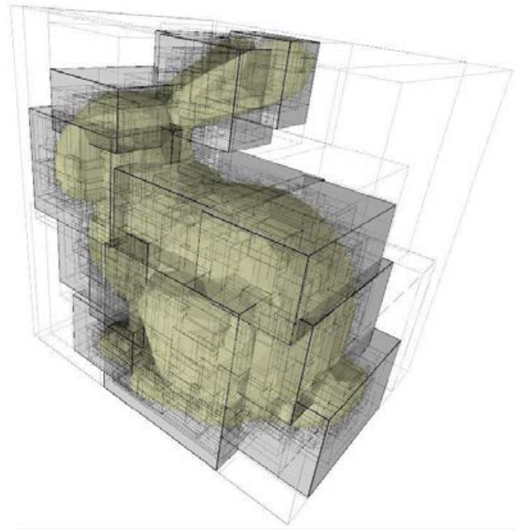
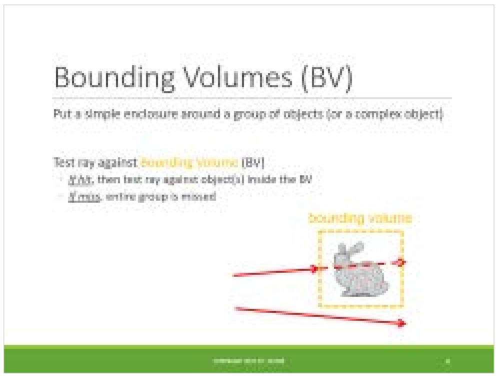
Acceleration Structures

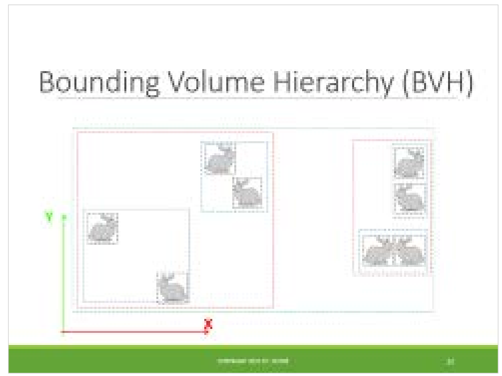


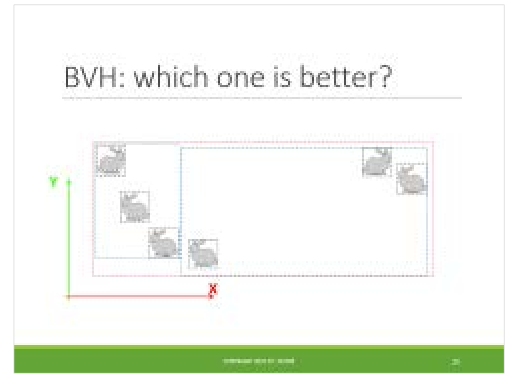
FUNDAMENTALS OF COMPUTER GRAPHICS

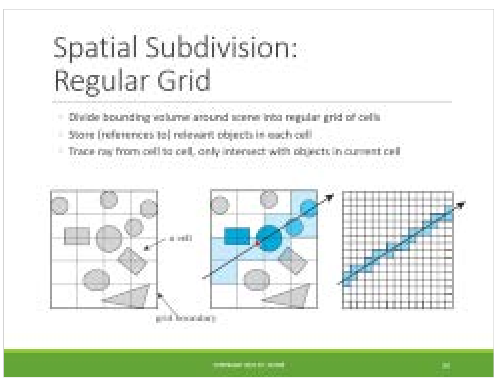
PHILIP DUTRÉ
DEPARTMENT OF COMPUTER SCIENCE

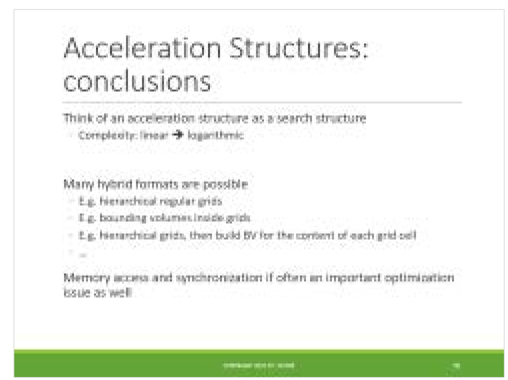
Overview Lecture











Relevant sections in book: Chapter 19, 22
(Illustrations from *Ray Tracing From The Ground Up*, *Physically-Based Rendering*, *Fundamentals of Computer Graphics*)
(Page numbering might skip some slides due to 'hidden' slides in my presentation.)

Algorithmic complexity of tracing rays?

Naïve approach:

- Test each ray vs each object
  ➔ computation time ~ #objects . #pixels . #lights

Acceleration structures:

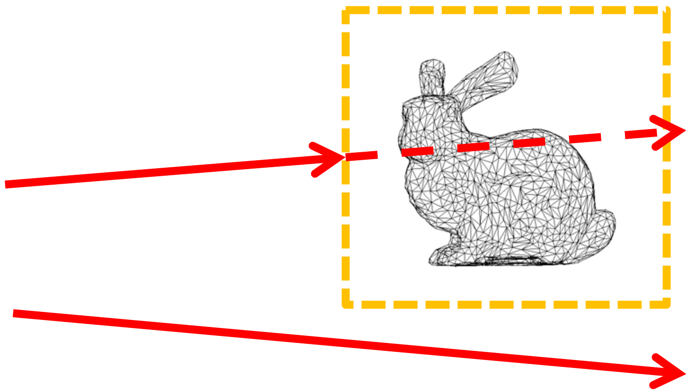
- Decrease #objects to be tested per ray
- Many approaches:
  - Bounding volumes, regular grids, hierarchical grids, …
- Finding the first intersection along a ray is essentially a search problem!

Bounding Volumes (BV)

Put a simple enclosure around a group of objects (or a complex object)

Test ray against Bounding Volume (BV)

- *<u>If hit</u>*, then test ray against object(s) inside the BV
- *<u>If miss</u>*, entire group is missed



Cost of Bounding Volumes

$$\underbrace{\text{cost\_BV} + \text{cost\_group} * \text{probability\_ray\_hits\_BV}}_{\text{cost per ray when using BV}} < \underbrace{\text{cost\_group}}_{\text{cost without BV}}$$

We want: cost_BV + cost_group * probability_ray_hits_BV < cost_group

cost_BV < cost_group * (1 - probability_ray_hits_BV)

cost_BV < cost_group * probability_ray_misses_BV

should be cheap

(simple intersection with BV)

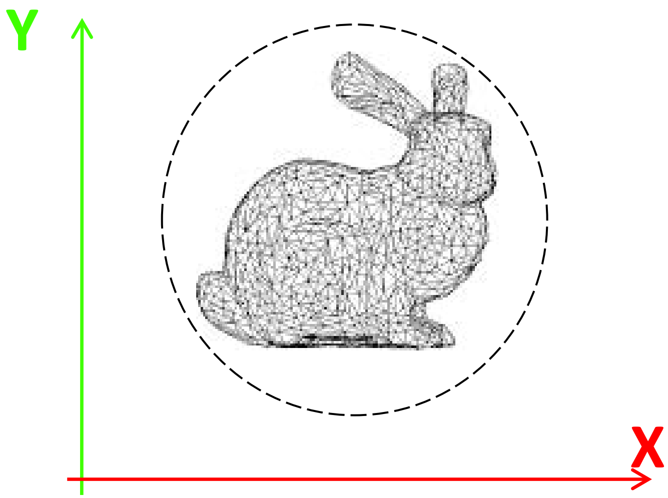
should be large

(BV should be tight)

Cheap vs. Tight

Spheres

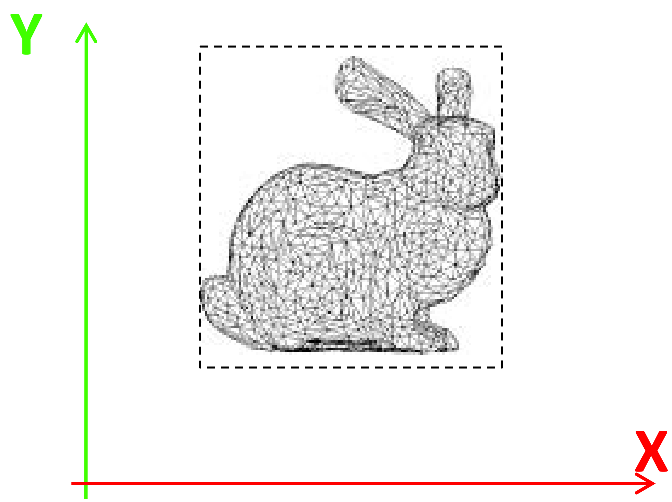
- easy to intersect, not always very tight
- difficult to find optimal enclosing sphere



Cheap vs. Tight

Axis-aligned Bounding Boxes (AABBs)

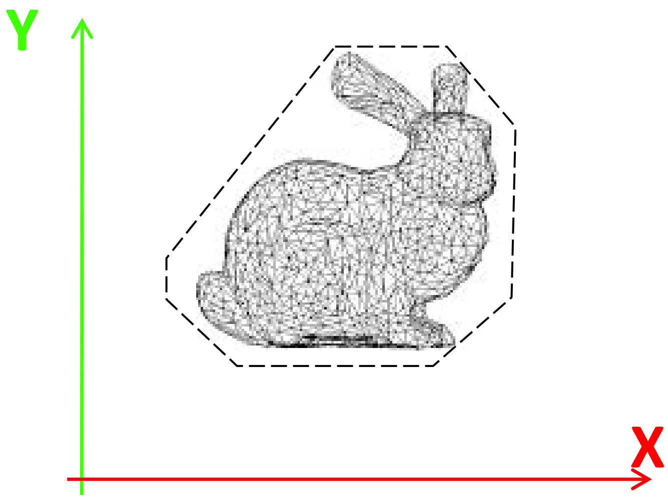
- easy to intersect, often tight
- easy to construct



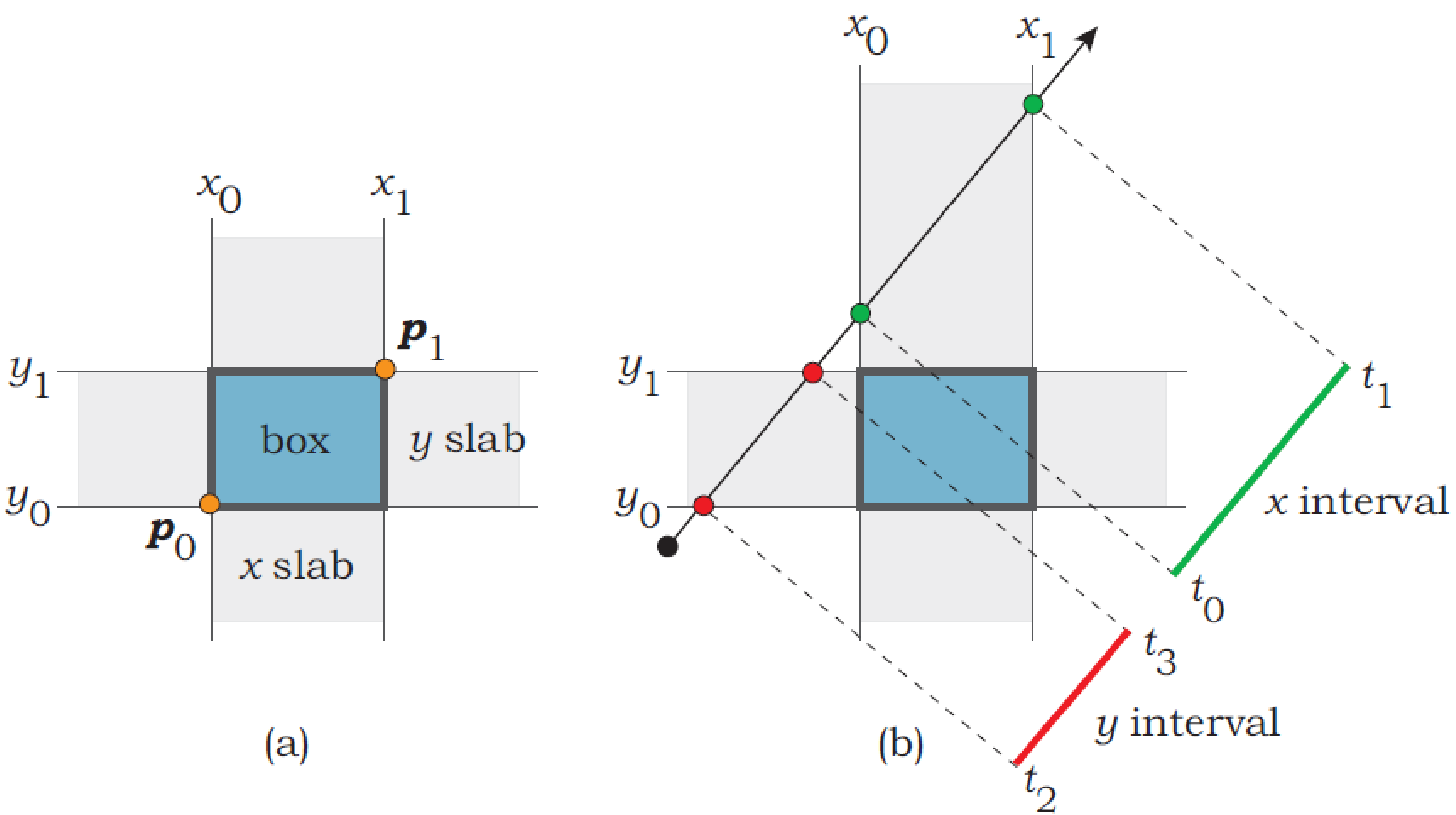
Cheap vs. Tight

Oriented Slabs

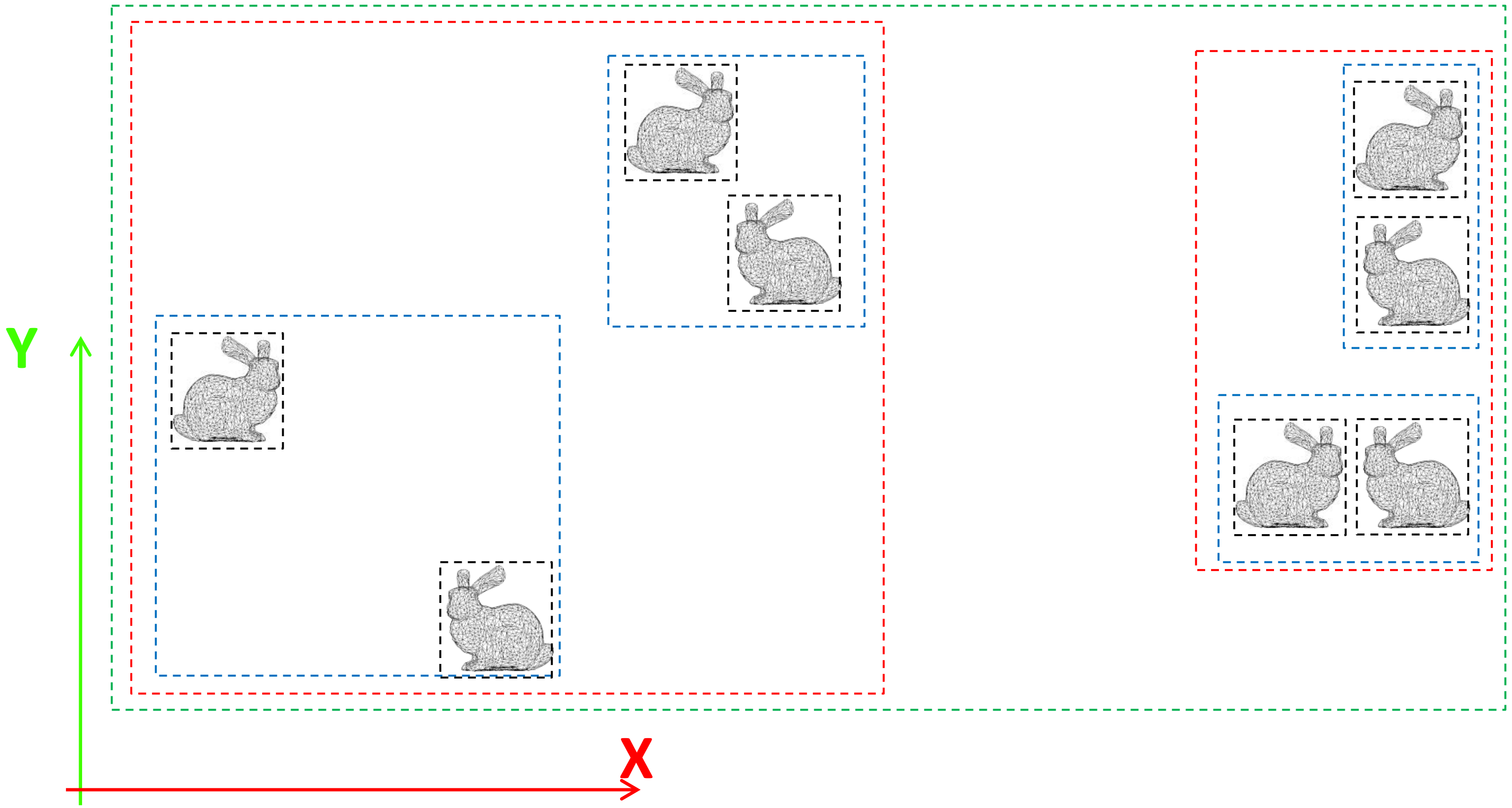
- relatively easy to intersect
- tighter for arbitrary objects



