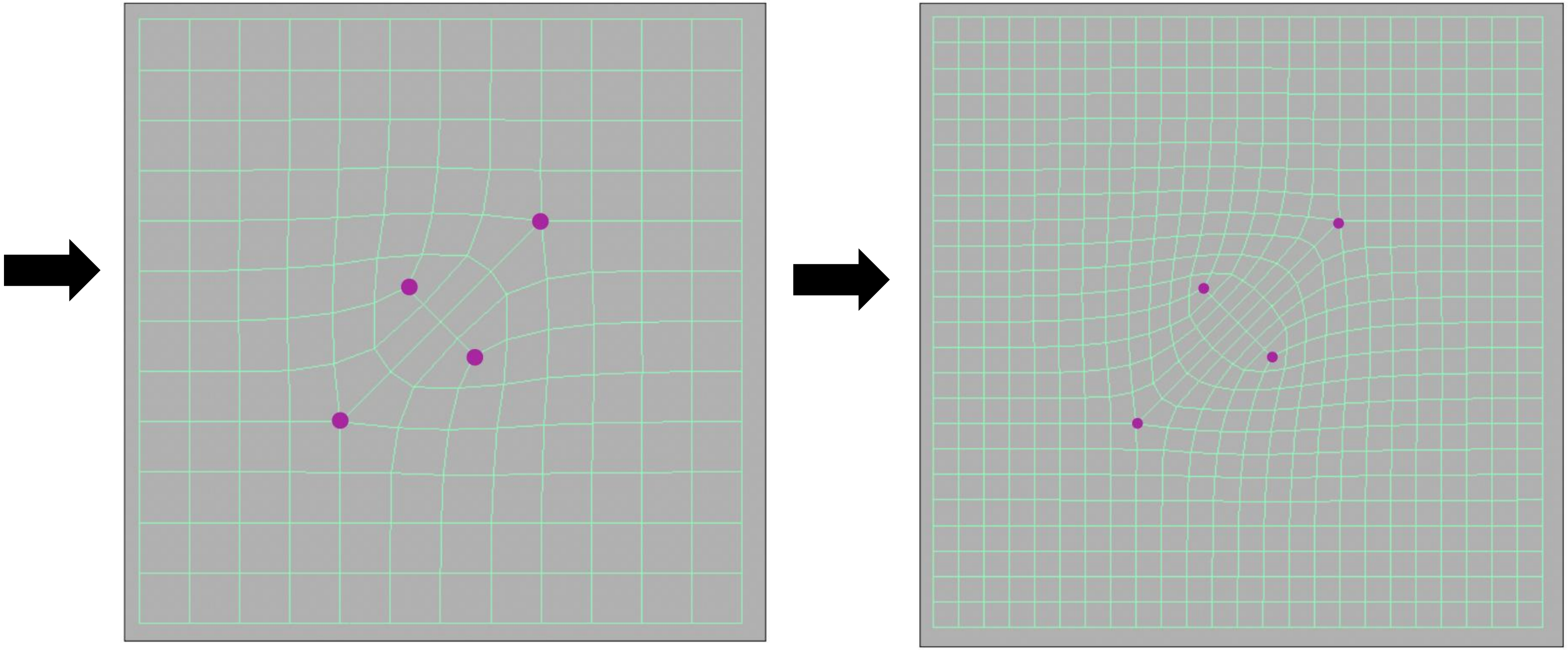
Regular vertices (degree=4)



Catmull-Clark Subdivision (General Mesh)

No more extraordinary vertices!

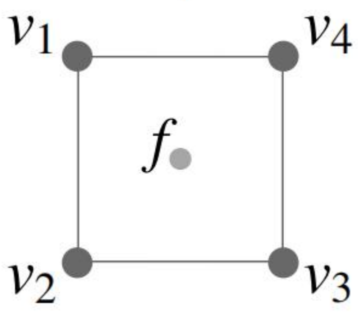
All quad faces!



Catmull-Clark Subdivision (General Mesh)

Three types of vertices

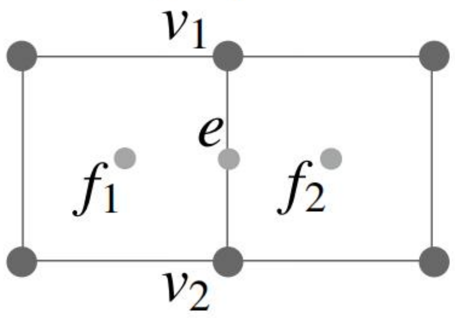
Face point

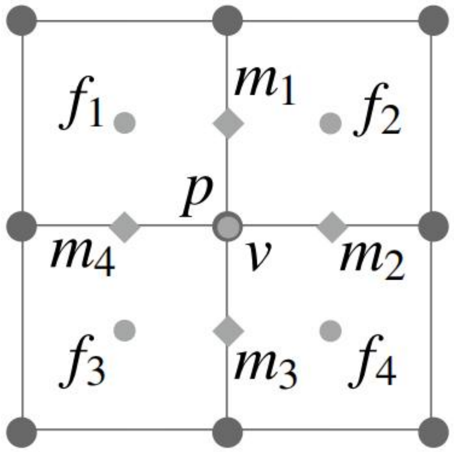


$$f = \frac{v_1 + v_2 + v_3 + v_4}{4}$$

$$e = \frac{v_1 + v_2 + f_1 + f_2}{4}$$

Edge point





Vertex point

$$v = \frac{f_1 + f_2 + f_3 + f_4 + 2(m_1 + m_2 + m_3 + m_4) + 4p}{16}$$

m midpoint of edge

p old "vertex point"

