BIH

(BOUNDING INTERVAL HIERARCHY)

PROJECT
Beschleunigungsstrukturen für echtzeitfähiges
Ray-Tracing auf aktueller
Hardware-Infrastruktur

Henning Gründl

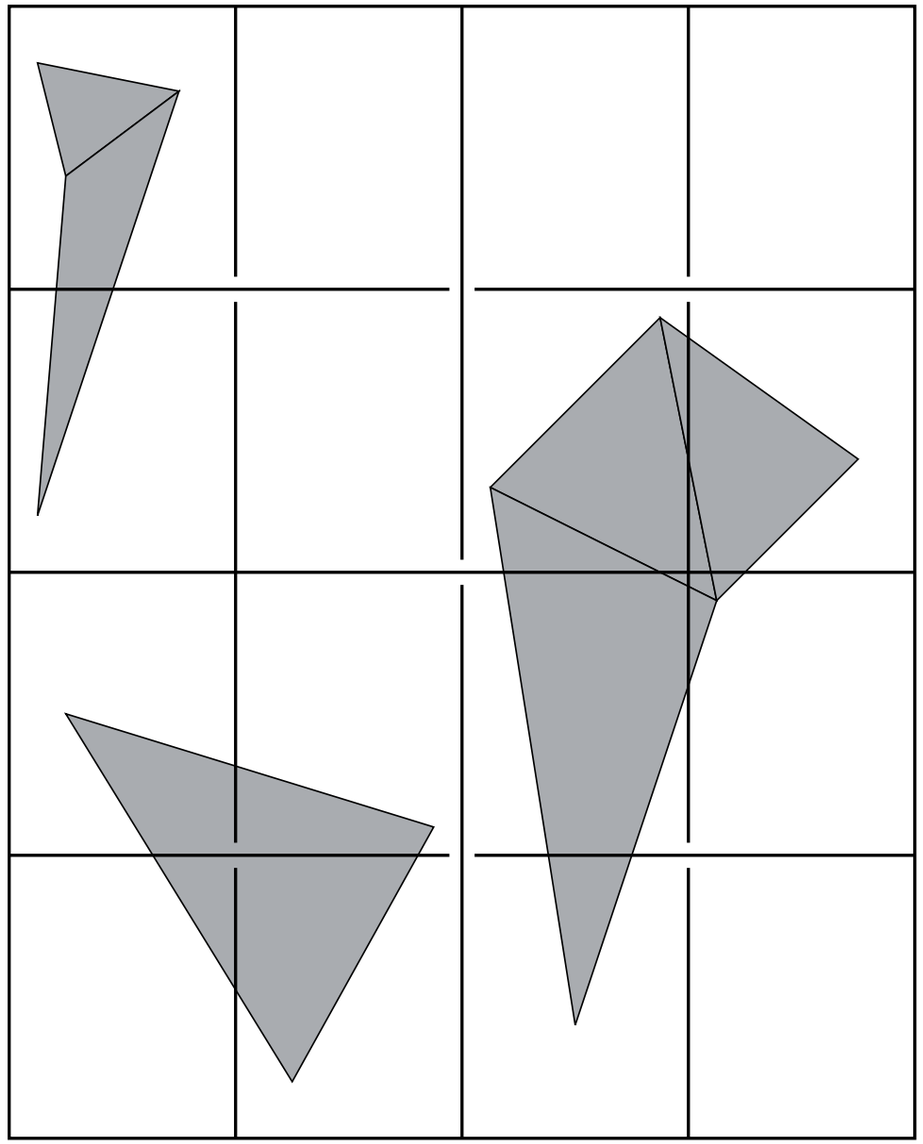
Bauhaus-Universität Weimar

ACCELERATION STRUCTURES

- **Partitioning Space**
- disjunct volume elements
- front-to-back traversal
- multiple object references
- unknown a priori memory footprint
- Binary Space Partition