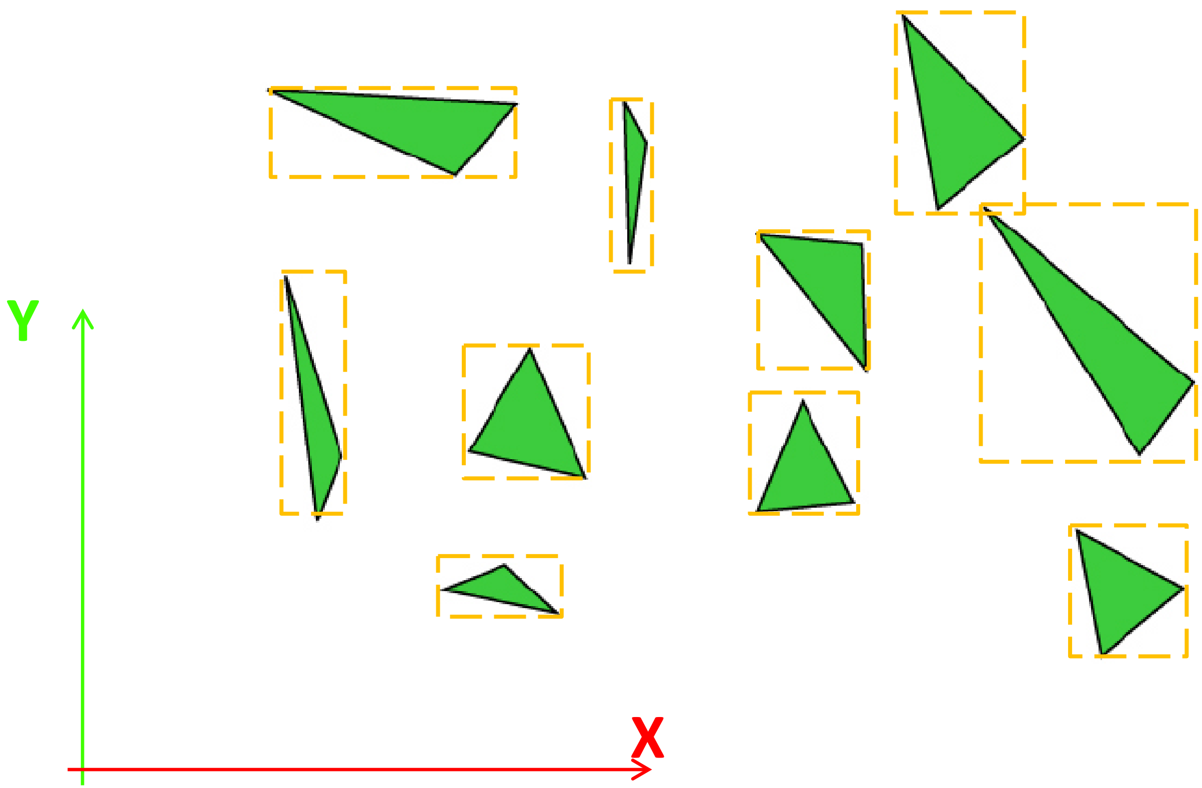
Axis-Aligned Bounding Box



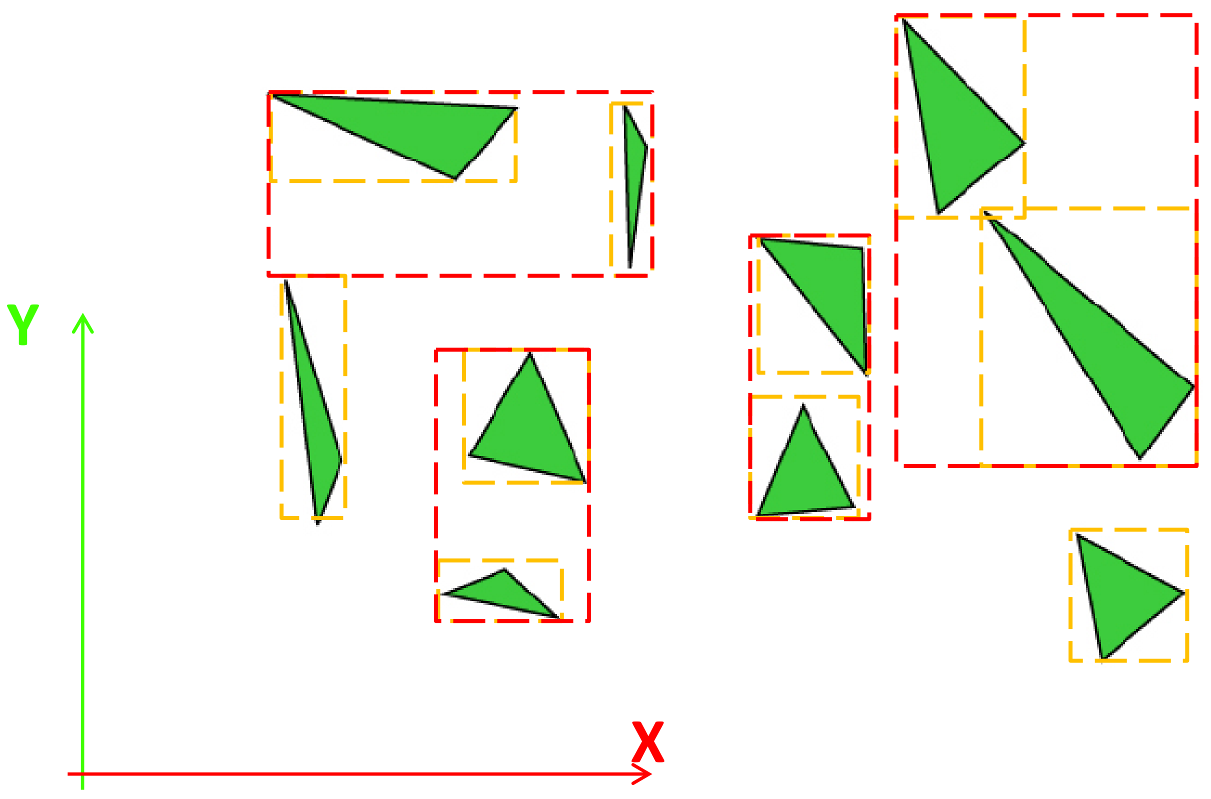
Bounding Volume Hierarchy (BVH)