Convergence: Overall Shape and Creases

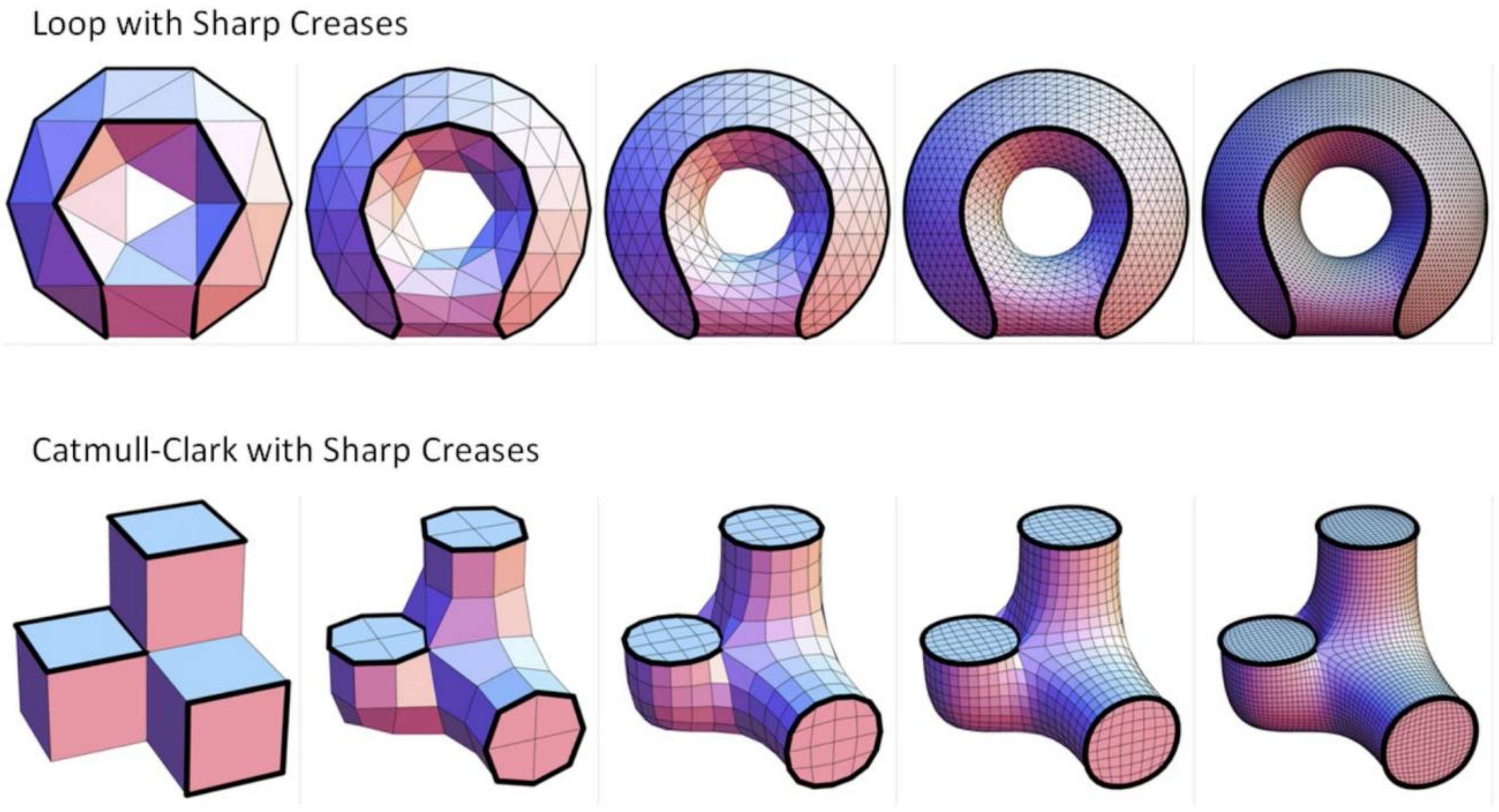
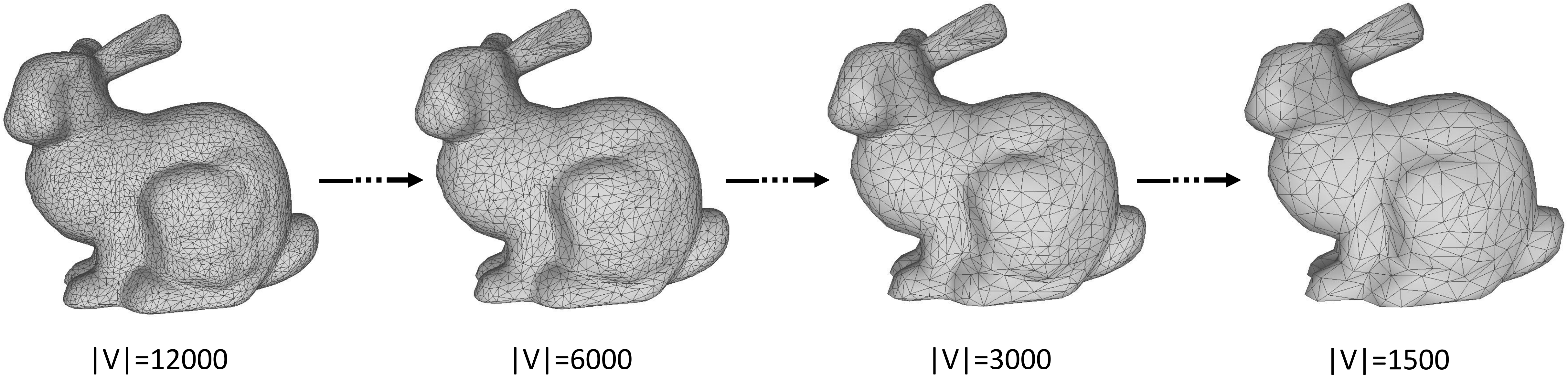


Figure from: Hakenberg et al. Volume Enclosed by Subdivision Surfaces with Sharp Creases

Mesh Simplification

Mesh Simplification

- Iteratively remove one vertex/edge per step



Mesh Simplification

- Initialization
- Repeat:
- select a vertex/edge to remove
- single-simplification step

When to stop?

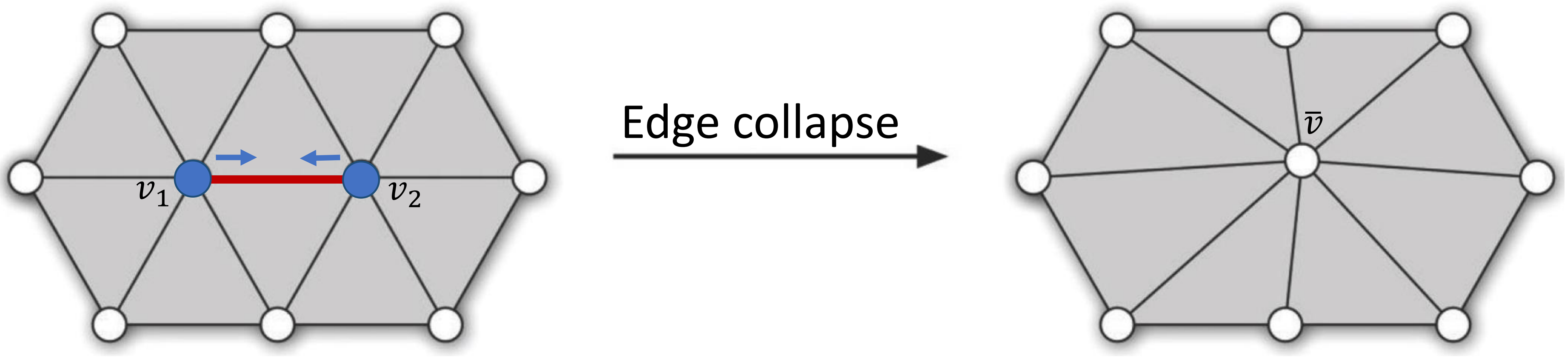
How to select?

How to simplify?

QEM(Quadric Error Metrics) Simplification

How to simplify?

Edge Collapse: one edge contracted to one vertex

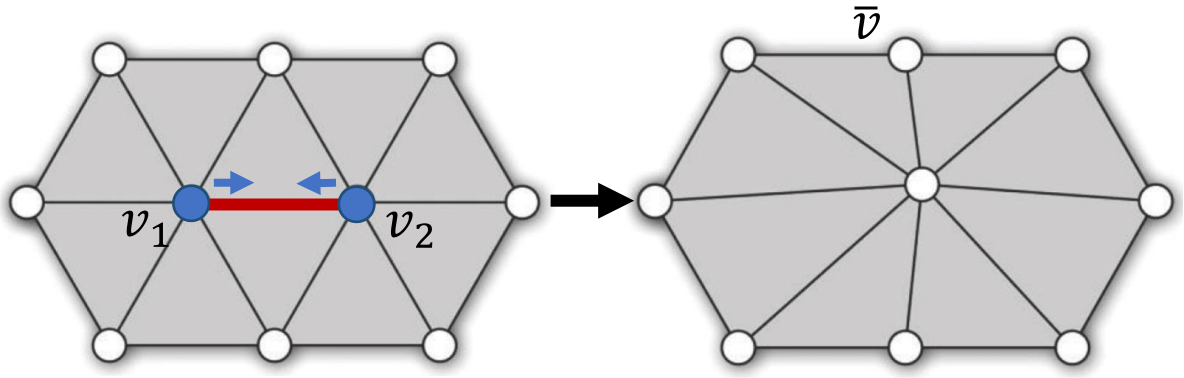


QEM(Quadric Error Metrics) Simplification

How to simplify?

Edge Collapse: one edge contracted to one vertex

Update position of new vertex: Quadric Error Metrics



优化收缩点$\bar{v}$的位置，使之尽可能接近原模型。
优化目标函数：

$$\bar{v} = \underset{v}{\operatorname{argmin}} \sum_{P \in plane(v_1) \cup plane(v_2)} distance(v, P)^2$$

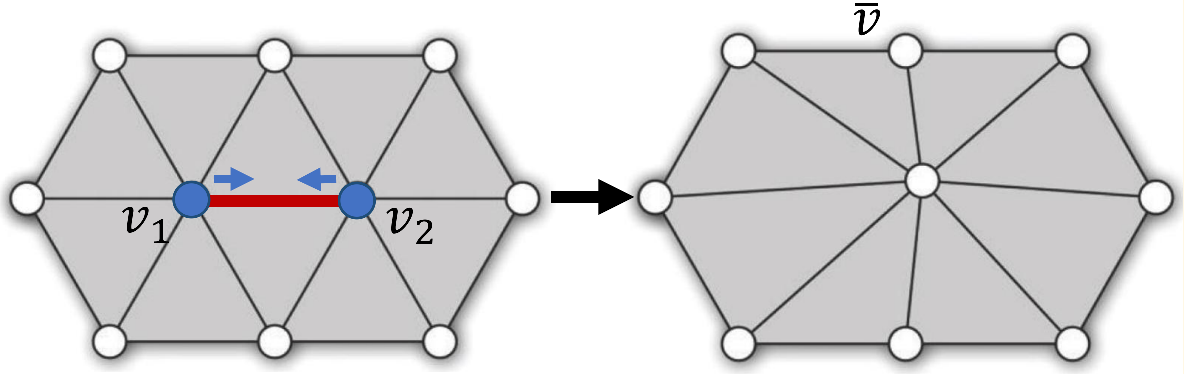
$plane(v)$指顶点v邻接的三角面片集合

QEM(Quadric Error Metrics) Simplification

How to simplify?