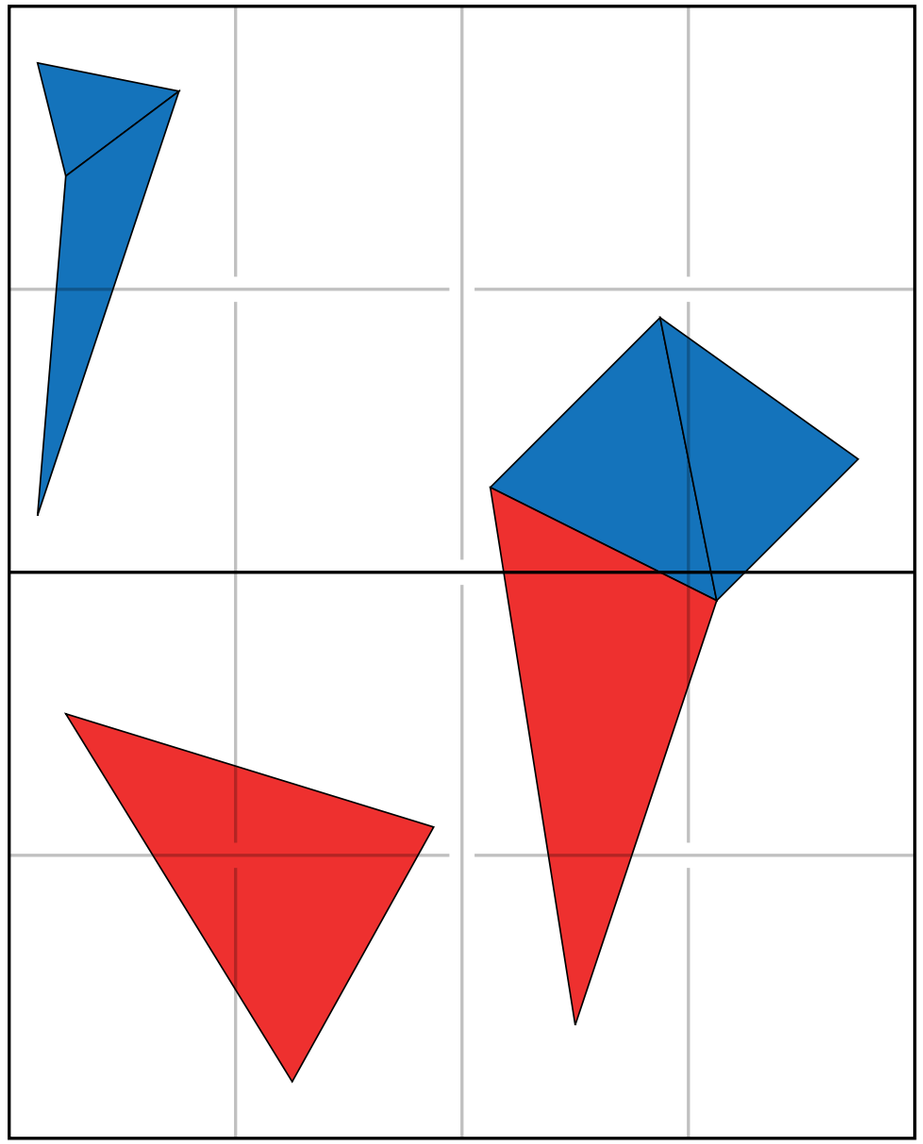
- **Partitioning Objects**
- objects referenced once
- memory requirements can be bounded a priori
- volume elements were not ordered
- do overlap
- Bounding Volume Hierarchy