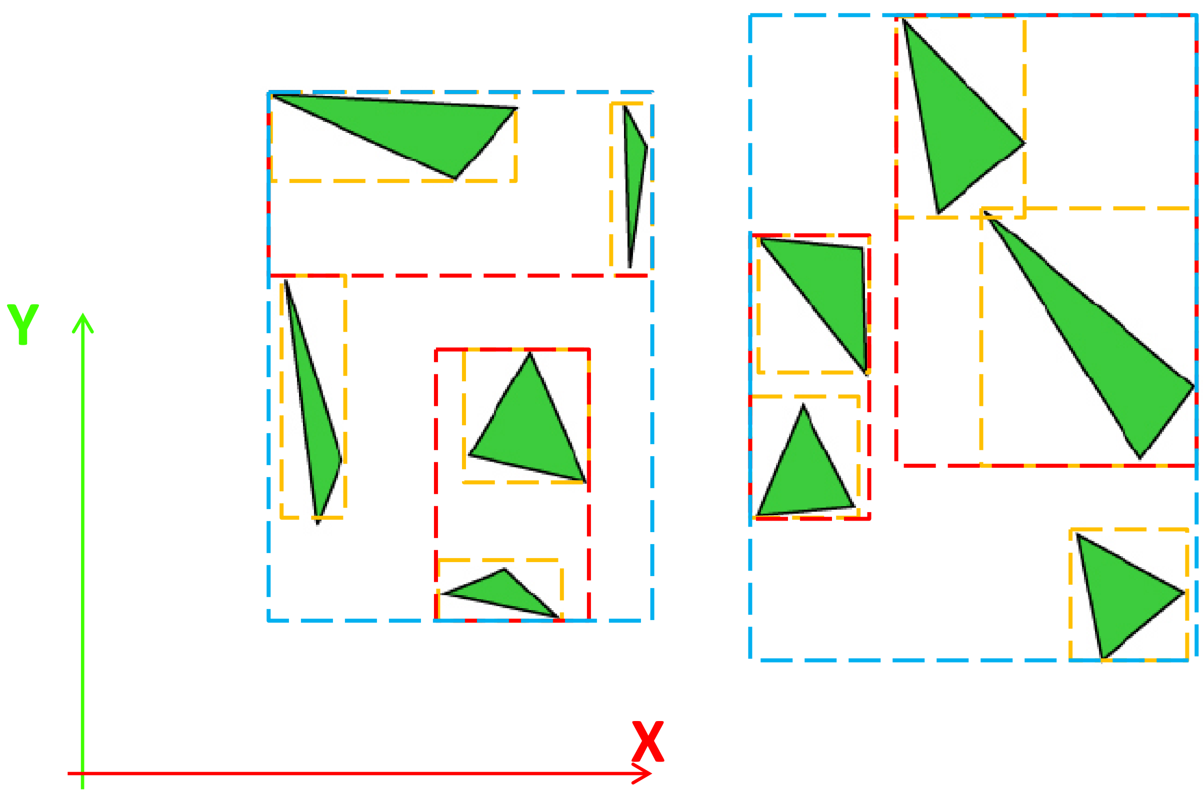
BVH: bottom-up construction



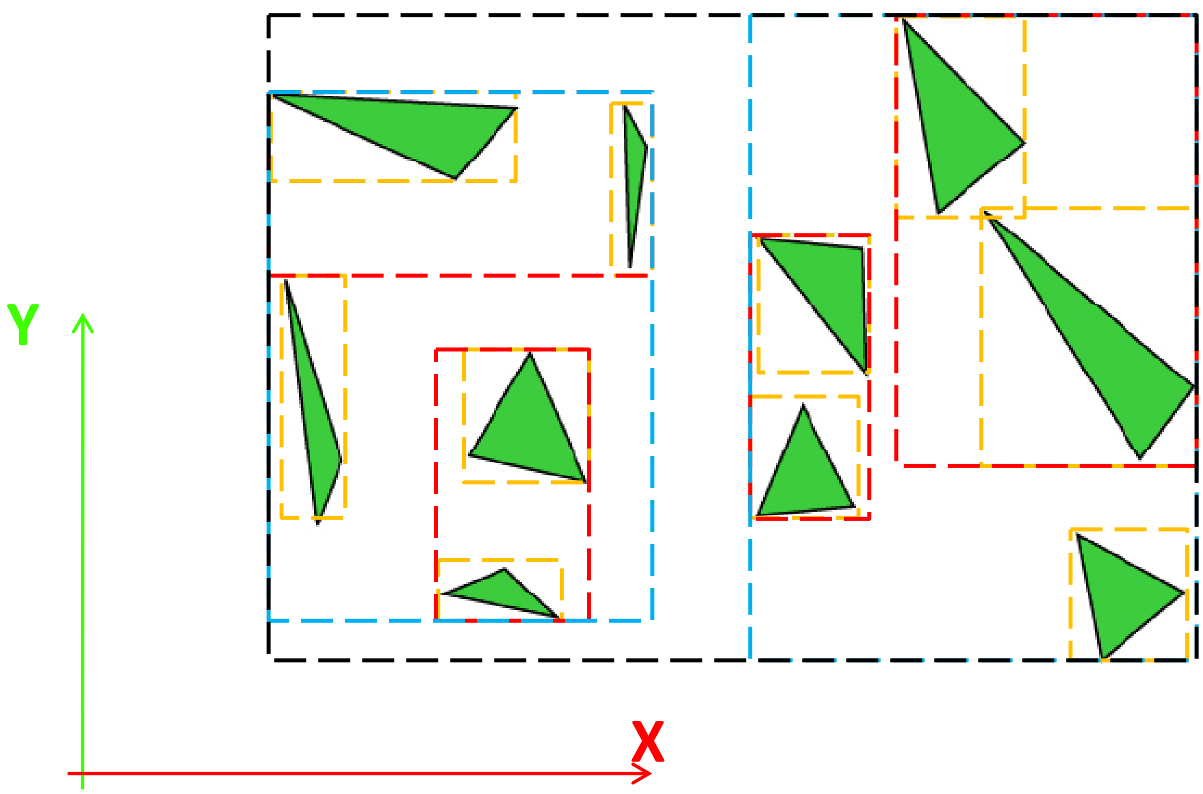
BVH: bottom-up construction



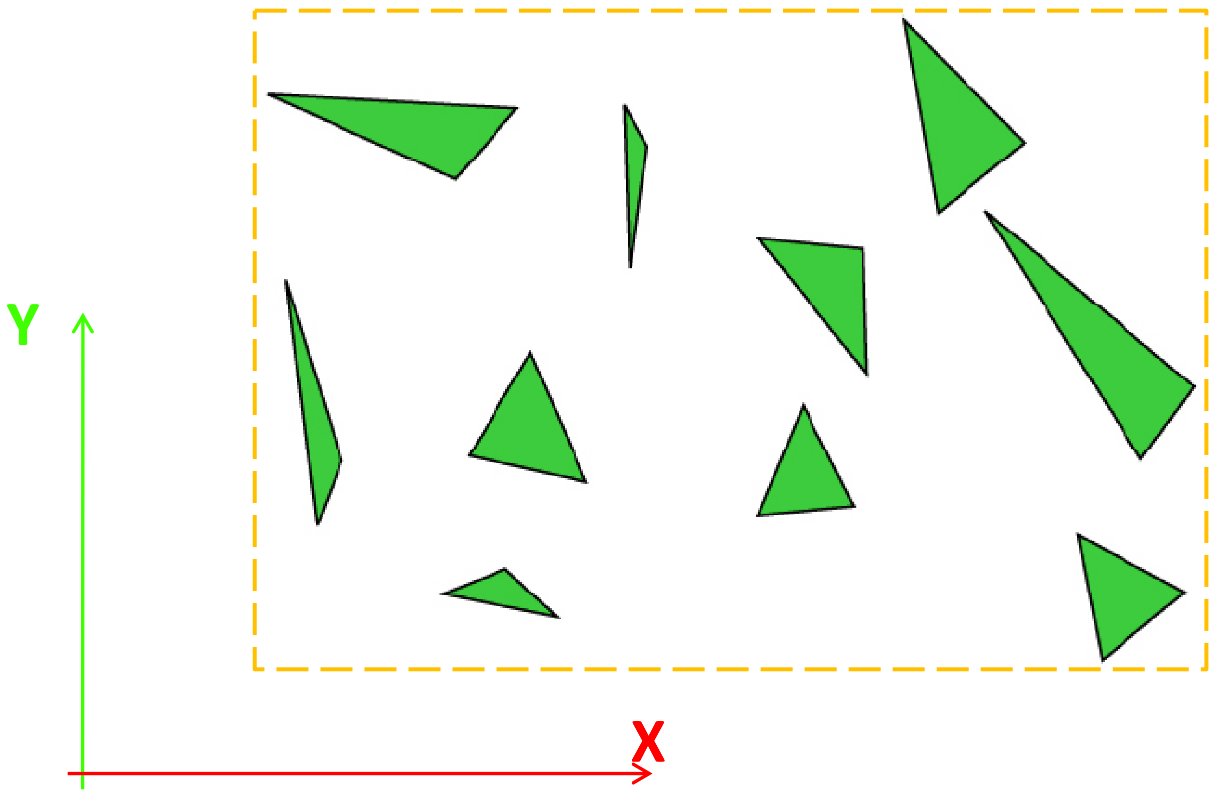
BVH: bottom-up construction



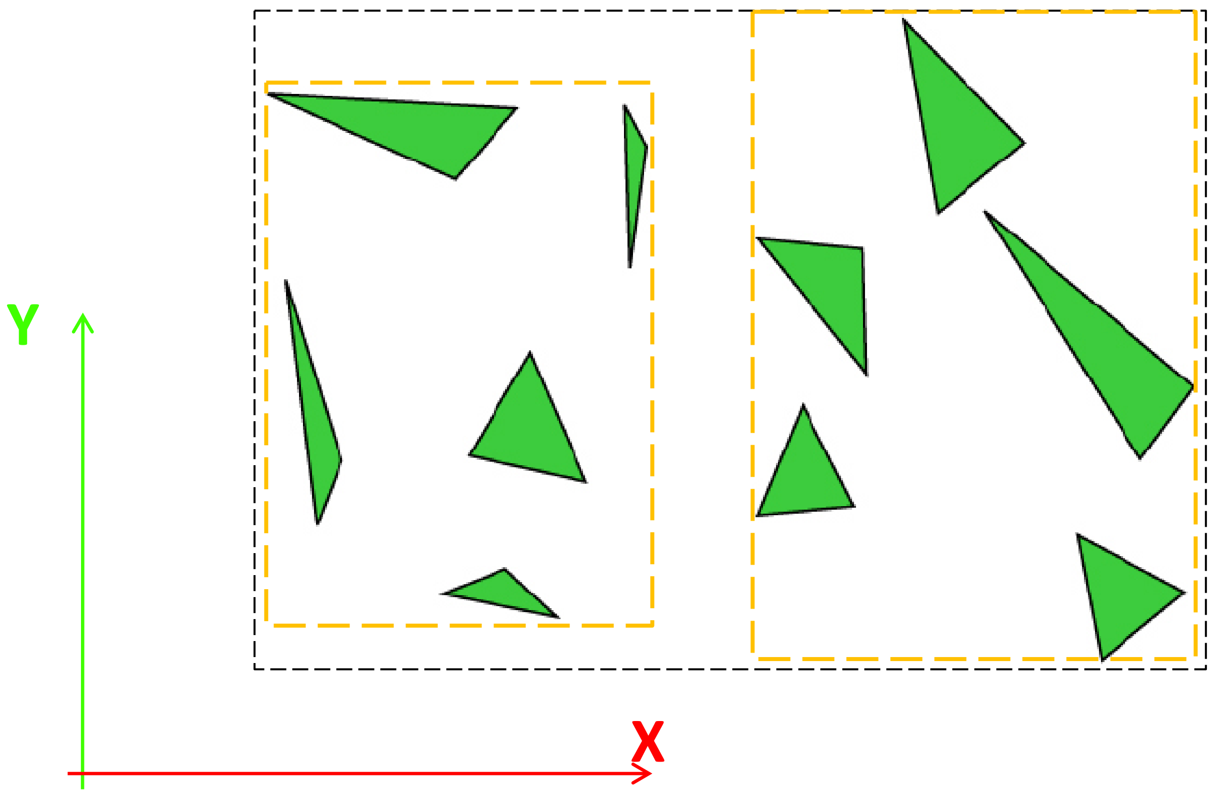
BVH: bottom-up construction



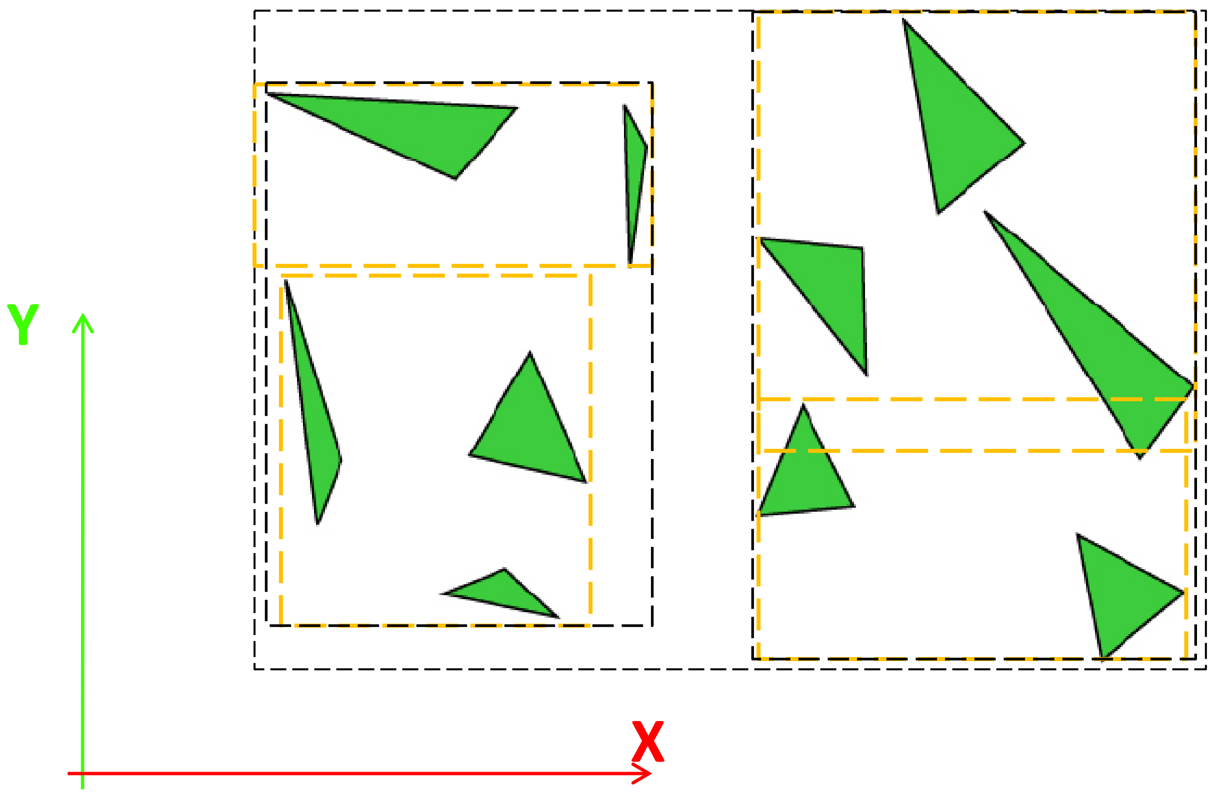
BVH: top-down construction



BVH: top-down construction



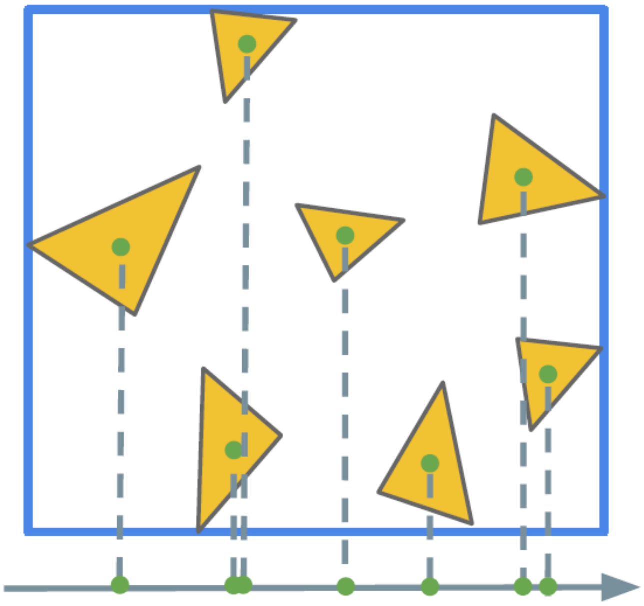
BVH: top-down construction

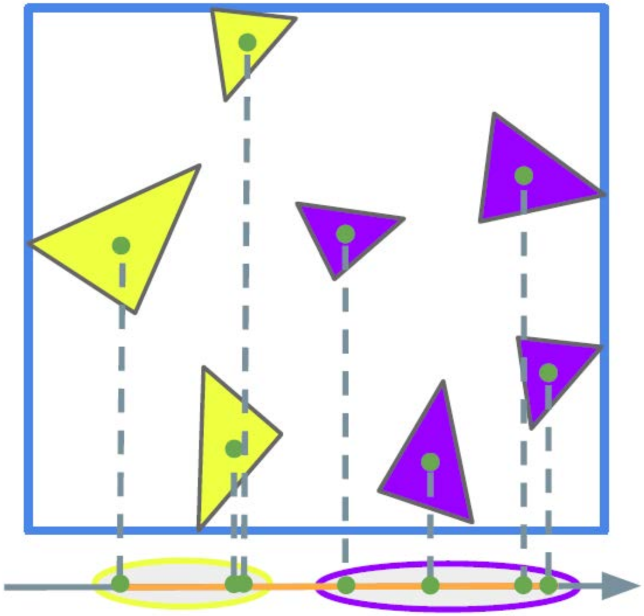


BVH: top-down construction

Use centroids of objects

- Split according to spatial median
- Spli according to object median

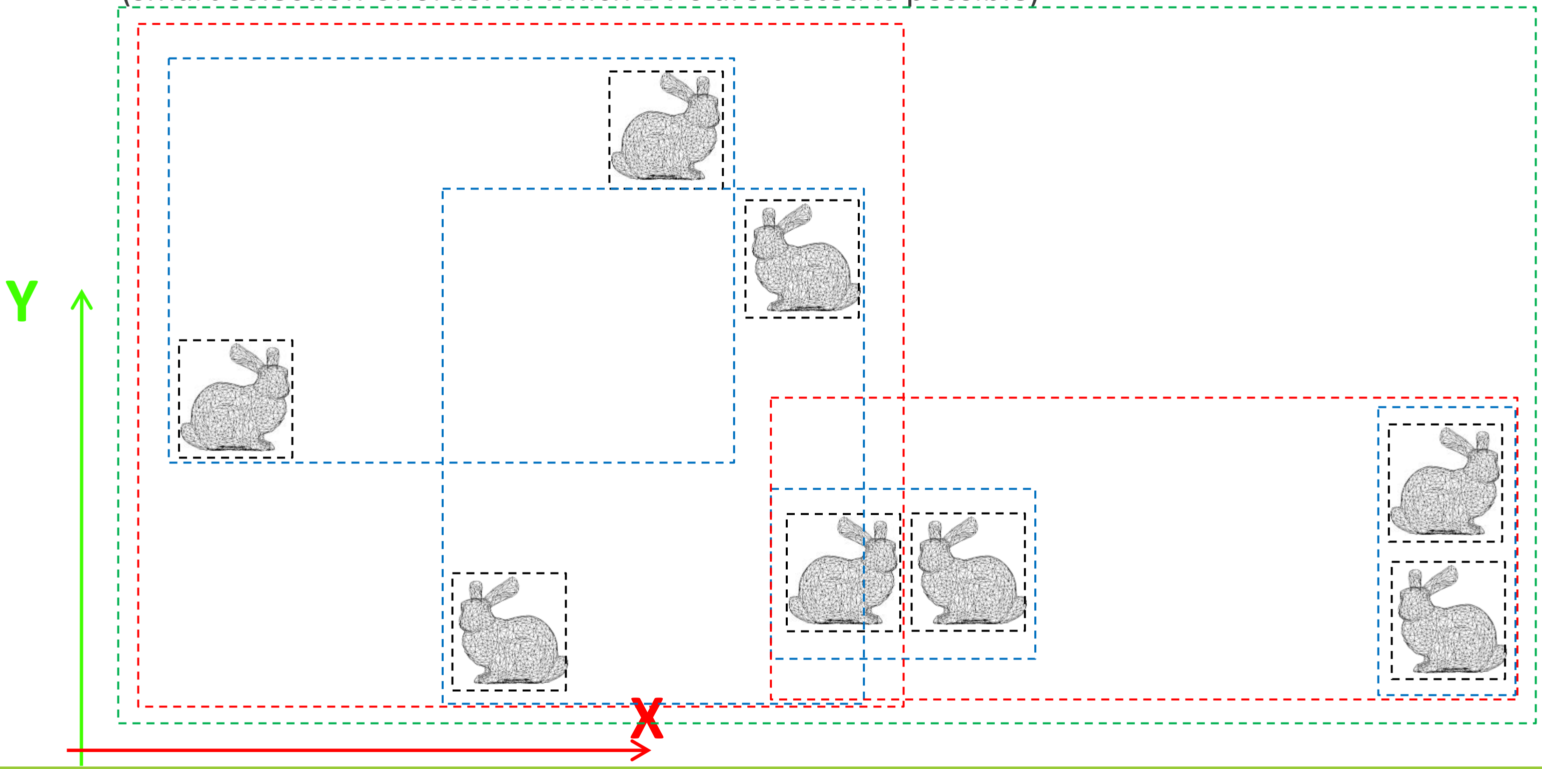




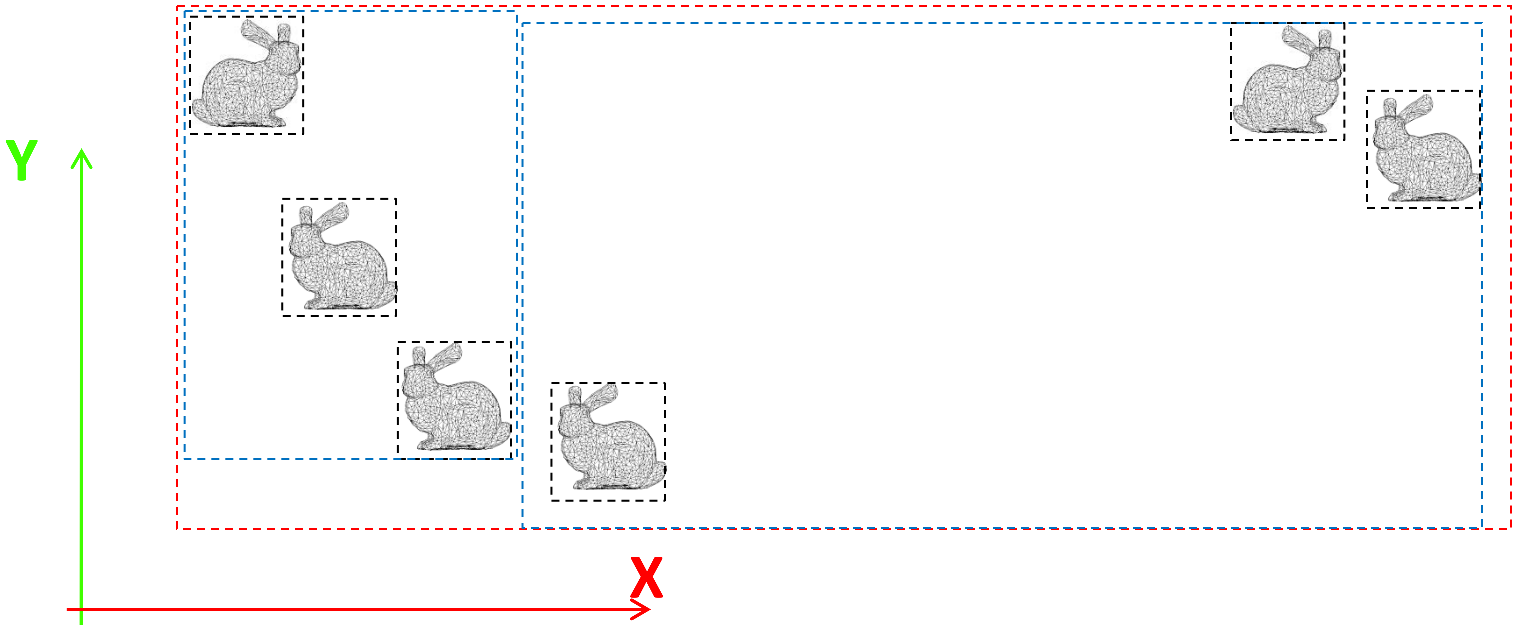
BVH: issues

BVs can overlap

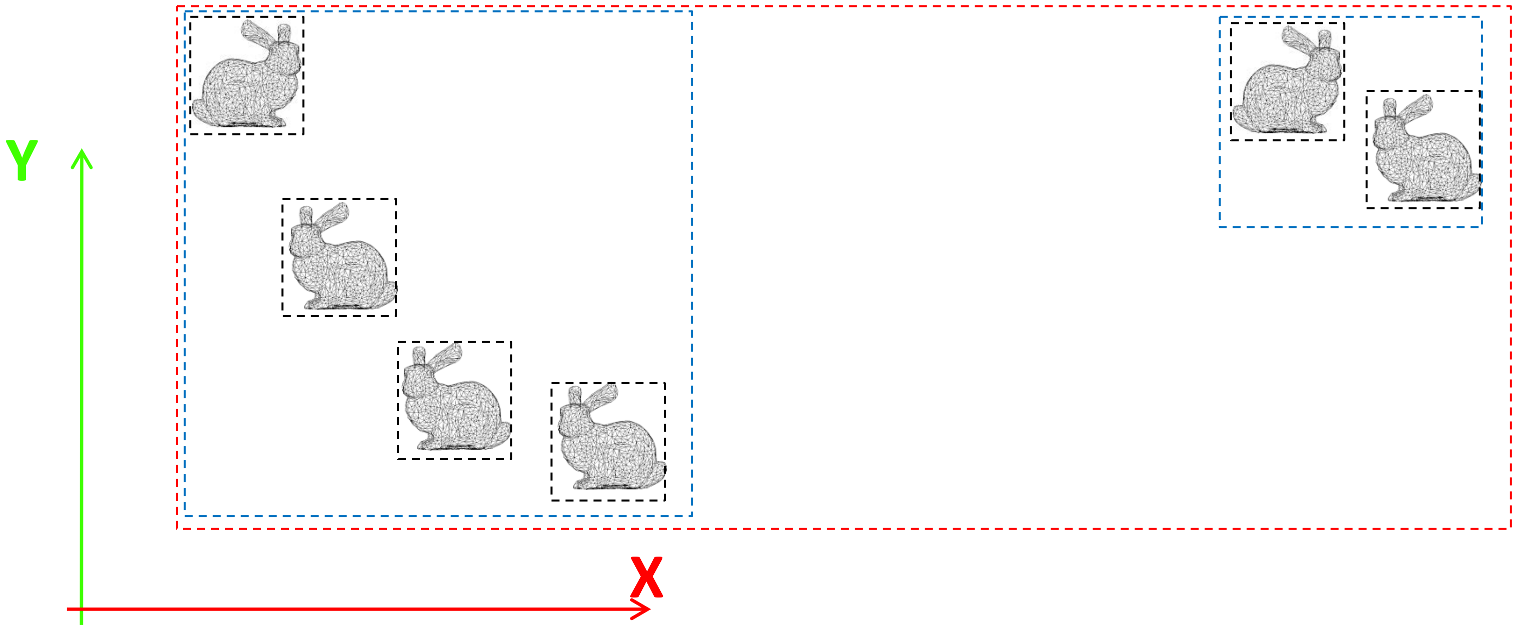
- Ray should be intersected with all BVs at same level in hierarchy
  - (smart selection of order in which BV's are tested is possible)



BVH: which one is better?

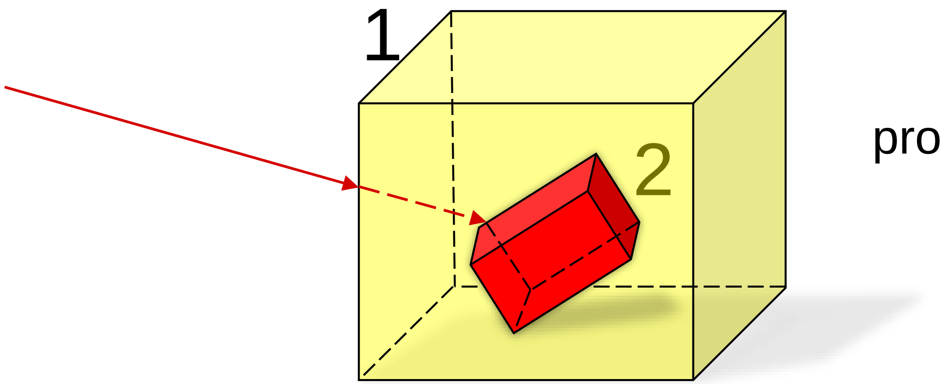


BVH: which one is better?