dge Collapse: one edge contracted to one vertex

Update position of new vertex: Quadric Error Metrics



优化收缩点$\bar{v}$的位置，使之尽可能接近原模型。
优化目标函数：

$$\bar{v} = \underset{v}{\operatorname{argmin}} \sum_{P \in plane(v_1) \cup plane(v_2)} distance(v, P)^2$$

$plane(v)$指顶点v邻接的三角面片集合

令平面P为$ax + by + cz + d = 0$，其中$a^2 + b^2 + c^2 = 1$。
记$\boldsymbol{p} = [a, b, c, d]^T$， $v = [x, y, z, 1]^T$ ，
则有：

$$distance(v, P)^2 = (v^T p)^2 = v^T p p^T v = v^T K_p v$$

因此，设$plane(v)$为顶点v邻接的三角面片，收缩点的位置可写为：

$$\bar{v} = \underset{v}{\operatorname{argmin}}\, v^T \left(\sum_{P \in plane(v_1) \cup plane(v_2)} K_p \right) v$$

又因：

$$\bar{v} \approx \underset{v}{argmin}\, v^T \left(\sum_{p \in plane(v_1)} K_p + \sum_{p \in plane(v_2)} K_p \right) v$$

所以，最终可写成：

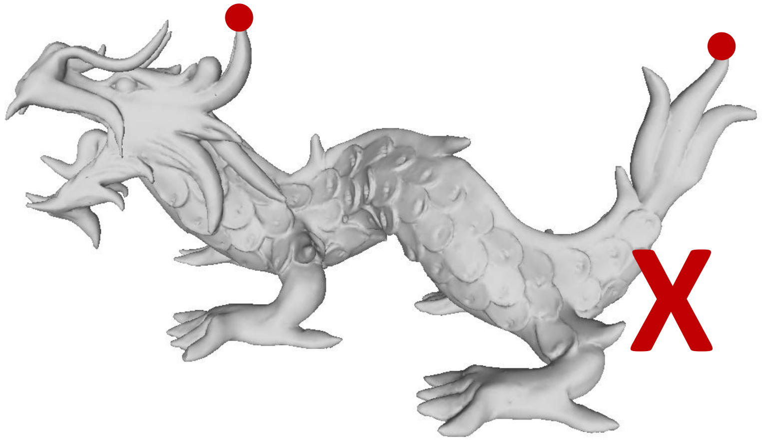
$$\bar{v} = \underset{v}{\operatorname{argmin}}\, v^T \boxed{(Q_1 + Q_2)} v$$

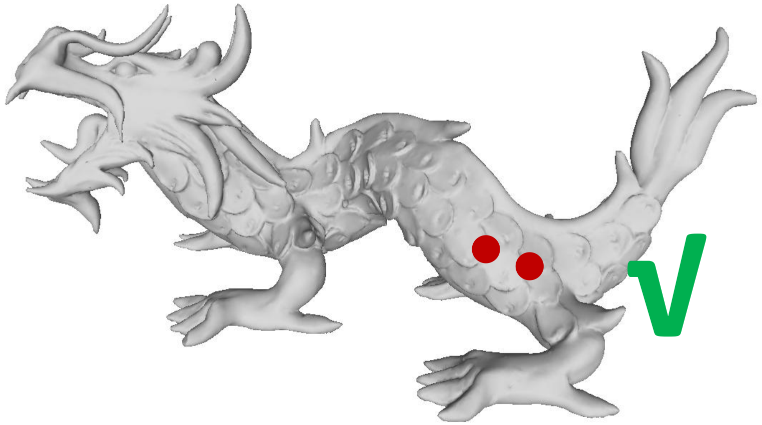
由v_1和v_2的邻接面计算得到

QEM(Quadric Error Metrics) Simplification

How to select?

- Select the connected nearby vertices

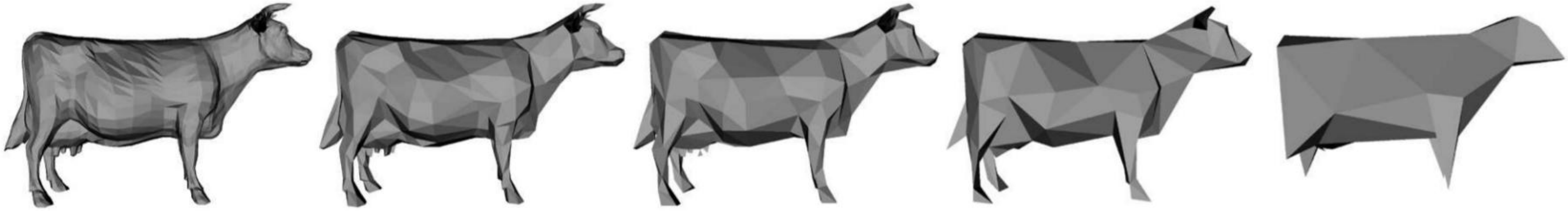




QEM(Quadric Error Metrics) Simplification

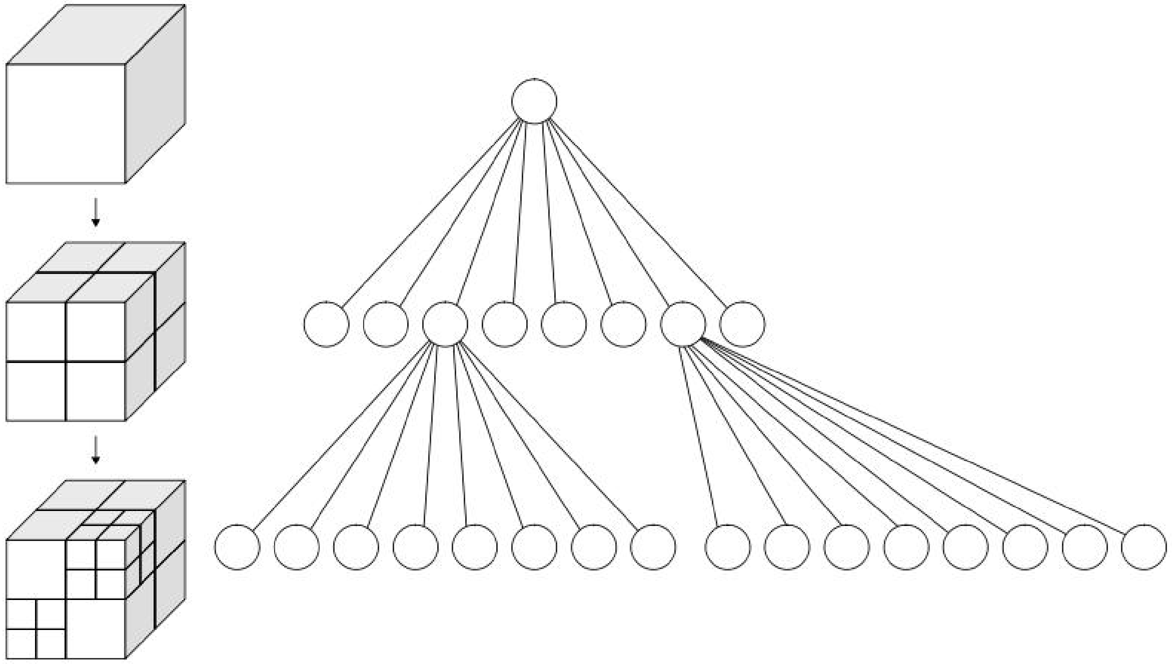
QEM Algorithm Outline

1. Compute the Q_v matrices for all the initial vertices.
2. Select all valid vertex pairs.
3. Compute the optimal $\overline{v}$ for each pair, $\Delta(\overline{v})$ becomes the cost of contracting that pair.
4. Place all the pairs in a heap keyed on cost with the minimum cost pair at the top.
5. Iteratively remove the pair (v_1, v_2) of least cost from the heap, contract this pair, and update the costs of all valid pairs involving the new vertex $\overline{v}$.