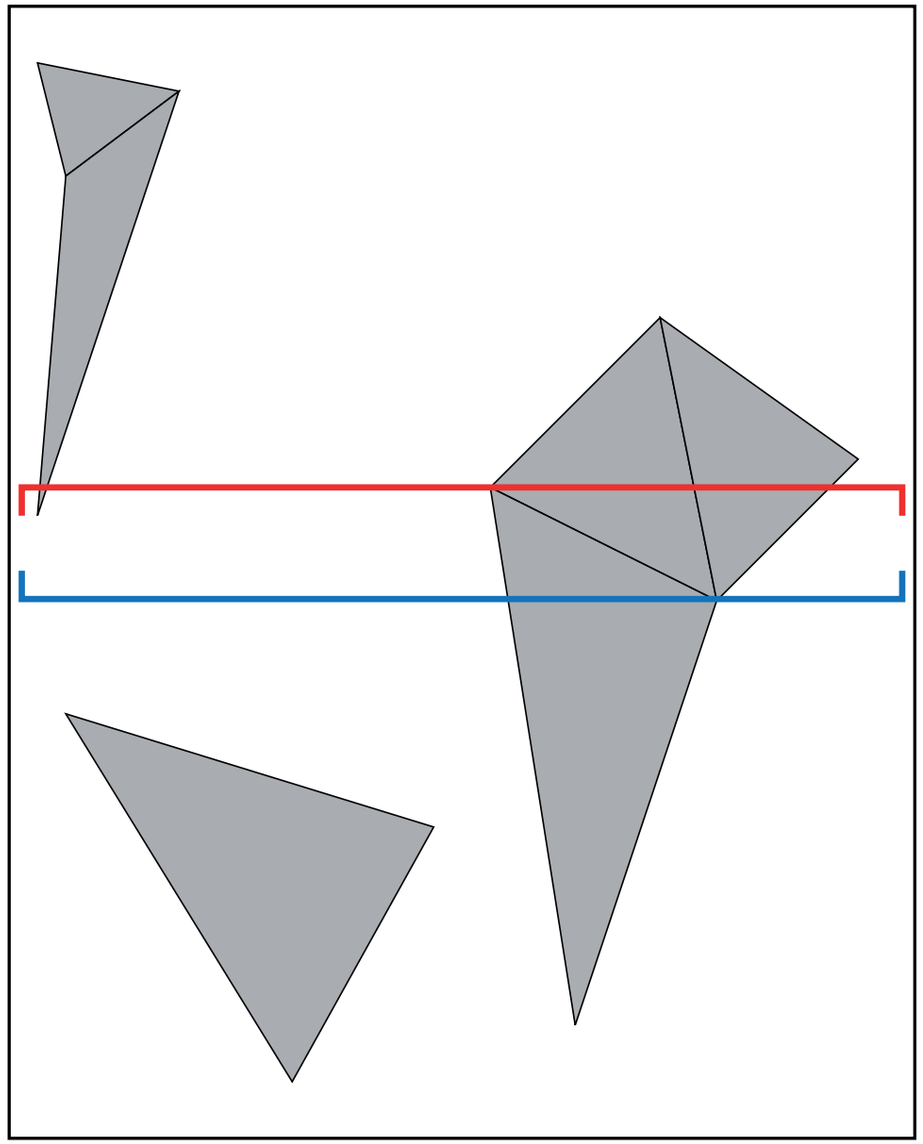
CONSTRUCTION



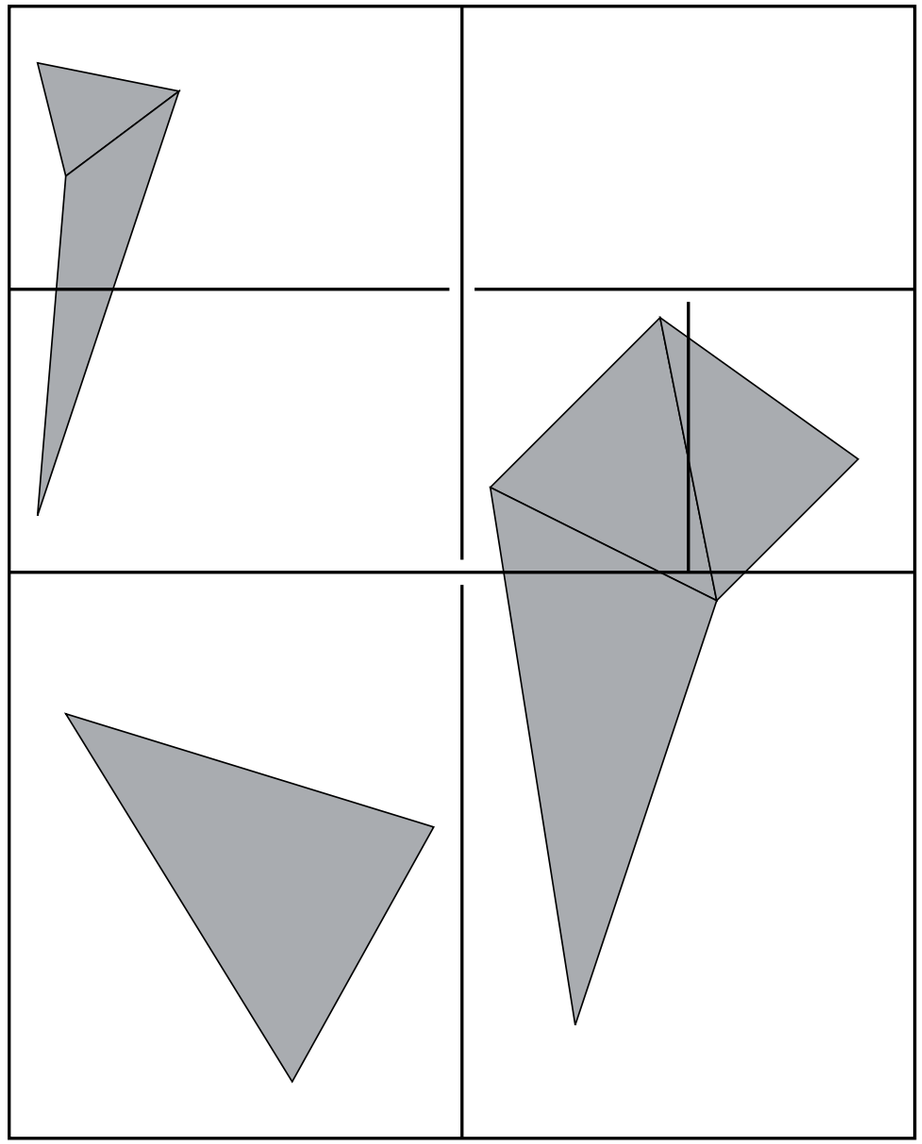
- cross-over of **partitioning object lists** and **spatial partitioning**
- not SAH-based
- hierarchically subdividing scene AABB
- candidate planes
- divde longest side in the middle until one object is left