Surface Area Heuristic

“If a ray hits a 3D object, what is the probability it will hit another object enclosed in the original object?”

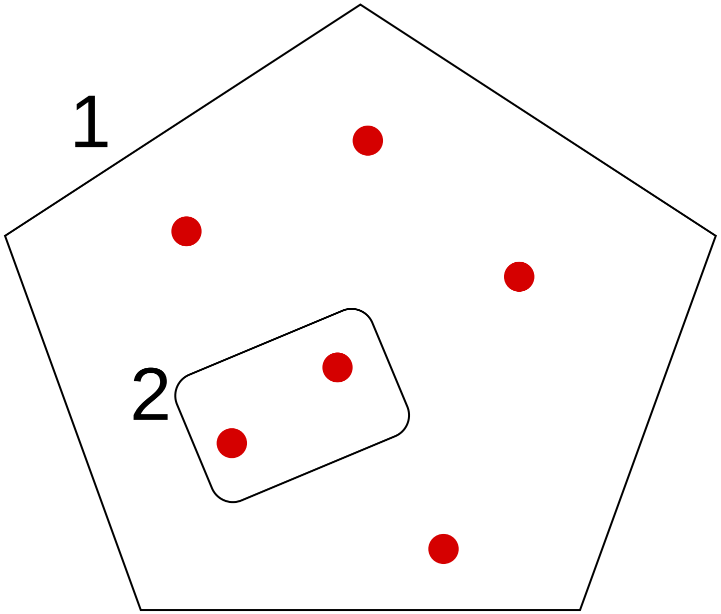


probability ray hits object 2, given it hits object 1?

(in BVH context: object 1 and 2 are both AABBs)

Surface Area Heuristic

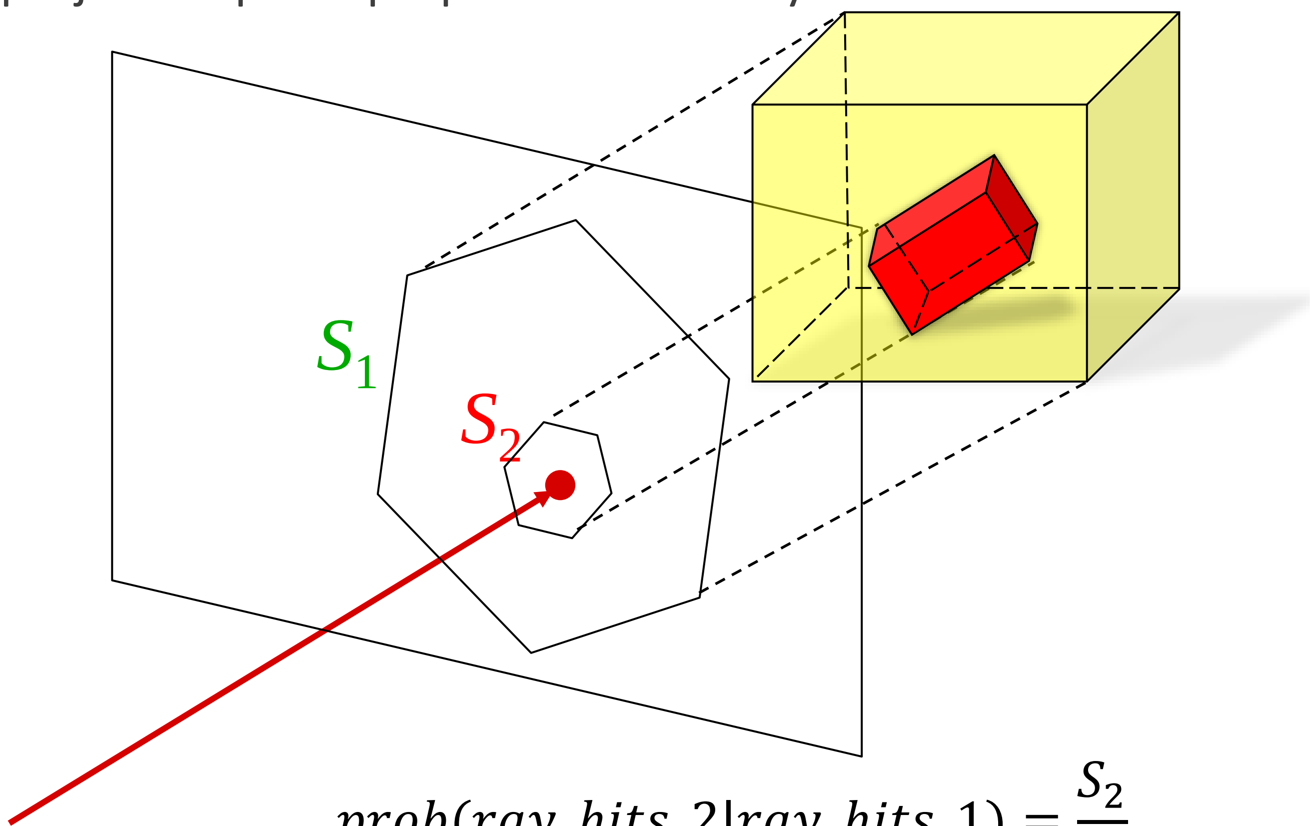
2D analogy: points in shape



$$prob(p \in 2 | p \in 1) = \frac{Surface(2)}{Surface(1)}$$

Surface Area Heuristic

3D: projection plane perpendicular to ray



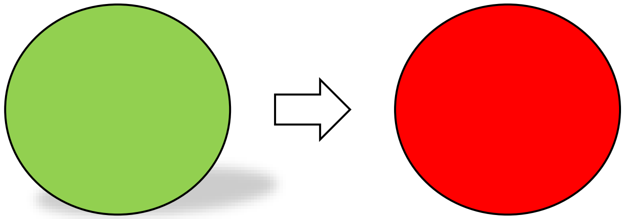
$$prob(ray\_hits\_2|ray\_hits\_1) = \frac{S_2}{S_1}$$

Surface Area Heuristic

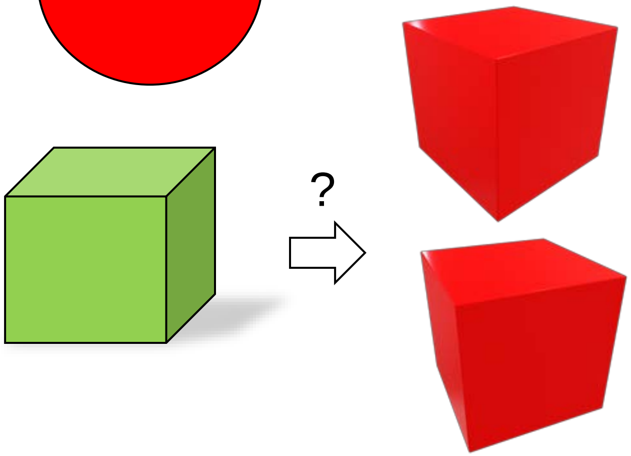
Average projected area for all possible ray directions?

"For a convex object, the average (orthogonal) projected area equals ¼ of the surface area"
(Cauchy, Crofton)

Sphere
- Area = $4\pi r^2$
- Average projected area = πr^2

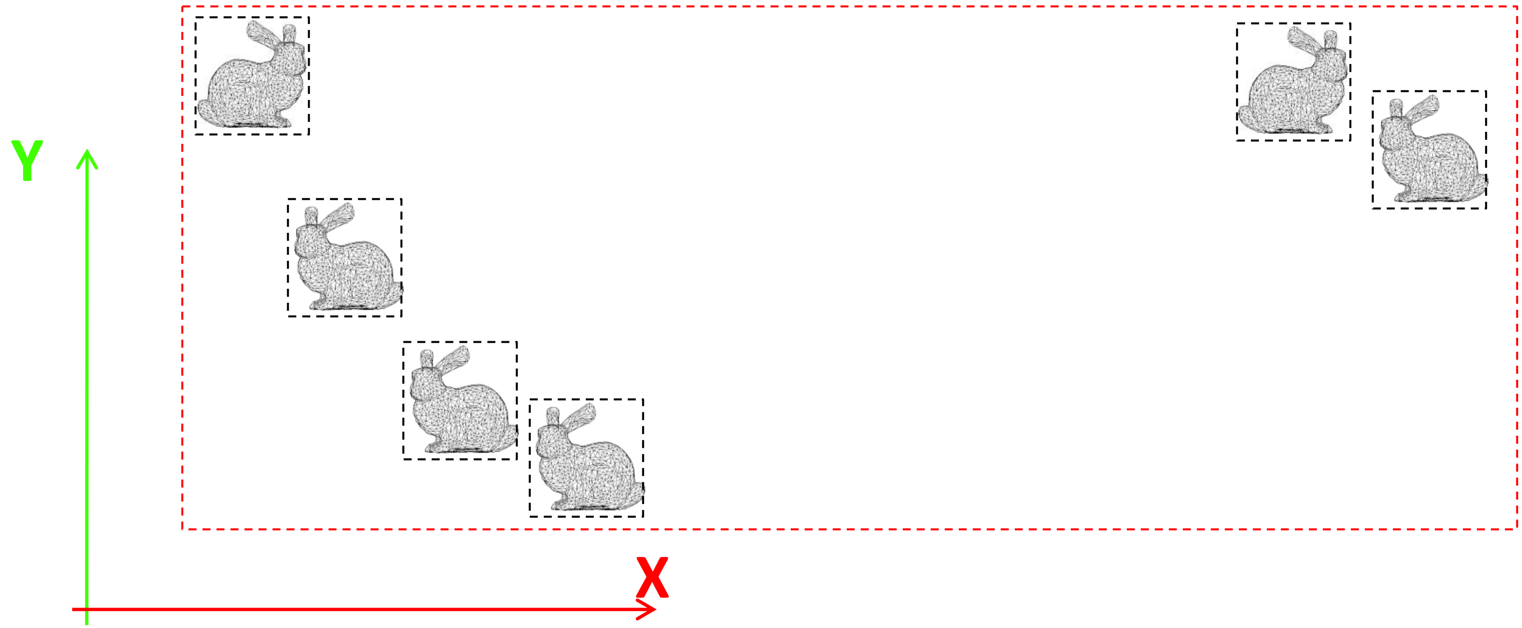


Cube
- Area = $6w^2$
- Average projected area =$1.5w^2$



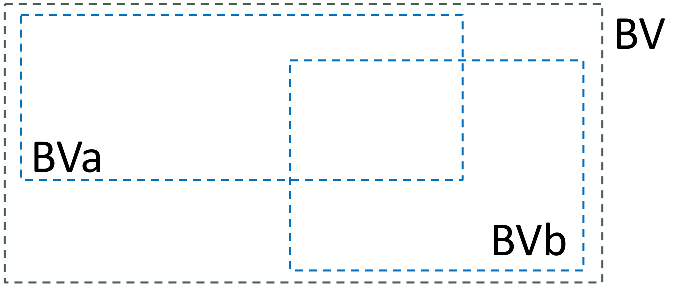
Bounding Volume Hierarchies: which one is better?

Evaluate heuristic greedily during top-down construction



Bounding Volume Hierarchies: which one is better?

Cost of tracing a ray through a BV:



$$Cost(BV) = C_t + prob(hit\_BVa).Cost(BVa) + prob(hit\_BVb).Cost(BVb)$$

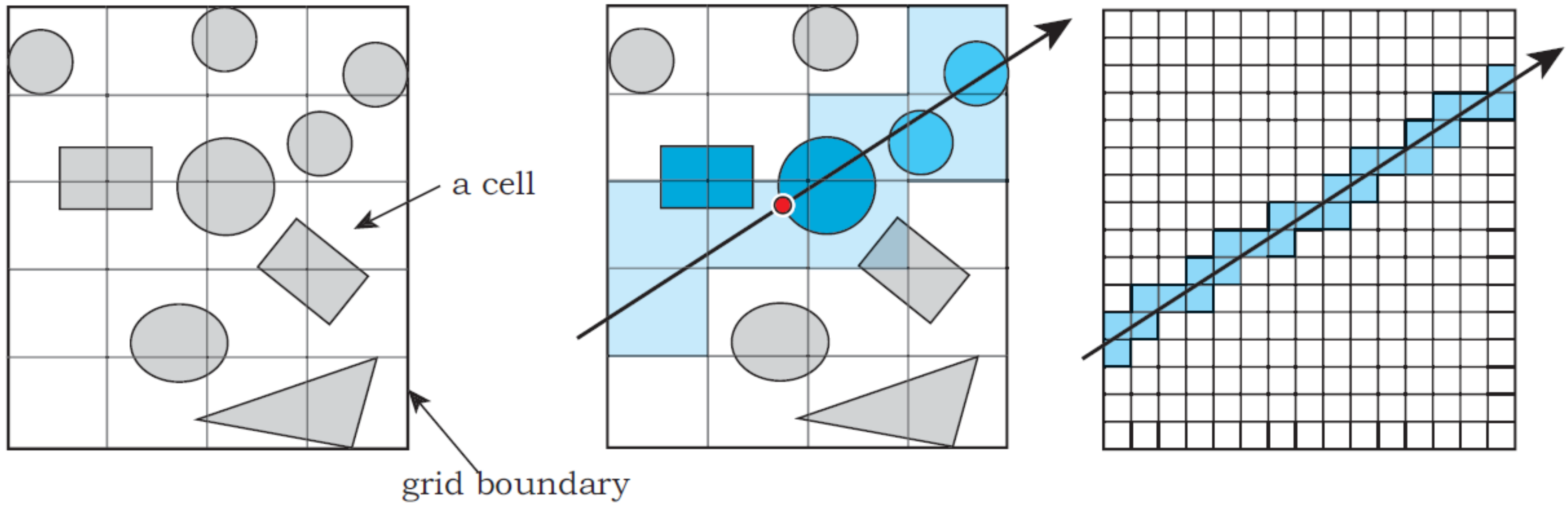
Fixed cost for traversing the cell
(determining exit point etc.)

Probability to hit bounding volume A / B:
proportional to surface area
of BVa or BVb vs surface area of parent BV

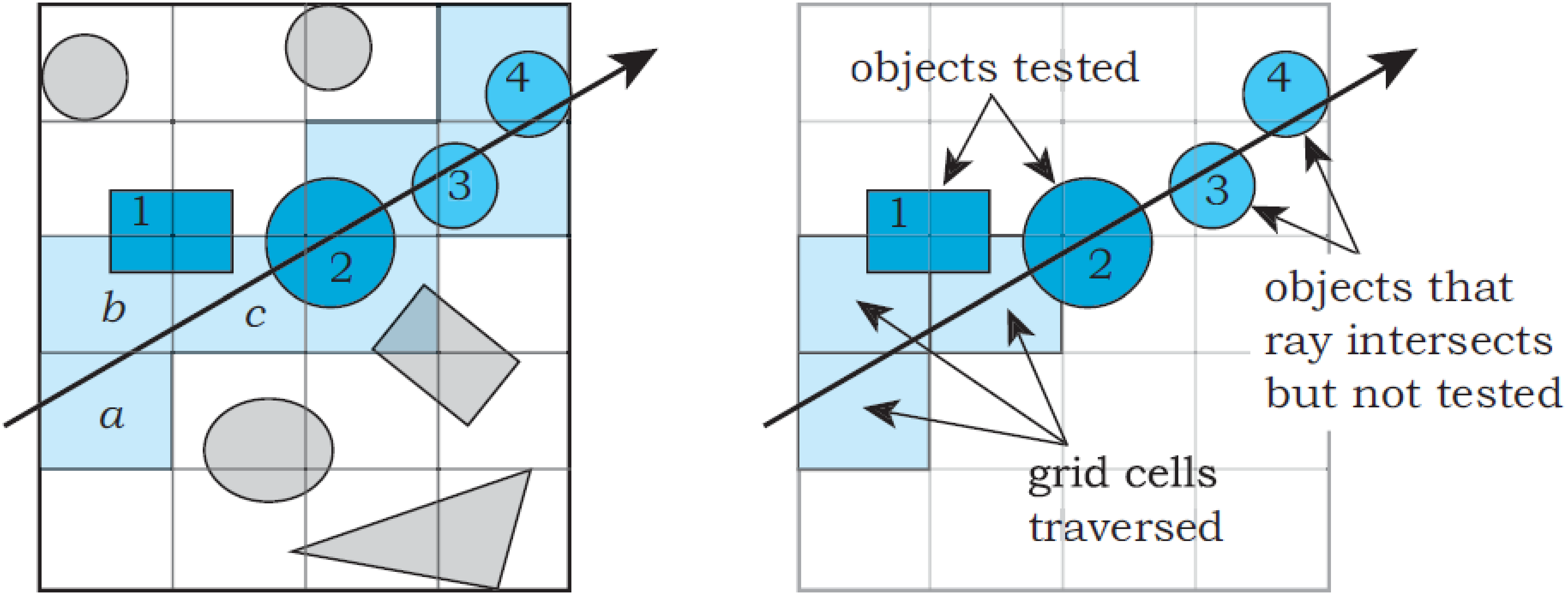
Estimate: proportional
to #triangles in BVa or BVb
(but really a recursive cost …)

Spatial Subdivision: Regular Grid

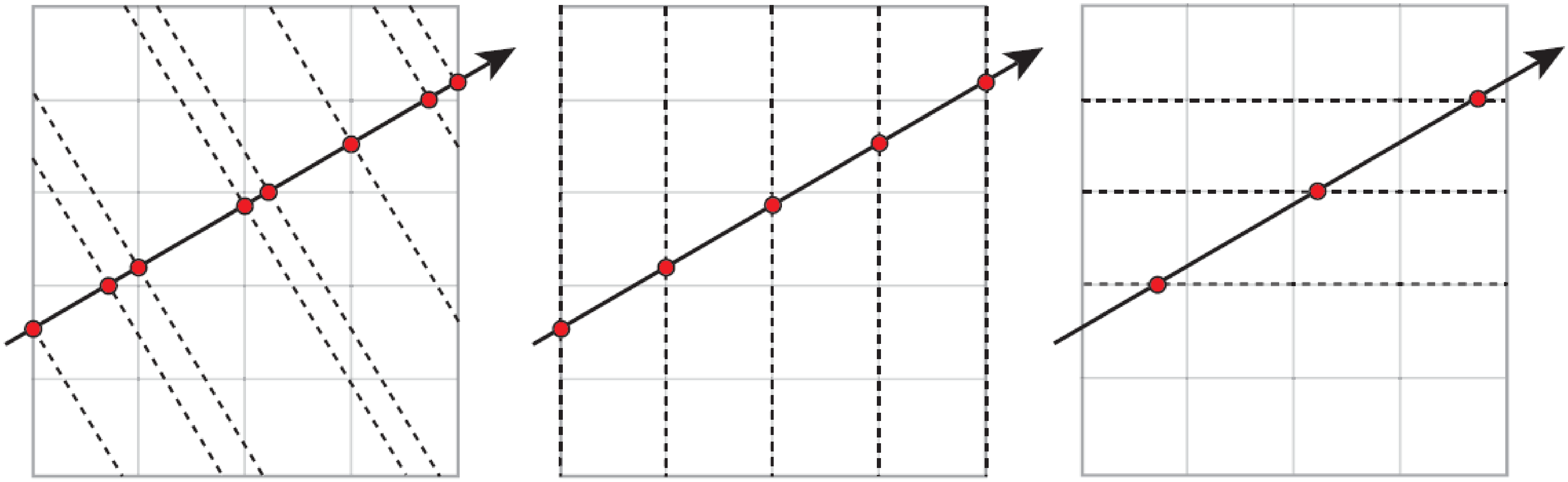
- Divide bounding volume around scene into regular grid of cells
- Store (references to) relevant objects in each cell
- Trace ray from cell to cell, only intersect with objects in current cell



Regular Grid: traversal



Regular Grid: traversal



Regular Grid: construction

Check bounding box of objects vs. regular grid of cells.