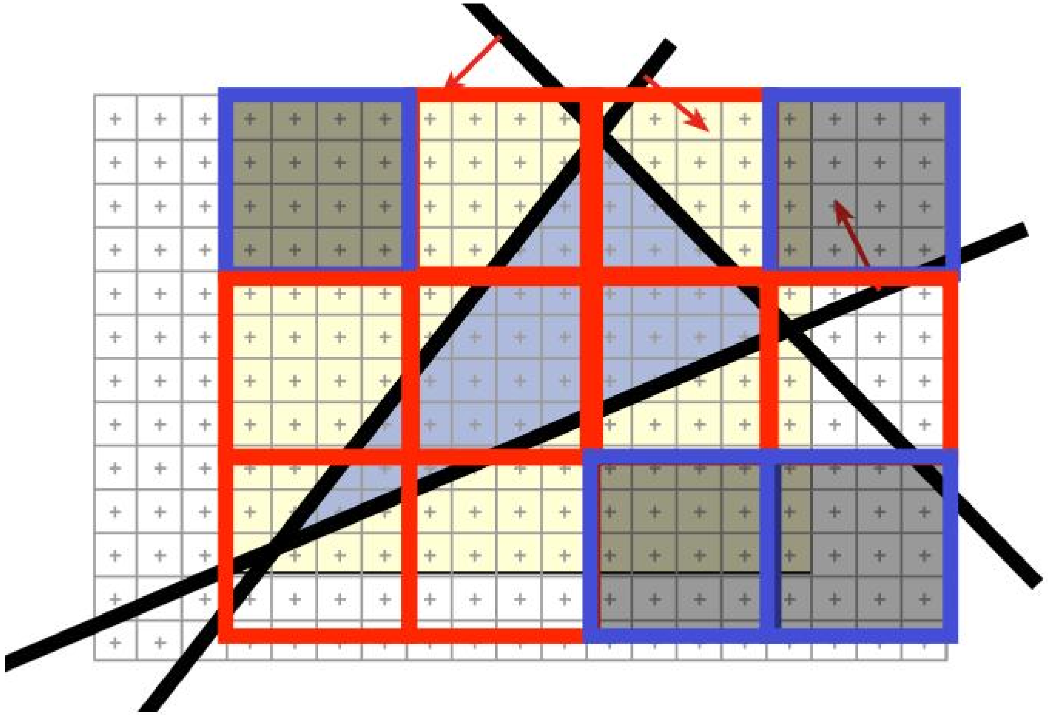
Simplification Results

Spatial Partition



Uniform Partition (Grids)

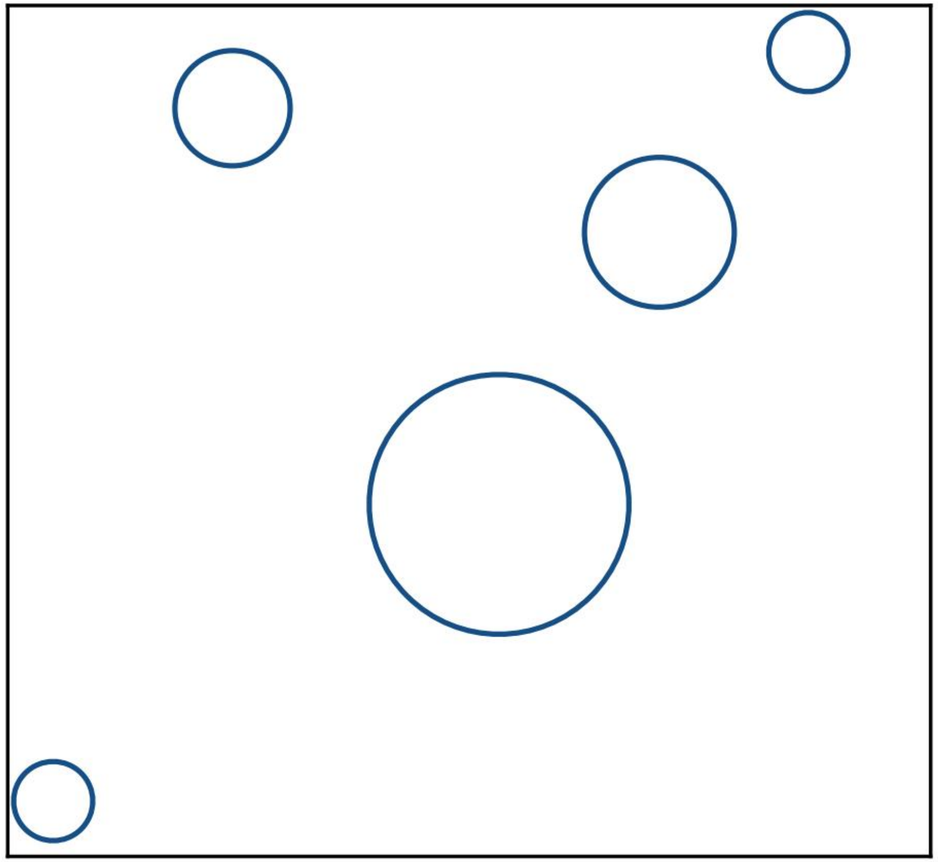
- Recall Rasterization



Test the blocks before
computing the pixels

Uniform Partition (Grids)

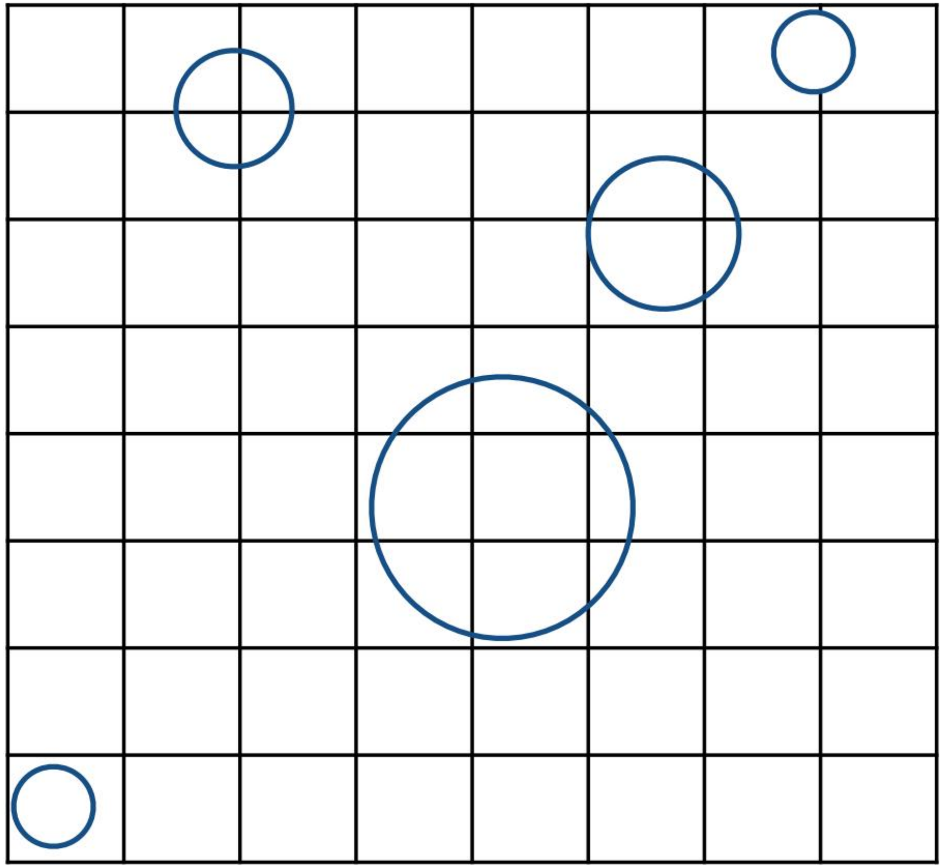
- Grid structure to accelerate the processing (collision, rendering, etc)



1. Find bounding box

Uniform Partition (Grids)