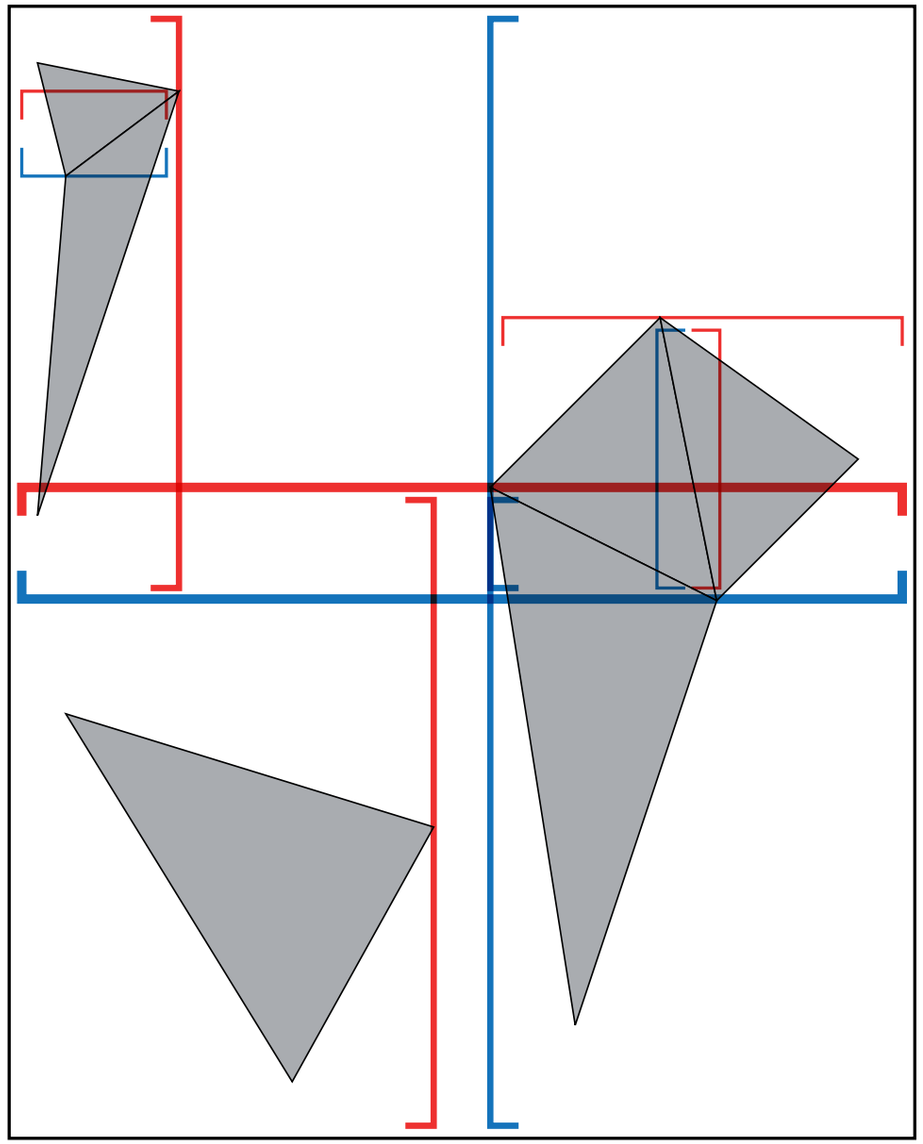
CONSTRUCTION



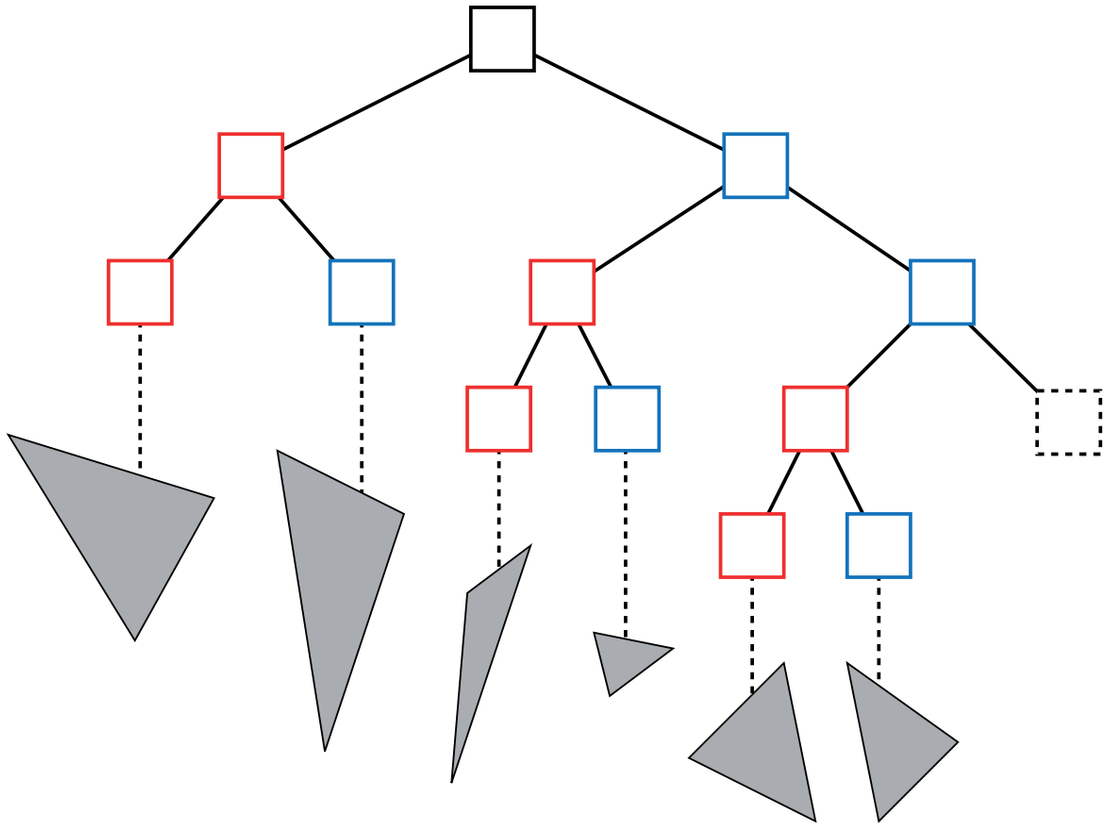


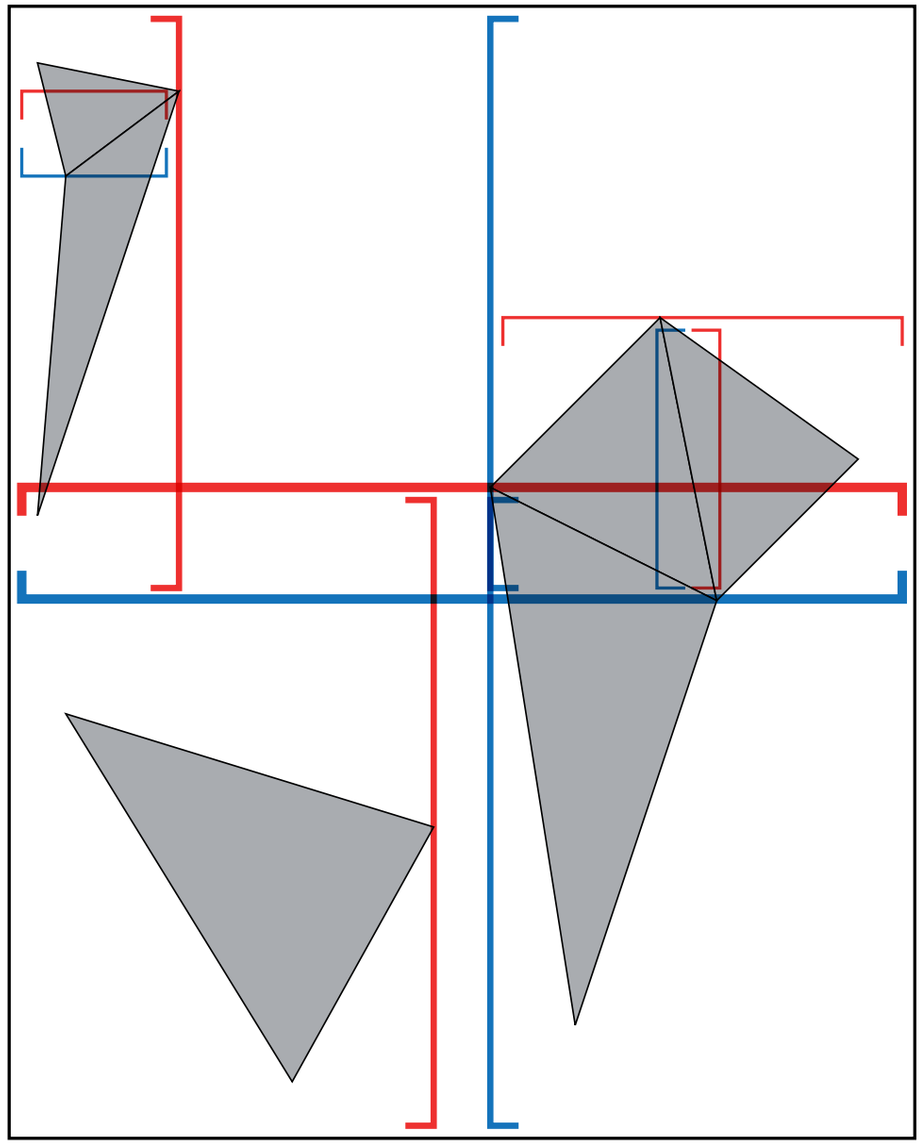