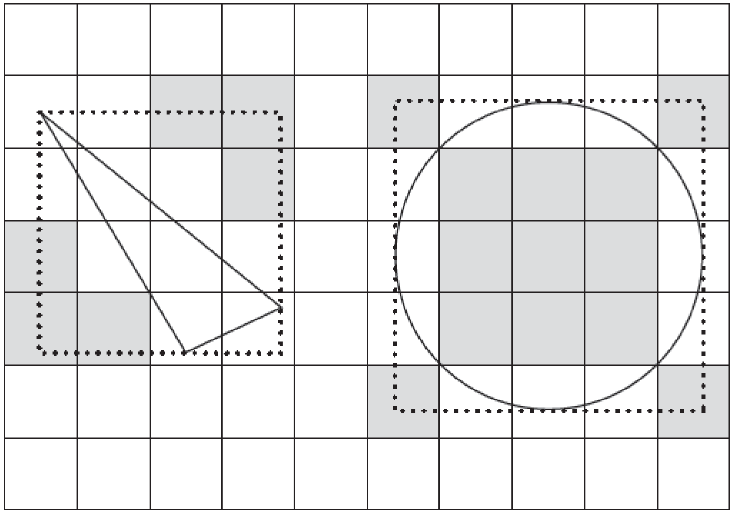
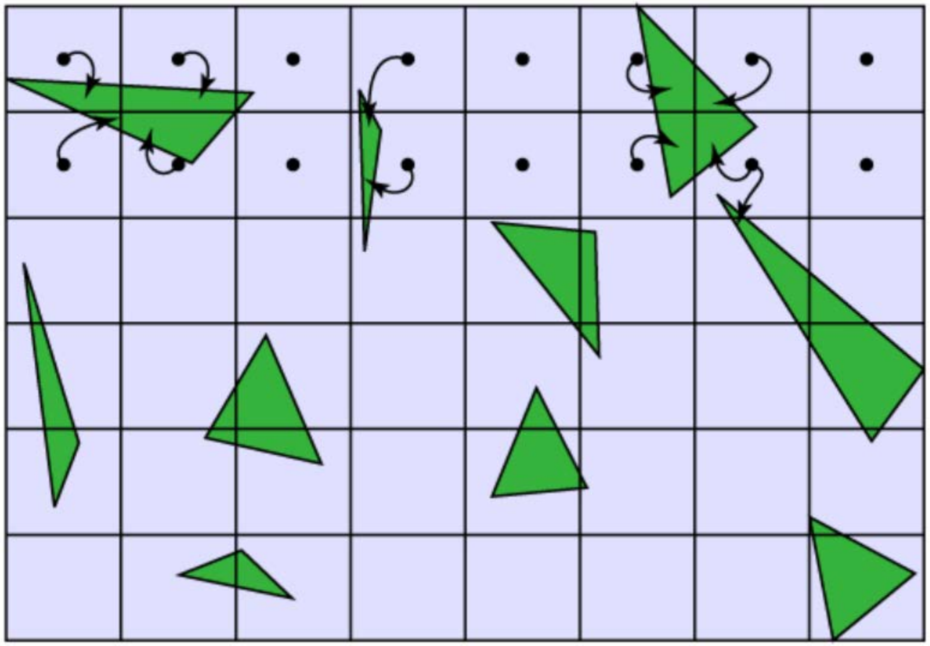


Figure 4.5: Two examples of cases where using the bounding box of a primitive to determine which grid voxels it should be stored in will cause it to be stored in a number of voxels unnecessarily. On the left, a long skinny triangle has a lot of empty space inside its axis-aligned bounding box, and it is unnecessarily added to the shaded voxels. On the right, the surface of the sphere doesn't intersect many of the voxels inside its bound, and they are also inaccurately included in the sphere's extent. While this error degrades performance, it doesn't lead to incorrect ray intersection results.

Regular Grid: storage

Grid subdivides the space, not the objects

- Store references to objects in each cell
  ➔ implies a collection of (object)refs per cell

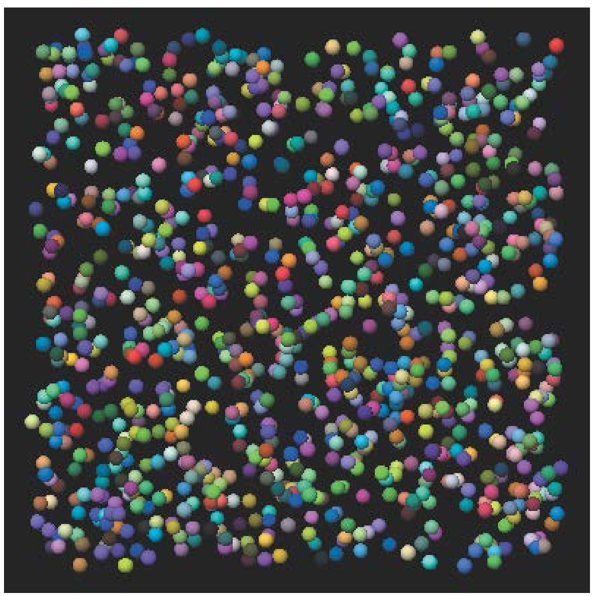


Regular Grid: results

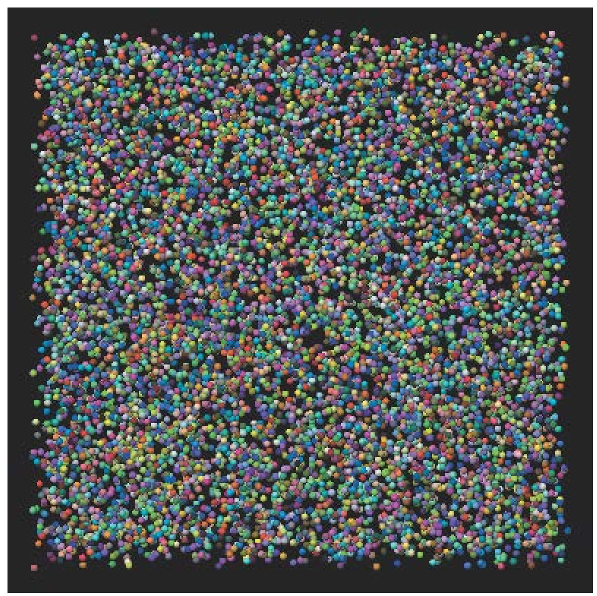
Random spheres distributed in unit cube

- #cells = 8*#objects

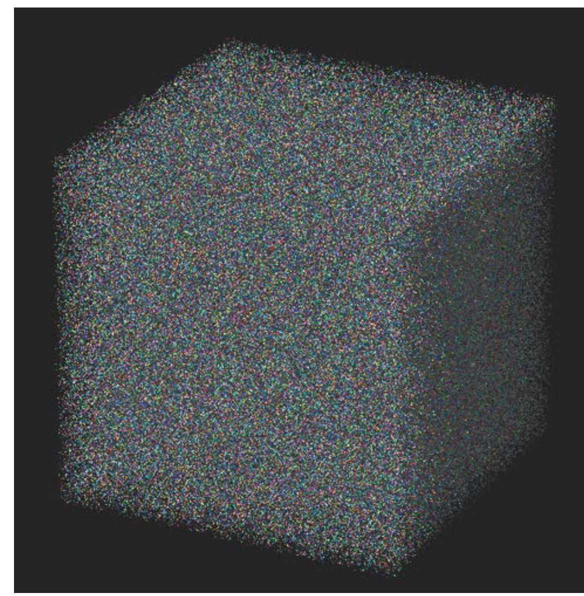
1000



10000



100000



Regular Grid: results

Random spheres distributed in unit cube

- #cells = 8*#objects

#spheres	Grid (sec)	No grid (sec)	Speed-up
10	1.5	2.5	1.6
100	2.0	16	8
1000	2.7	164	61
10000	3.8	2041	537
100000	4.7	22169	4717
1000000	5.2	---	~46.307

Regular Grid: resolution of grid

R cells total (usually a cubed number)

n = number of objects

- On average, n/R objects per cell
- (is this a valid assumption?)

Average number of cells traversed by a ray? $\sqrt[3]{R}$

- (is there a lower bound? upper bound?)

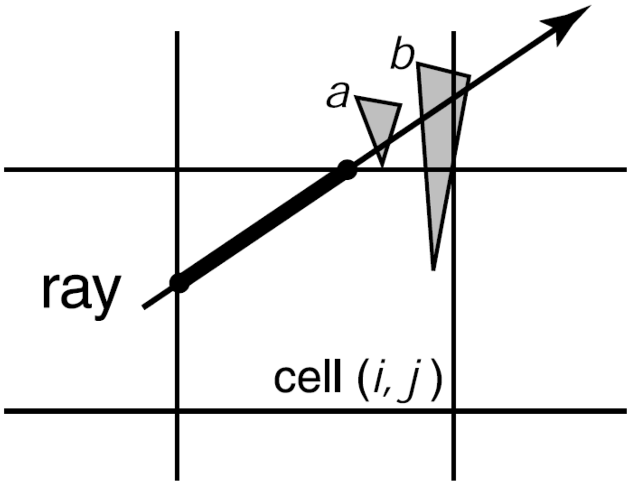
Cost: $C = C_{start-up} + \sqrt[3]{R}(C_{cell-to-cell} + C_{intersect}\mathrm{n/R})$

- Which minimizes to: $R = 2nC_{intersect}/C_{cell-to-cell}$

Regular Grid: issues

Check whether intersection point with object is located in current cell

- If not → possible false intersection results



How to avoid multiple intersection tests?

- Object might belong to different cells
  - (test object each time when we pass through all the cells?)
- Mailbox idea:
  - Store last ray identifier & intersection result in mailbox per object
  - Upon new intersection query, check whether ray identifier is already stored in mailbox

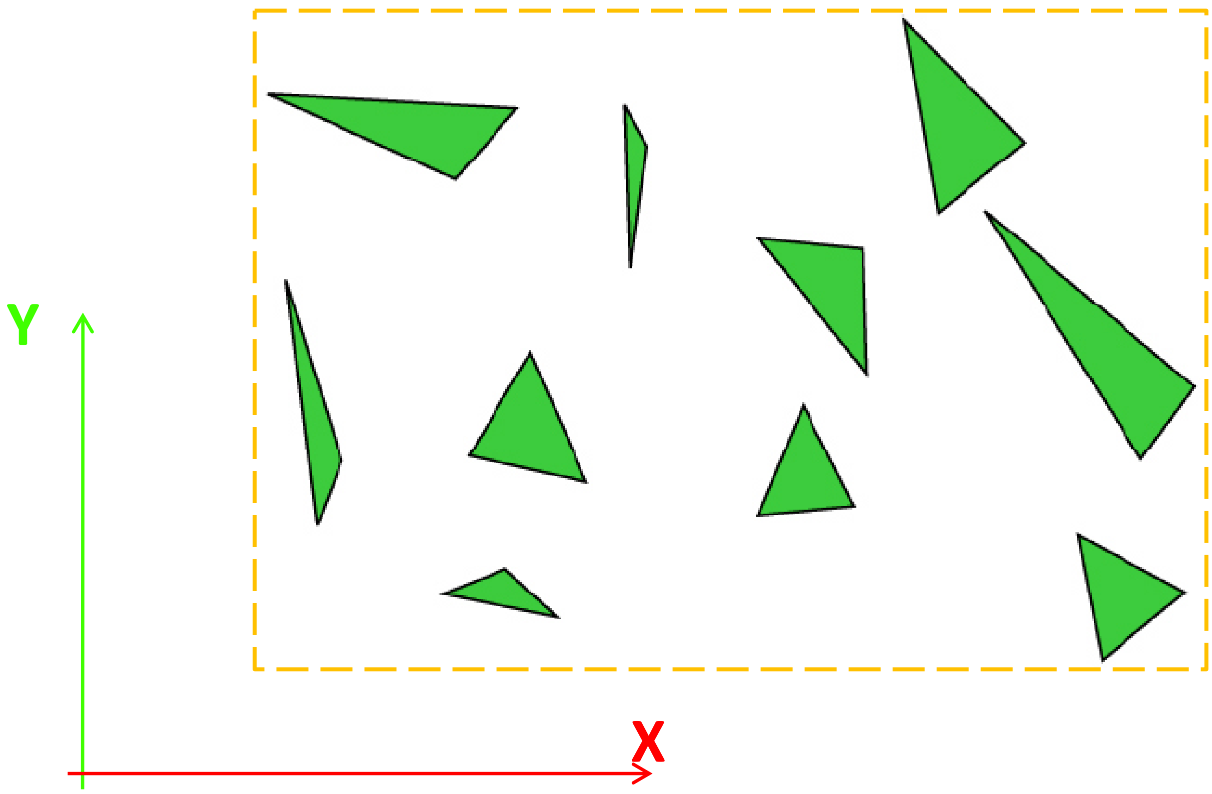
Hierarchical space subdivision

Cells can be subdivided into smaller cells when necessary
- (regular grid within a regular grid)

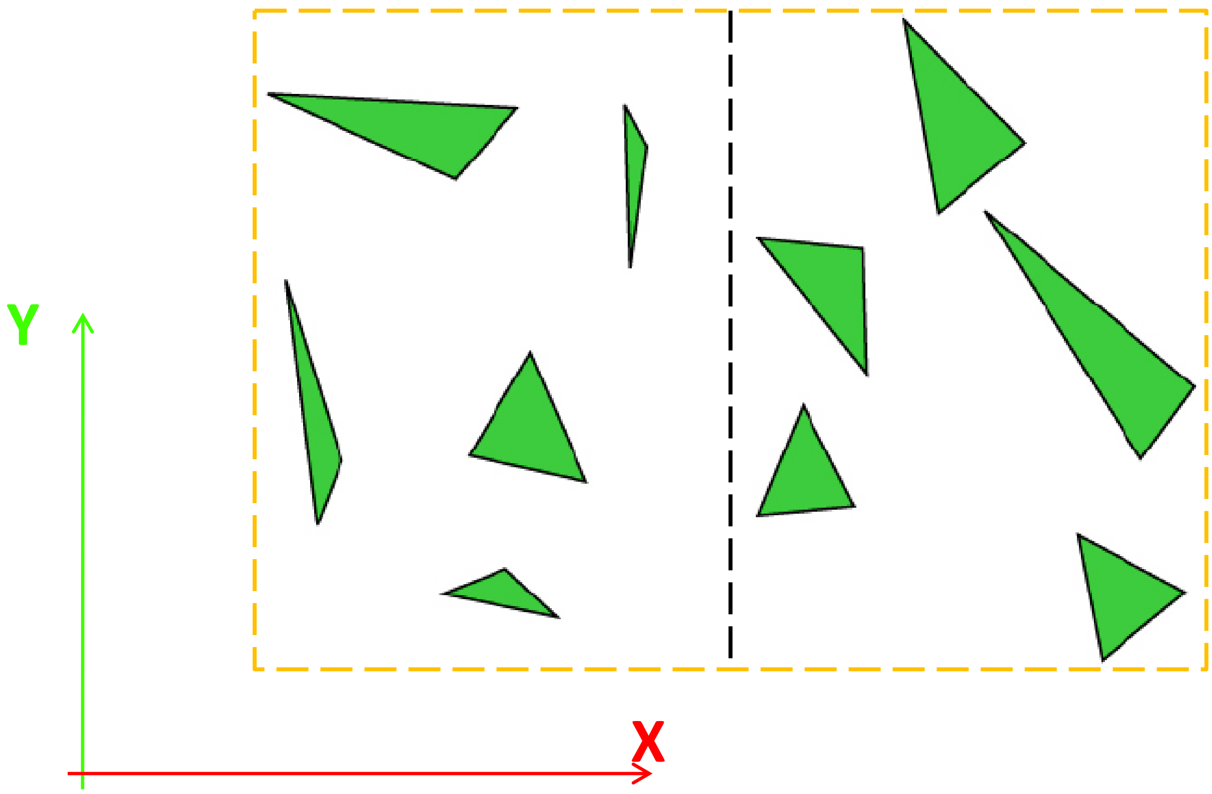
Kd-Trees
- Cells are hierarchically subdivided by axis-aligned planes