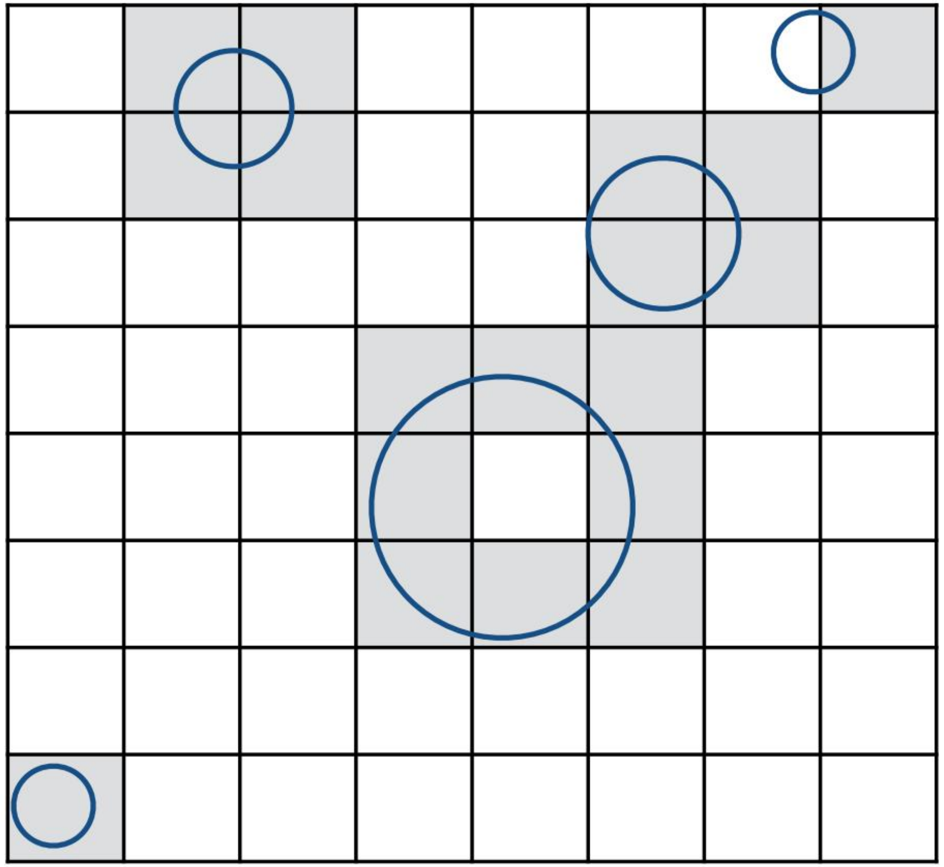
- Grid structure to accelerate the processing (collision, rendering, etc)



1. Find bounding box
2. Create grid

Uniform Partition (Grids)

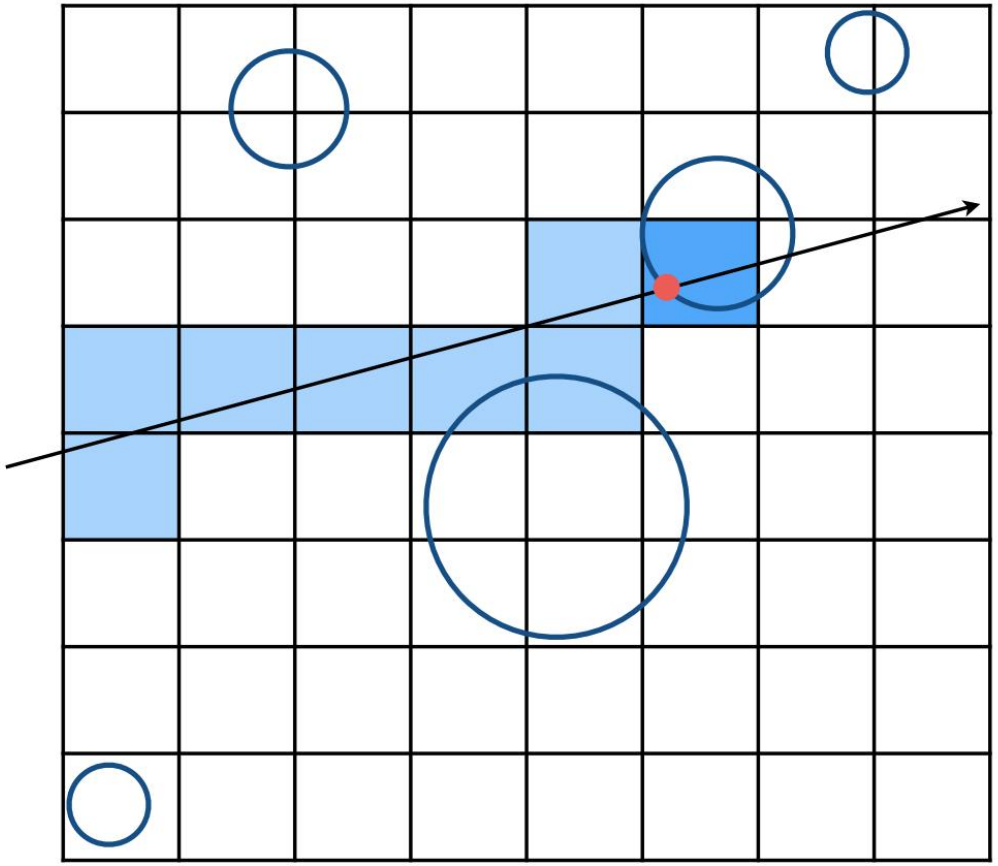
- Grid structure to accelerate the processing (collision, rendering, etc)



1. Find bounding box
2. Create grid
3. Store each object in overlapping cells

Uniform Partition (Grids)

- Grid structure to accelerate the processing (collision, rendering, etc)

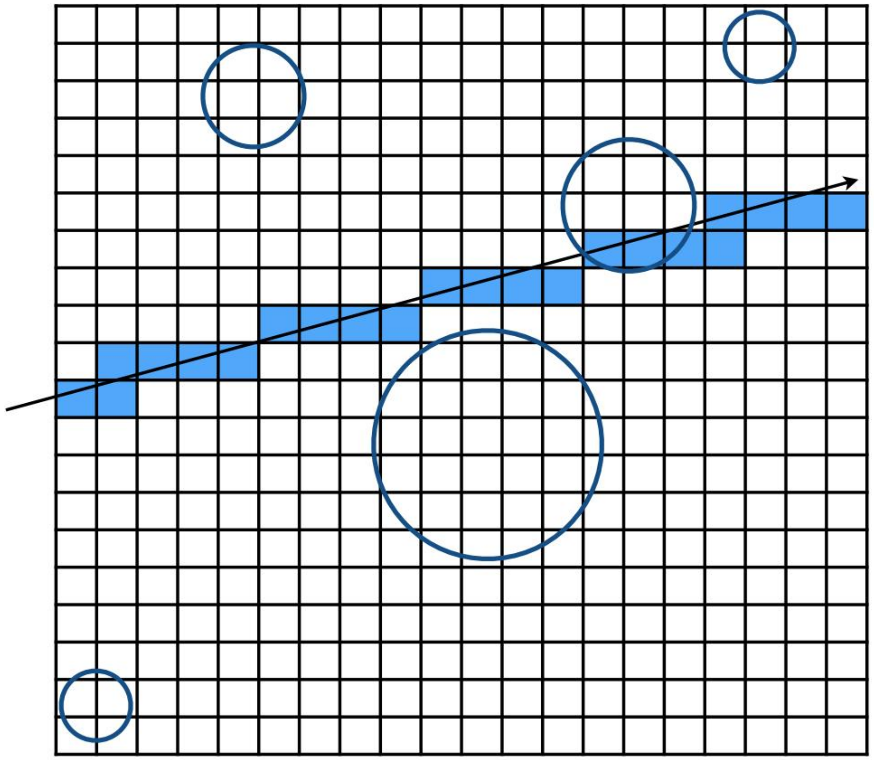


Step through grid in ray traversal order

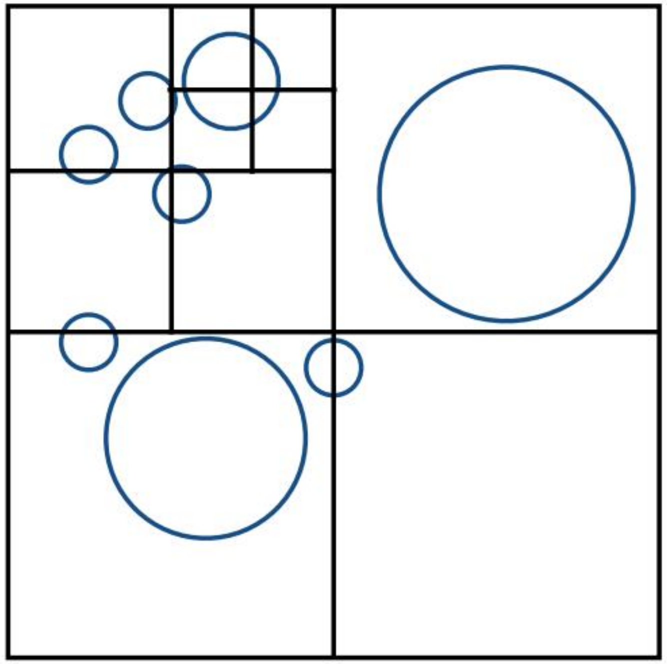
For each grid cell
Test intersection
with all objects
stored at that cell