CONSTRUCTION





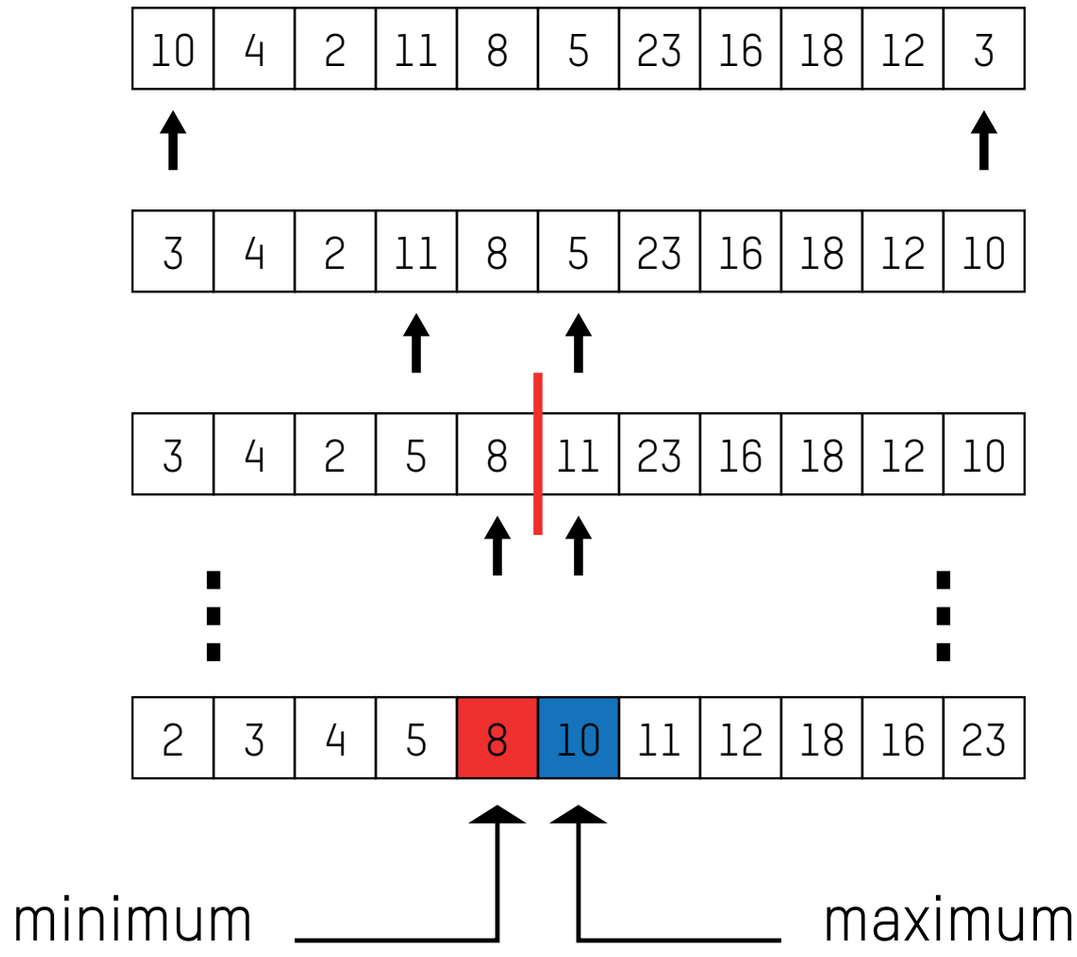
CONSTRUCTION





CONSTRUCTION

- object sorting in place
- similar to quicksort
- **O(**n log n**)**
- candidate plane on the x-axis
- width of the scene BB in x-direction is 18, candidate plane at **9**