Many other data structures are possible:
- BSP-trees, Octrees, Tetrahedralizations, …

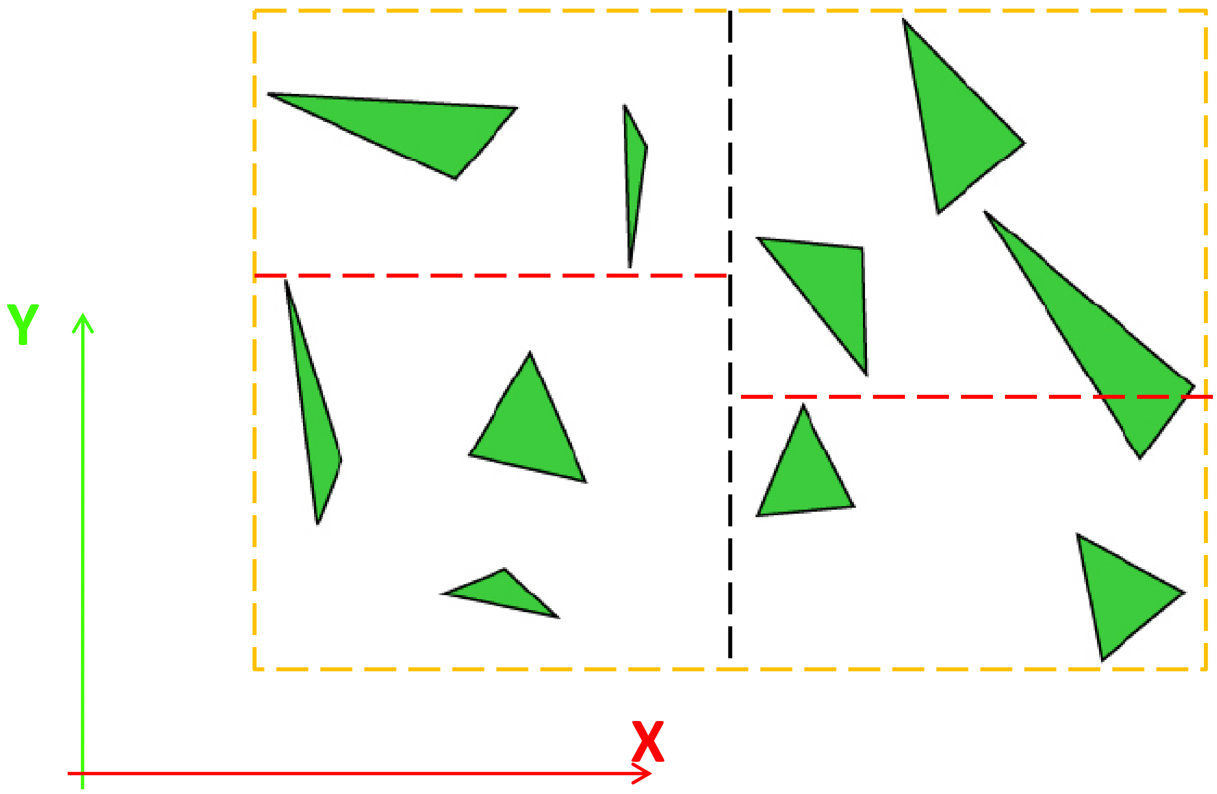
Hierarchical space subdivision



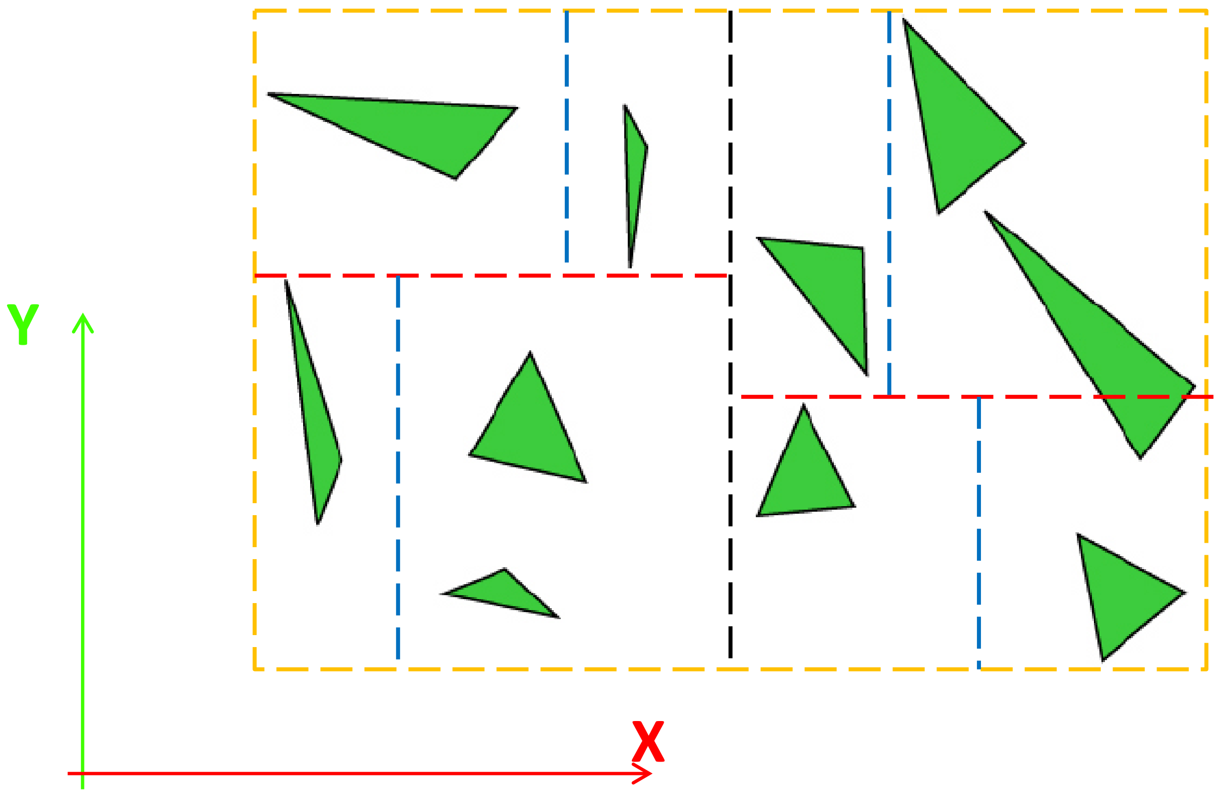
Hierarchical space subdivision



Hierarchical space subdivision



Hierarchical space subdivision



Optimal hierarchy?

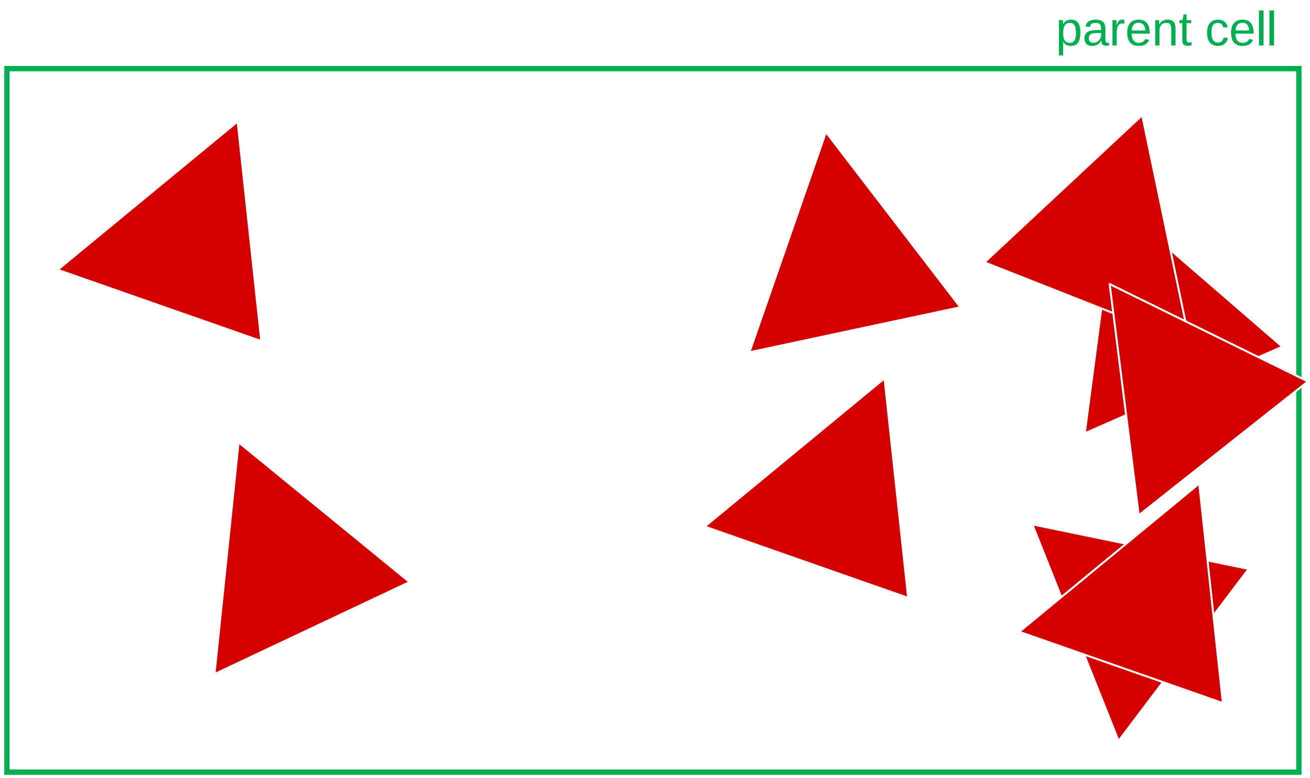
Intuition:

- Equal number of objects in both sub-cells
- (balanced tree)

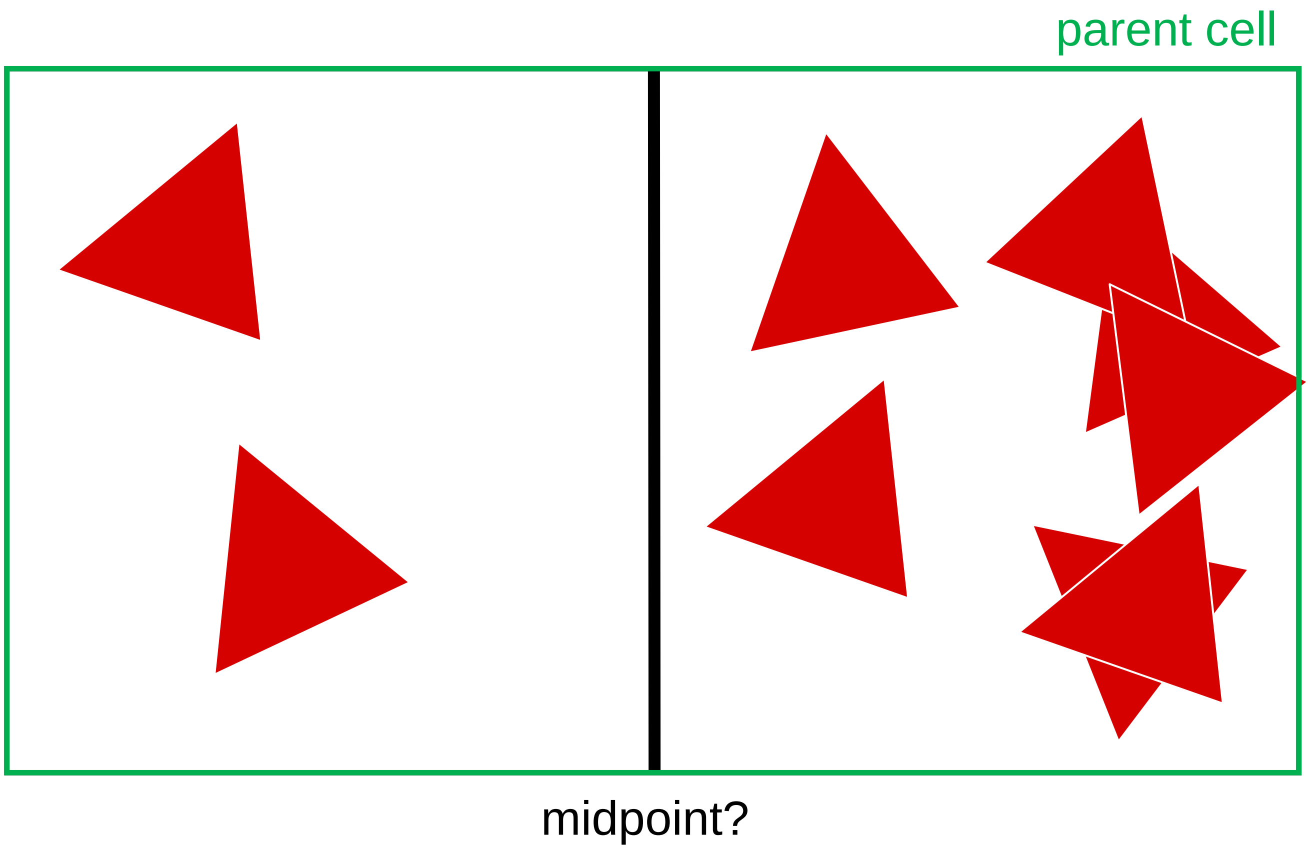
But!

- Objects are not located uniformly in space
- Empty space?
- Majority of objects in a few densely packed cells?
  or
  Majority of objects divided over as many cells as possible?

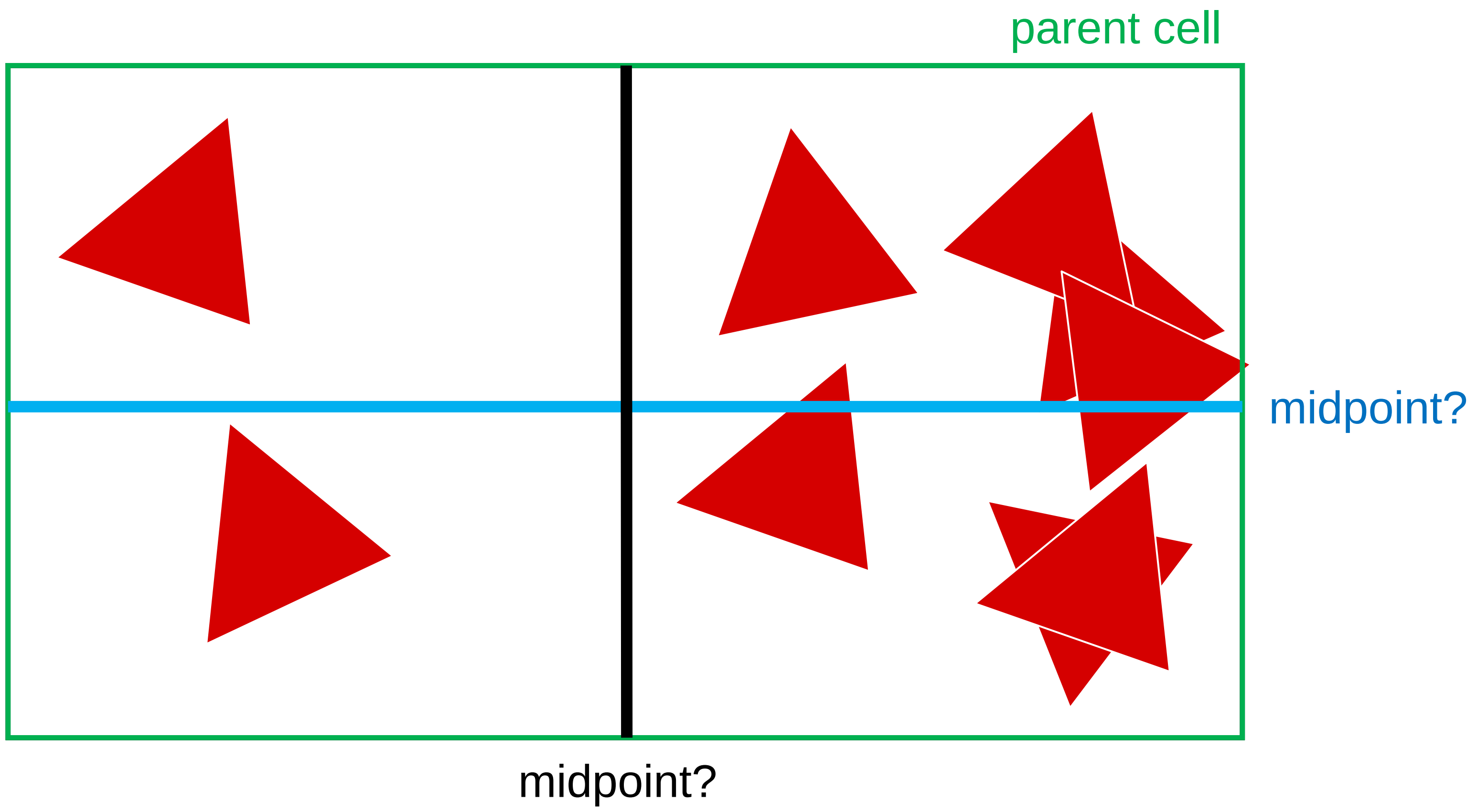
How to choose a subdivision?