Uniform Partition (Grids)

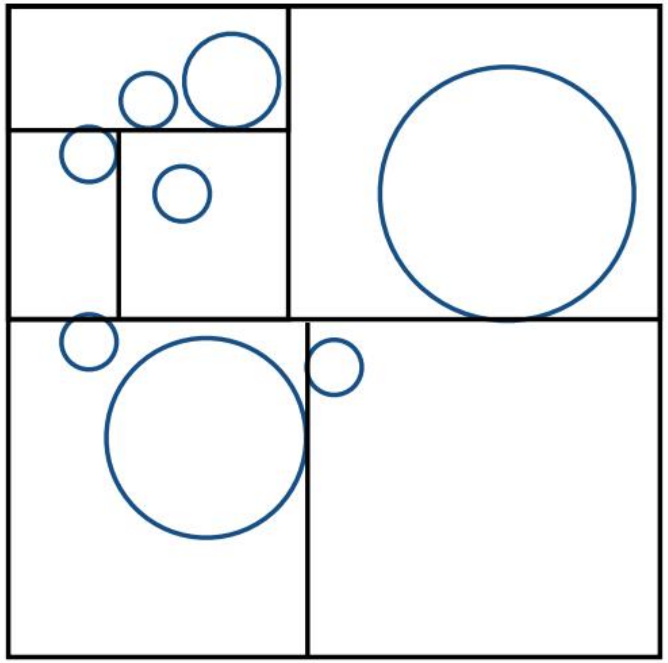
- High-resolution grids: high quality, but high computation cost
- Low-resolution grids: efficient computation, but poor quality