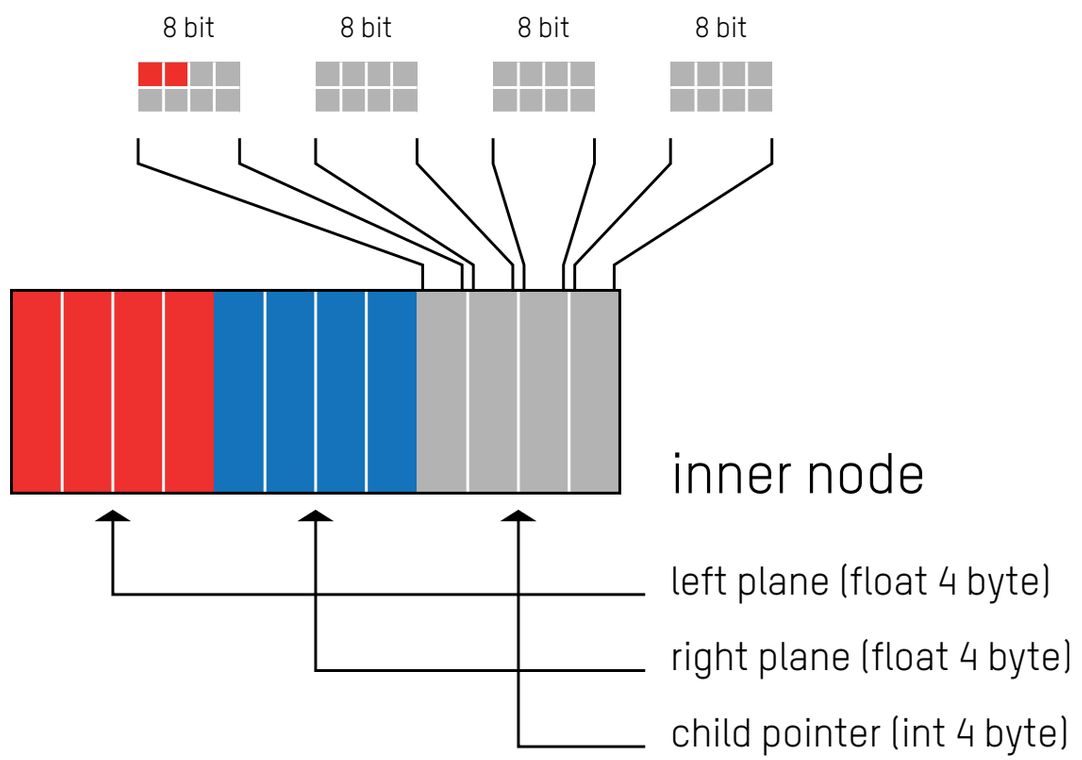


DATA STRUCTURE

```
typedef struct
{
    unsigned  int index ;           // x: 00, y: 01, z: 10, leaf: 11
    union
    {
        unsigned  int items[2] ;    // leaf only
        float clip[2];              // internal node only
    } ;
} BIH_Node ;
```

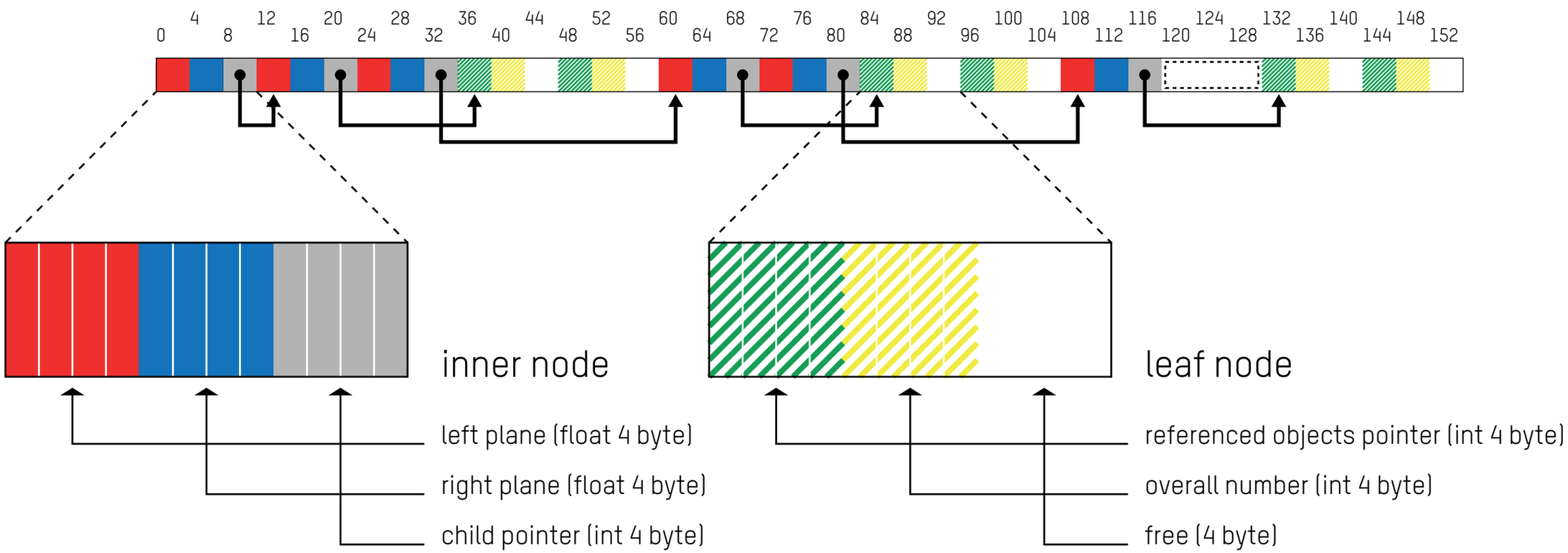
DATA STRUCTURE

- 2^{32} ... 2^4 2^3 2^2 2^1 2^0
- use the lower two bits of the children-pointer
- indicate the axis (00: x, 01: y, 10: z)
- leaf (11)