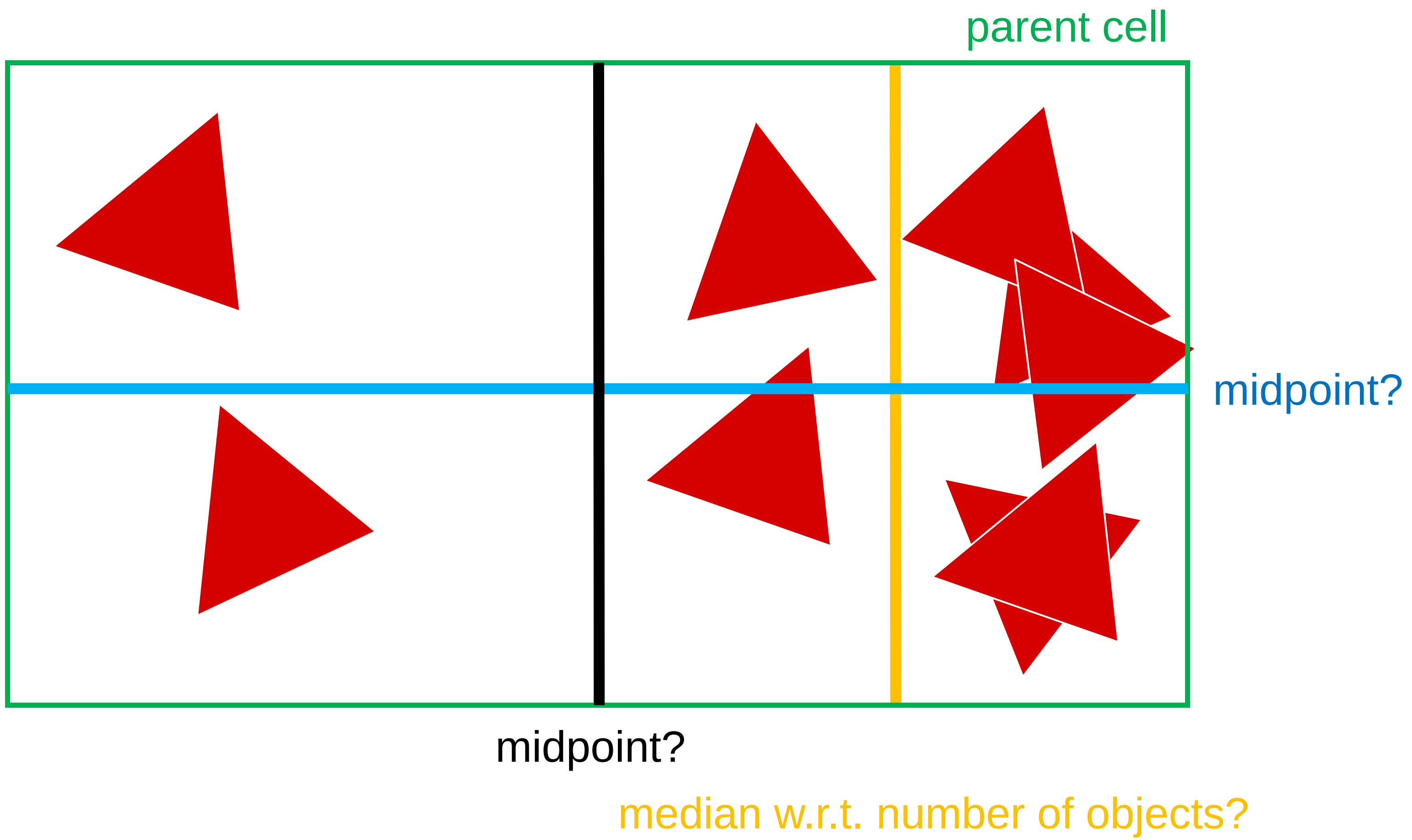
How to choose a subdivision?



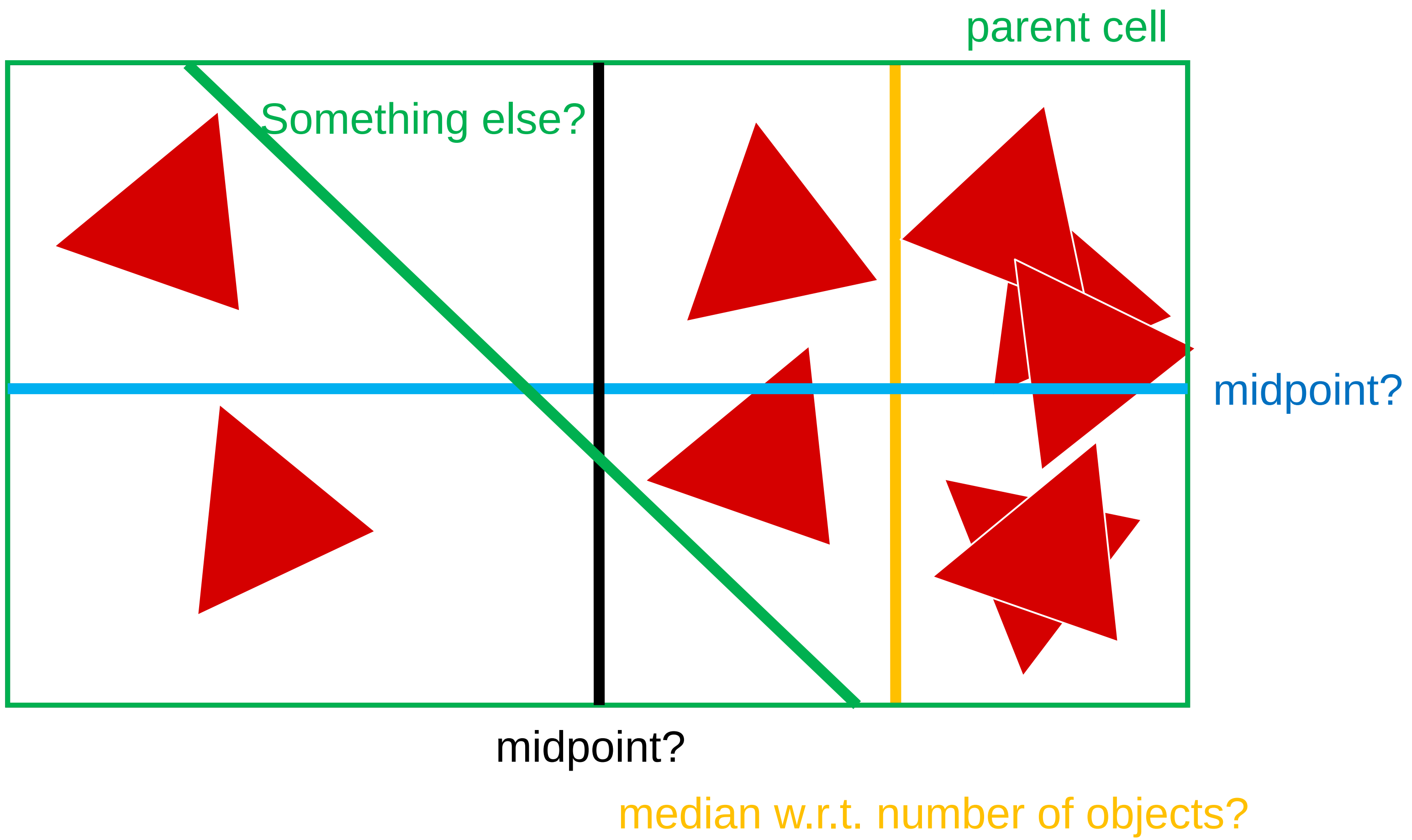
How to choose a subdivision?



How to choose a subdivision?

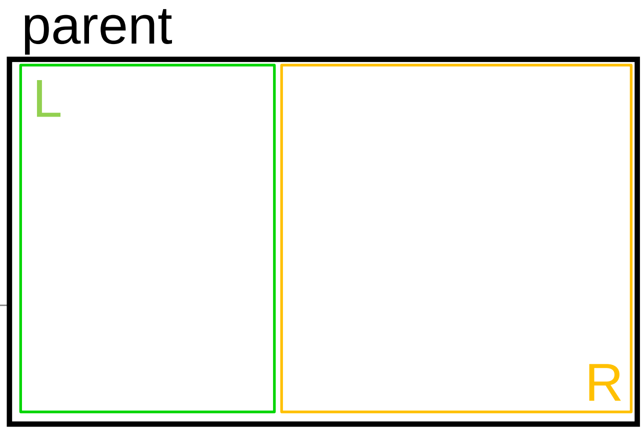


How to choose a subdivision?



Surface Area Heuristic

Cost of tracing a ray through a parent cell:



$$Cost(parent) = C_t + prob(hit\_L).Cost(L) + prob(hit\_R).Cost(R)$$

Fixed cost for traversing the cell
(determining exit point etc.)

Probability to hit left or right cell:
proportional to surface area
of left / right child cell vs. parent cell

Estimate: proportional
to #triangles in cell
(but really recursive …)

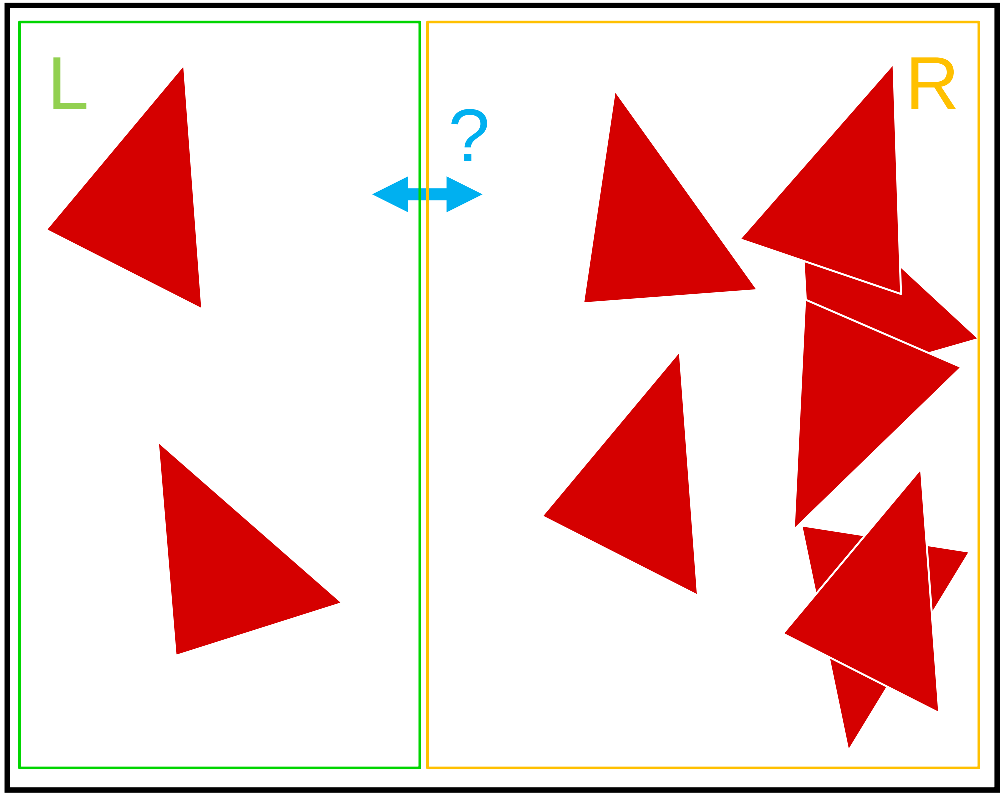
$$Cost(parent) = C_t + \frac{S_L}{S_{parent}}.N_L.C_i + \frac{S_R}{S_{parent}}.N_R.C_i$$

Surface Area Heuristic

$$Cost(parent) = C_t + \frac{S_L}{S_{parent}}.N_L.C_i + \frac{S_R}{S_{parent}}.N_R.C_i$$

$$= C_t + (S_L.N_L + S_R.N_R).\frac{C_i}{S_{parent}}$$

$$= C_t + \underbrace{(S_L.(N_L - N_R) + S_{tot}.N_R)}_{\text{either monotonically increasing or decreasing between objects}}.\frac{C_i}{S_{cell}}$$



$S_L + S_R = S_{tot}$

➔ only test 2 possible plane locations

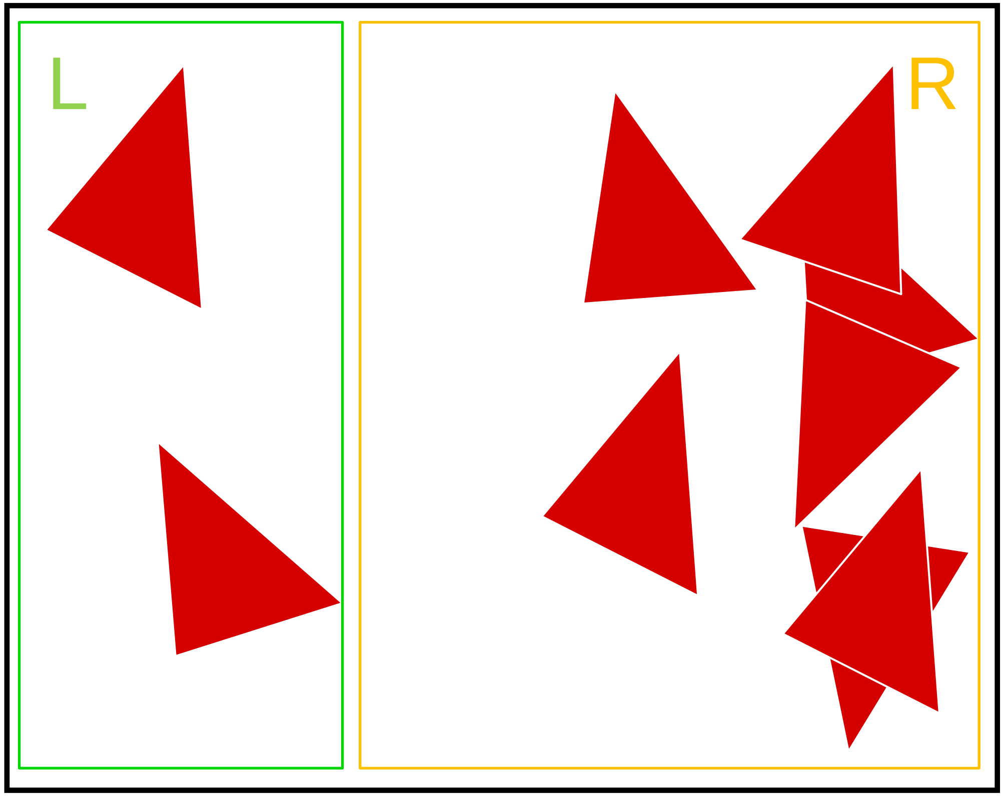
Surface Area Heuristic

$$Cost(parent) = C_t + \frac{S_L}{S_{parent}}.N_L.C_i + \frac{S_R}{S_{parent}}.N_R.C_i$$

$$= C_t + (S_L.N_L + S_R.N_R).\frac{C_i}{S_{parent}}$$

$$= C_t + \underbrace{(S_L.(N_L - N_R) + S_{tot}.N_R)}_{}.\frac{C_i}{S_{cell}}$$

either monotonically increasing or decreasing between objects



$S_L + S_R = S_{tot}$

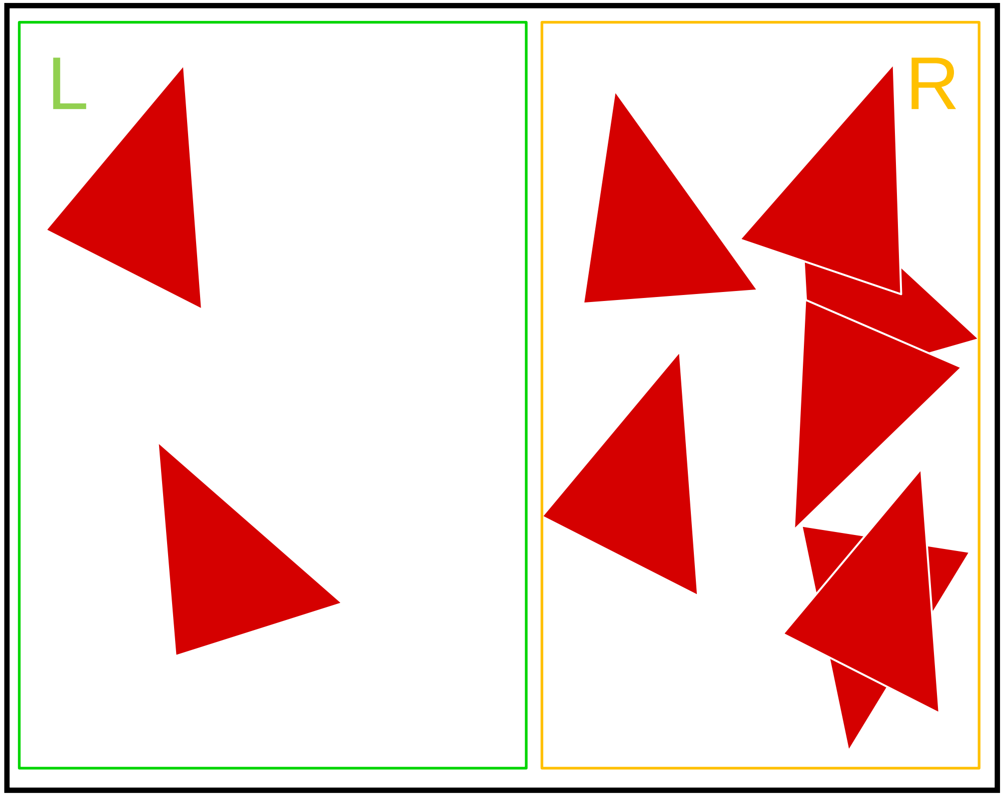
➔ only test 2 possible plane locations

Surface Area Heuristic

$$Cost(parent) = C_t + \frac{S_L}{S_{parent}}.N_L.C_i + \frac{S_R}{S_{parent}}.N_R.C_i$$