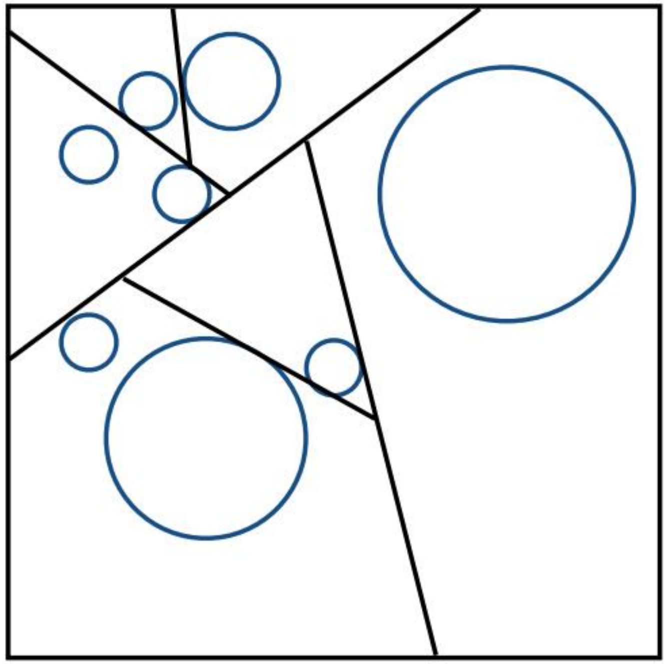
Spatial Partitions



OcTree



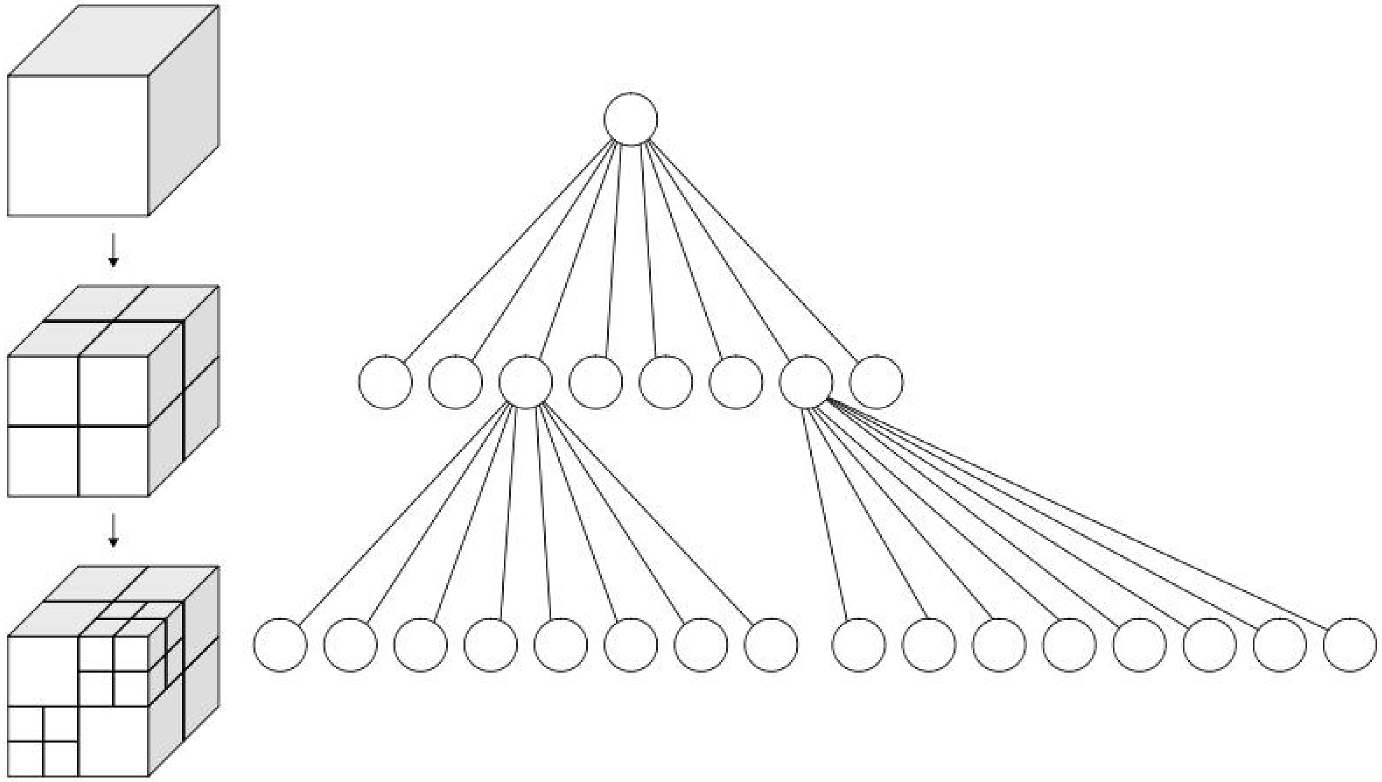
KD-Tree



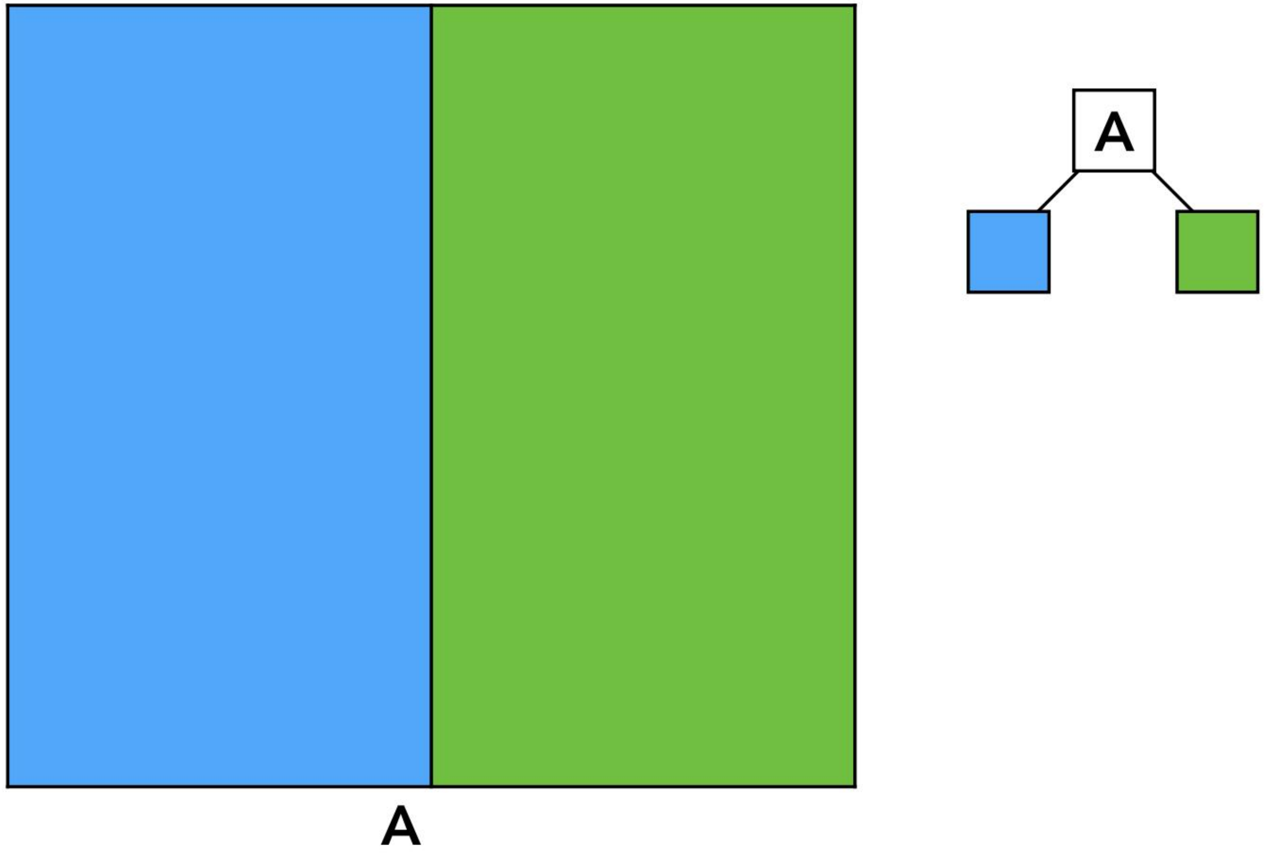
BSP-Tree

OcTree

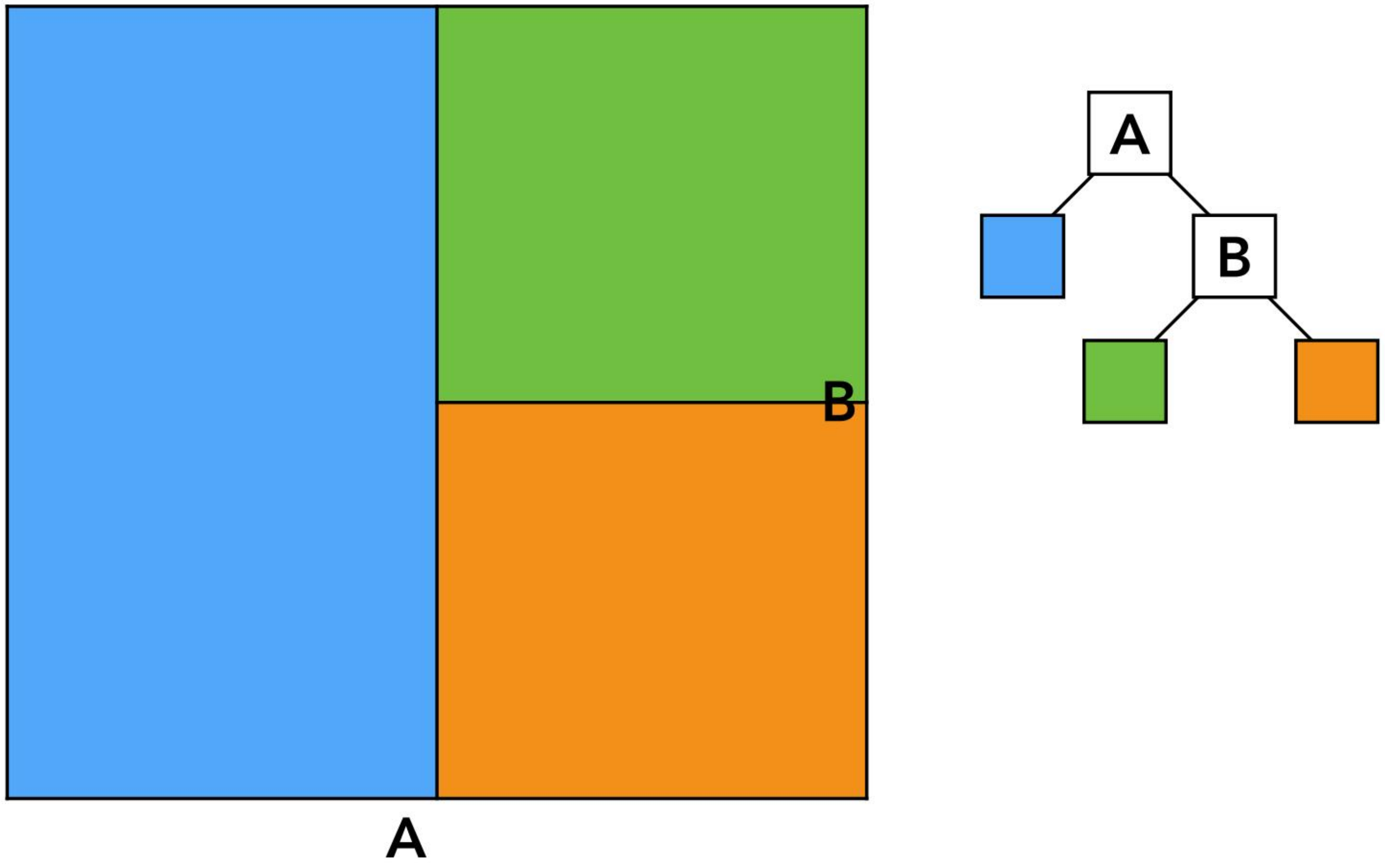
- Each internal node has exactly eight children
- Split around a point