$$= C_t + (S_L.N_L + S_R.N_R).\frac{C_i}{S_{parent}}$$

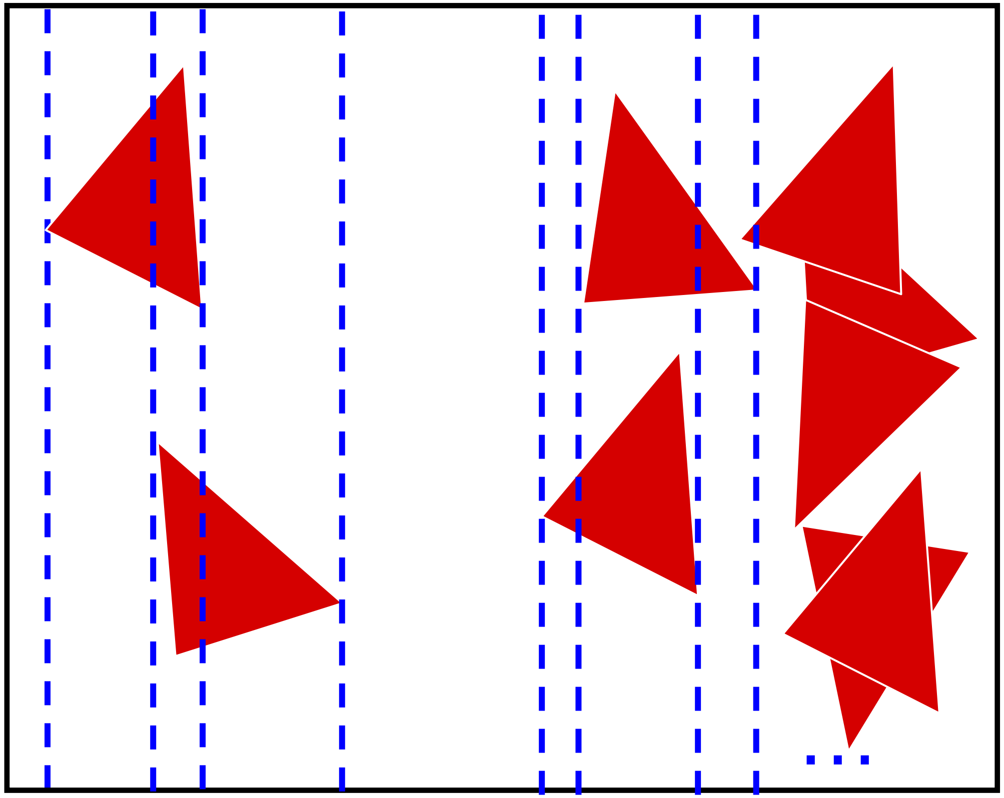
$$= C_t + \underbrace{(S_L.(N_L - N_R) + S_{tot}.N_R)}_{\text{either monotonically increasing or decreasing between objects}}.\frac{C_i}{S_{cell}}$$



$S_L + S_R = S_{tot}$

➔ only test 2 possible plane locations

Surface Area Heuristic



For n objects in cell:

- Test 2n planes for each splitting dimension
- evaluate SAH for each plane location

➔ "optimal" splitting plane is found

In practice:

- Test a limited number of plane locations
- Pick best one

➔ "good enough" splitting plane is found

Cost for each subnode?

$$Cost(parent) = C_t + \frac{S_L}{S_{parent}}.N_L.C_i + \frac{S_R}{S_{parent}}.N_R.C_i$$

Estimate for cost per child-cell:

- proportional to #objects in each cell (greedy algorithm for subdivision)

But:

- Subcells are recursively subdivided as well
  - ➔Our estimated cost is too high
- However, works well in practice

When to stop splitting?

Stop when:

- cost_no_split < cost_split

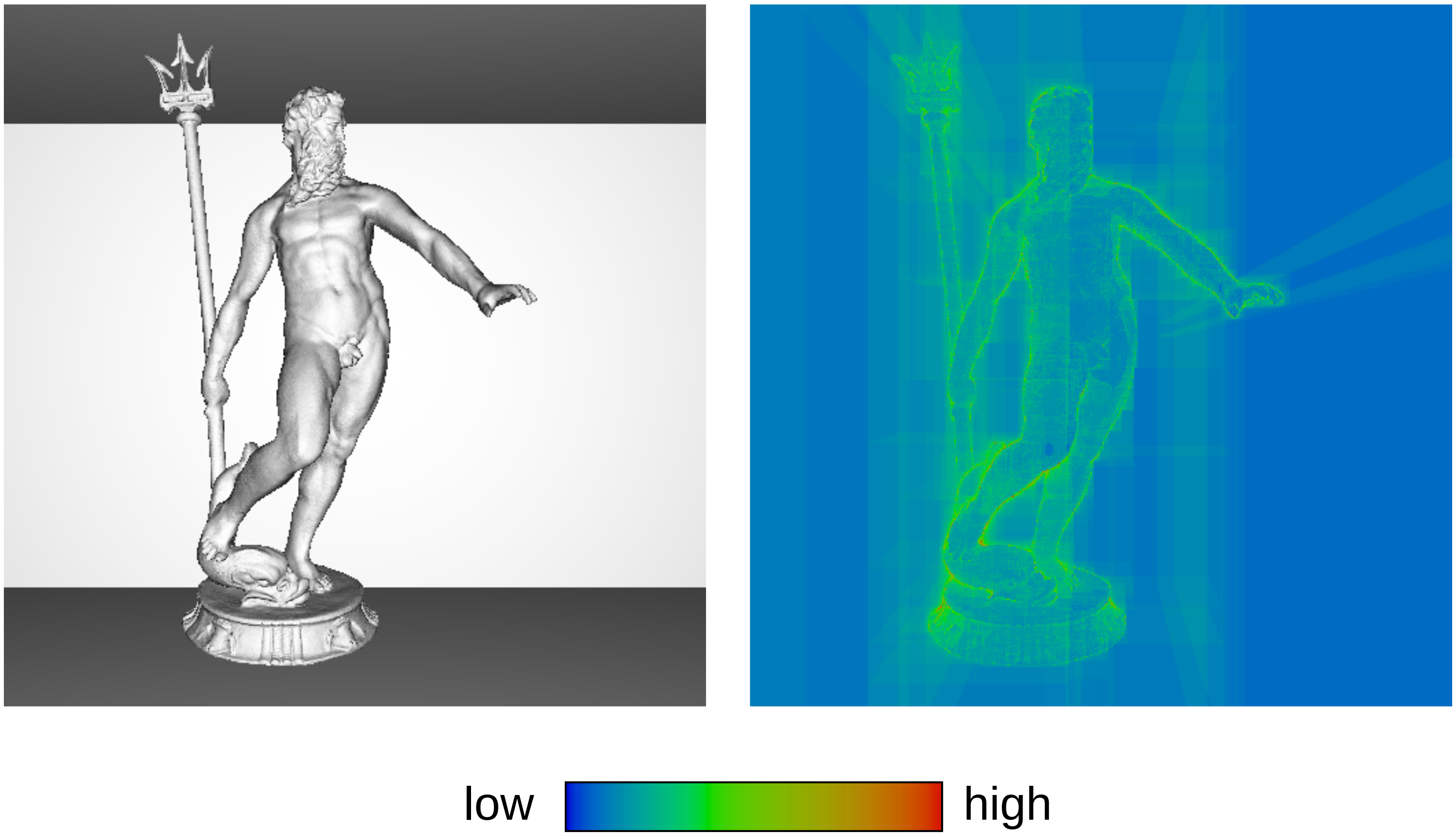
But:

- Sometimes more than 1 subdivision is needed to lower the cost
- Concealed empty space in objects might not be discovered when recursive splitting is stopped too soon.

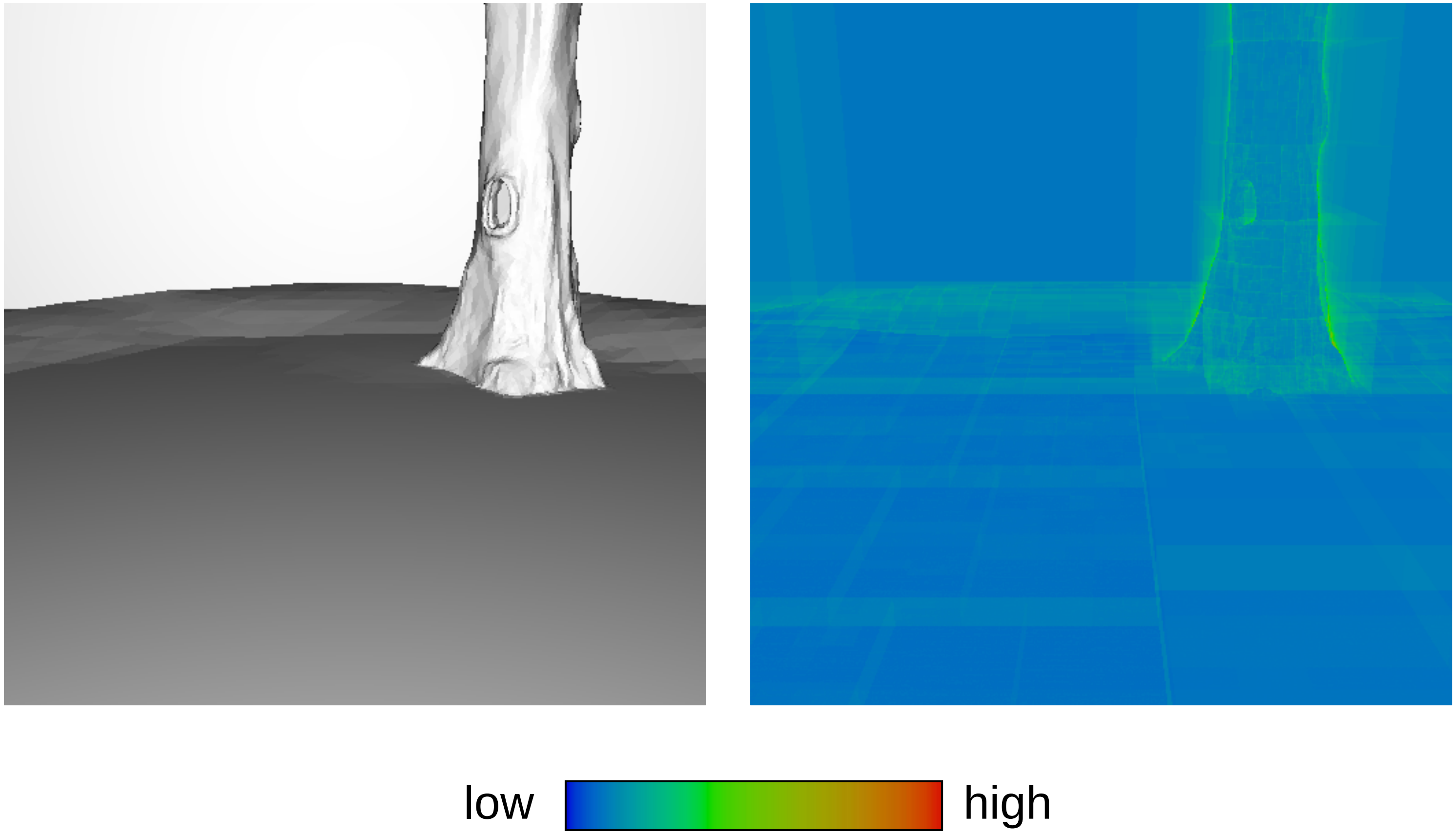
Practice:

- Splits continue till a fixed number of geometric primitives (triangles) is reached … often between 2 and 8.

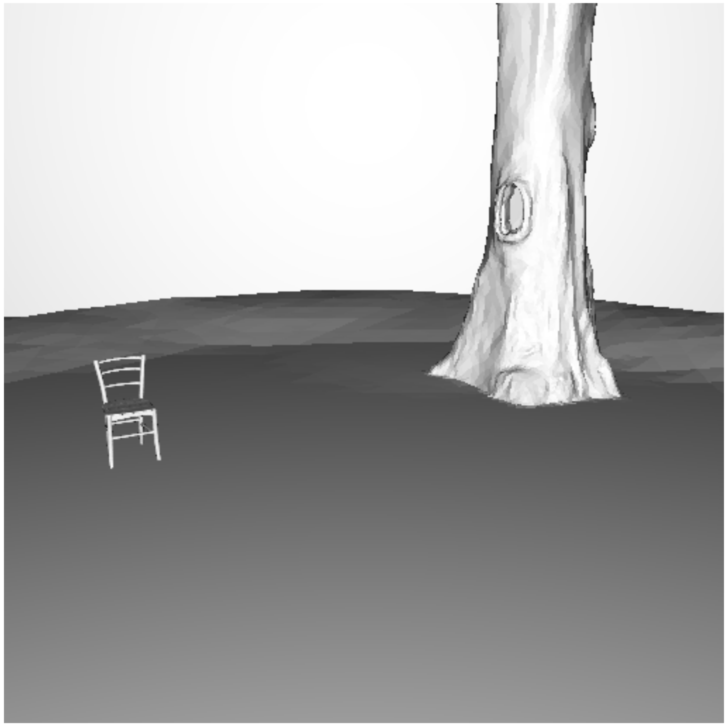
K-d tree results

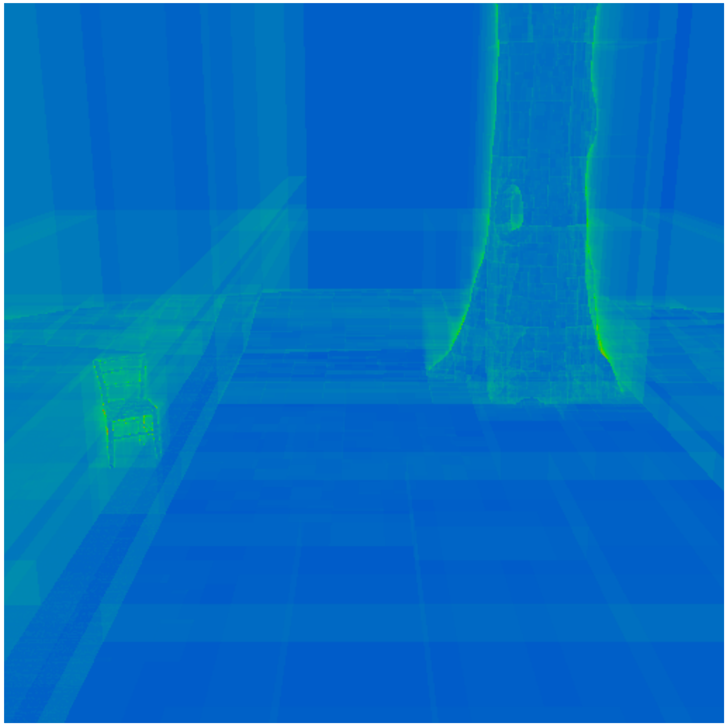


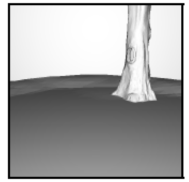
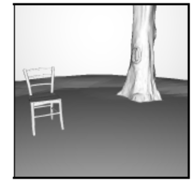
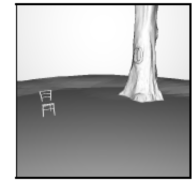
K-d tree results



K-d tree results





	neptune	forest	chair	forest & large chair	forest & small chair	armadillo
						
scene statistics						
# triangles	2.64M	17.52k	408.41k	425.92k	425.92k	345.95k
kd-tree						
# leaf nodes	7.65M	47.27k	1.76M	1.65M	1.52M	452.50k
# n-emp leaf nodes	4.01M	27.51k	992.63k	921.46k	848.34k	203.38k
construction time	42.30 s	0.30 s	10.62 s	10.72 s	10.00 s	2.48 s
avg # triangles / n-emp leaf node	2.56	2.54	2.42	2.49	2.48	2.31
exp # traversal steps	19.98	15.78	17.19	18.47	18.27	19.89
exp # leafs visited	5.57	4.79	5.17	5.20	5.14	5.61
exp # intersections	3.79	3.99	3.82	3.79	3.78	3.76
exp cost	375.60	316.59	334.27	352.72	349.70	373.47
render time	0.37 s	0.23 s	0.29 s	0.24 s	0.22 s	0.28 s
avg # nodes / ray	45.17	38.94	43.78	32.58	30.86	46.22
avg # intersections / ray	2.24	2.41	2.19	2.43	2.40	2.20

Acceleration Structures: conclusions

Think of an acceleration structure as a search structure

- Complexity: linear ➔ logarithmic

Many hybrid formats are possible

- E.g. hierarchical regular grids
- E.g. bounding volumes inside grids
- E.g. hierarchical grids, then build BV for the content of each grid cell
- …

Memory access and synchronization if often an important optimization issue as well