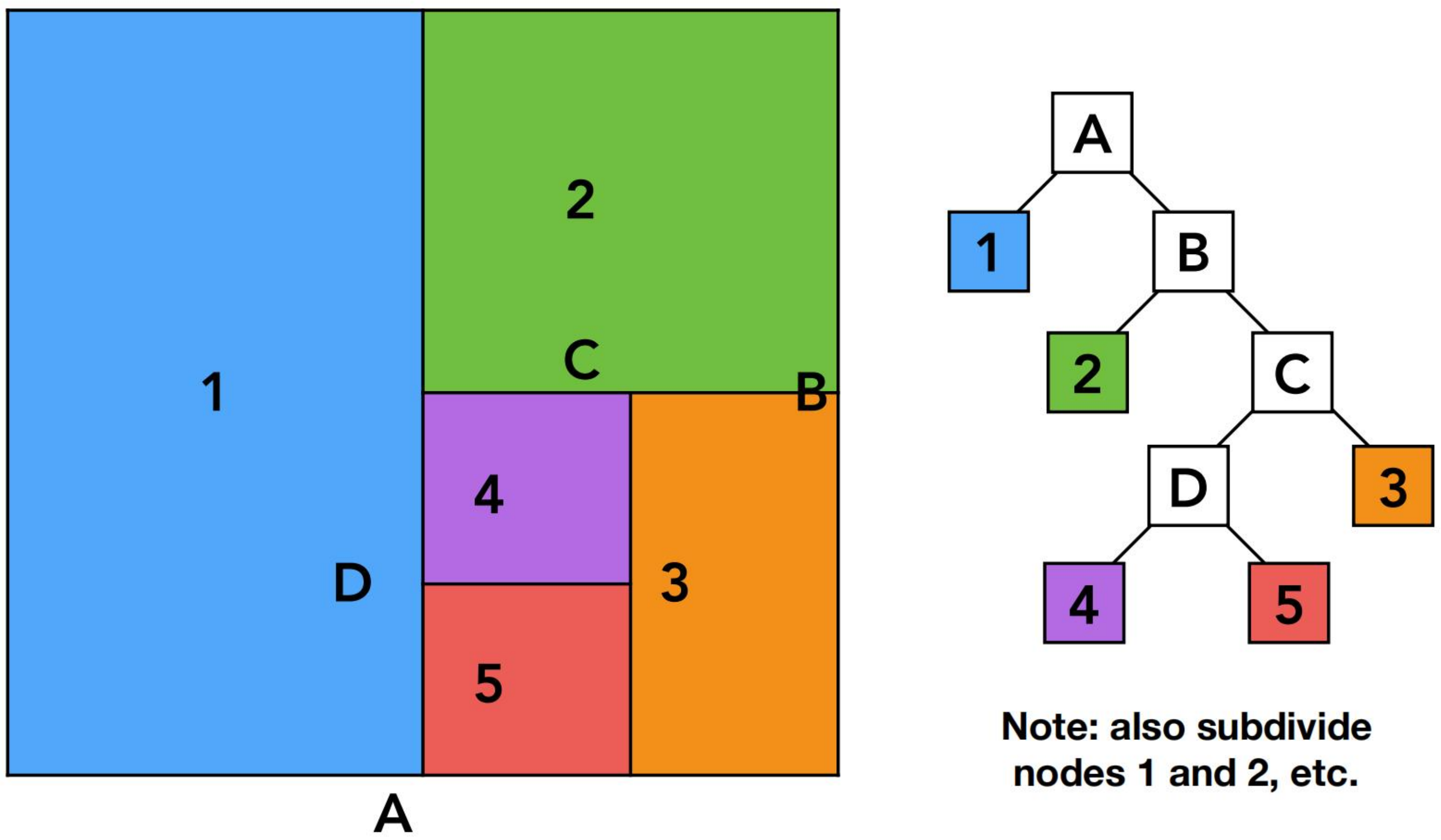
KD-Tree



KD-Tree



KD-Tree



Note: also subdivide nodes 1 and 2, etc.

KD-Tree

Split along a dimension

Internal nodes store

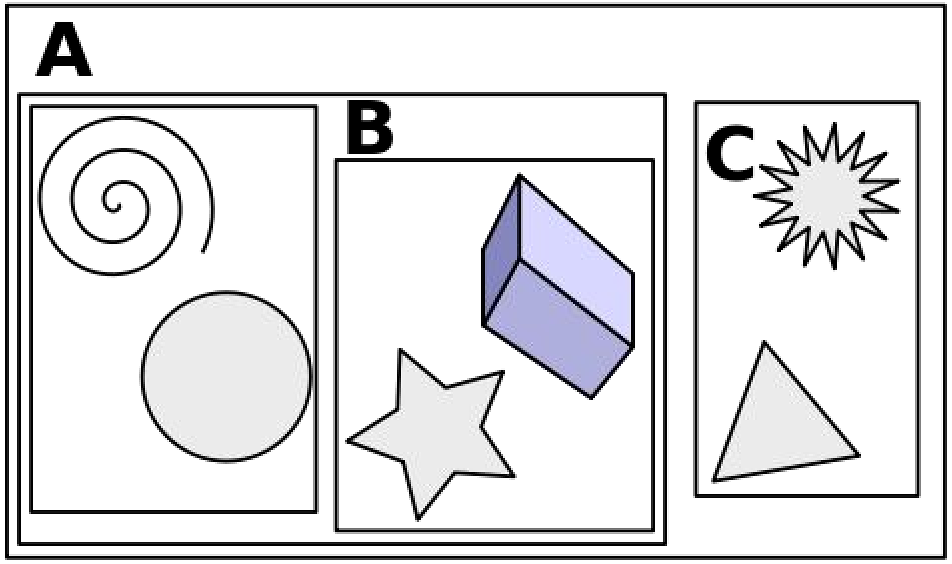
- split axis: x-, y-, or z-axis
- split position: coordinate of split plane along axis
- children: pointers to child nodes
- No objects are stored in internal nodes

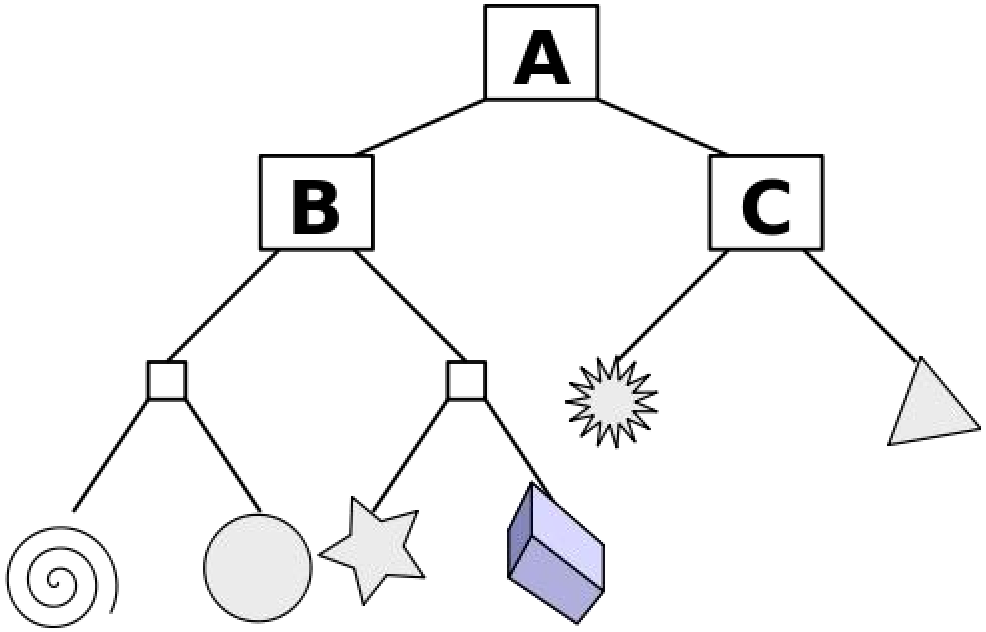
Leaf nodes store

- list of objects

Bounding Volume Hierarchy (BVH)

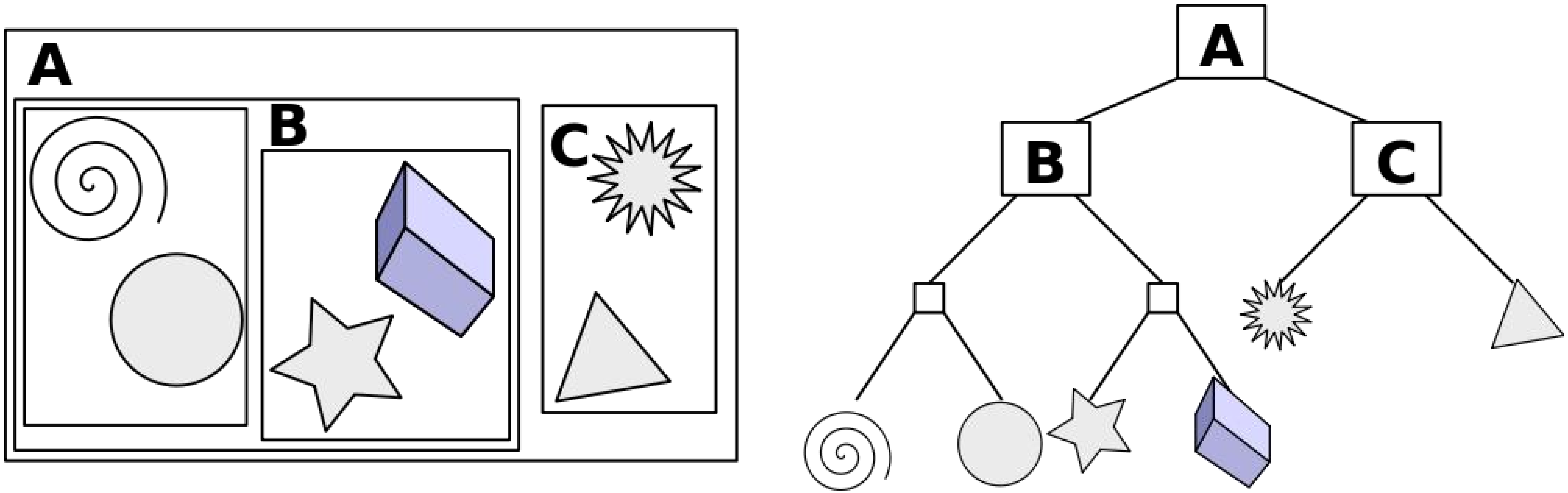
- Primitives are stored in the leaves
- Each node stores a bounding box of the primitives in the nodes beneath it





Bounding Volume Hierarchy (BVH)

- BVHs are more efficient to build than kd-trees
- But less efficient in ray tracing



Thank you