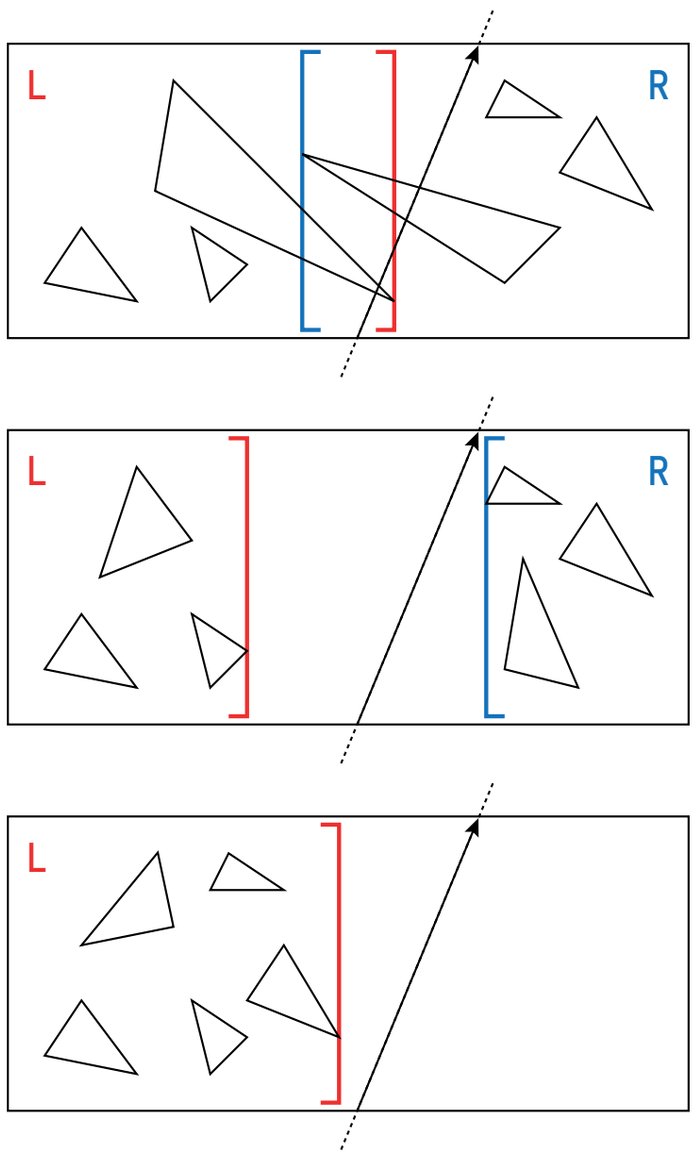


DATA STRUCTURE

- serialized bih-tree layout
- four-byte-boundaries

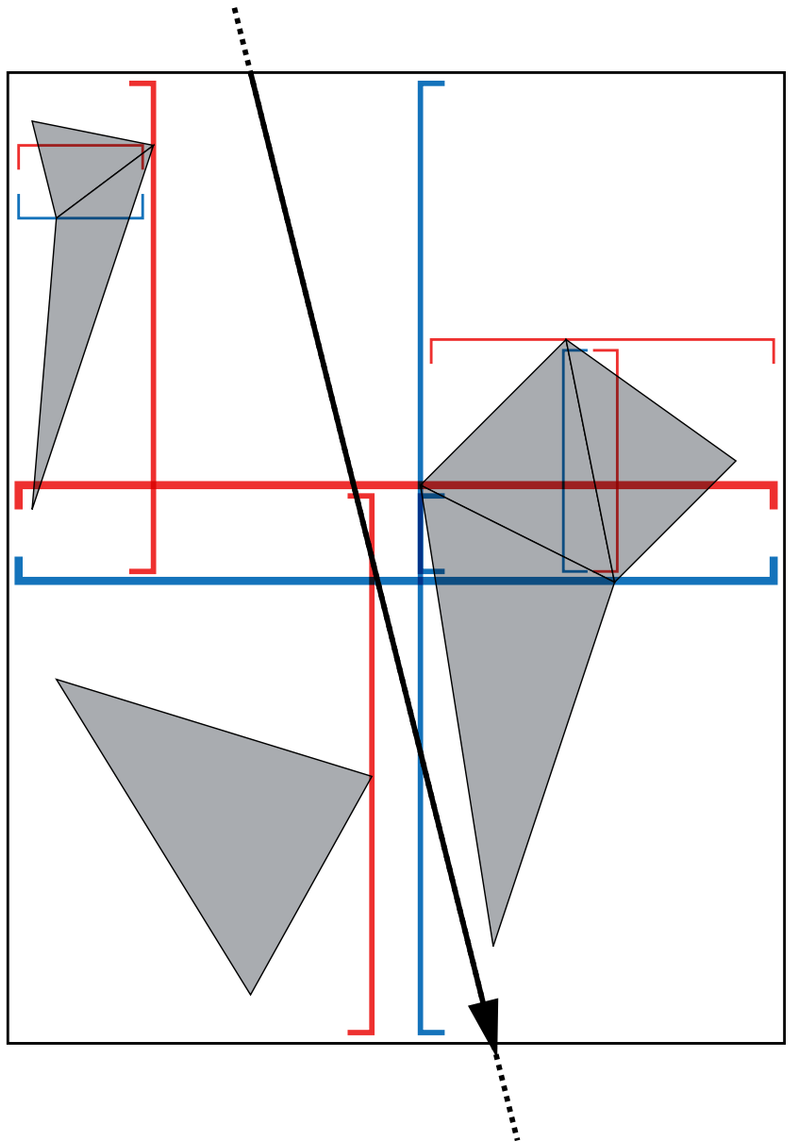


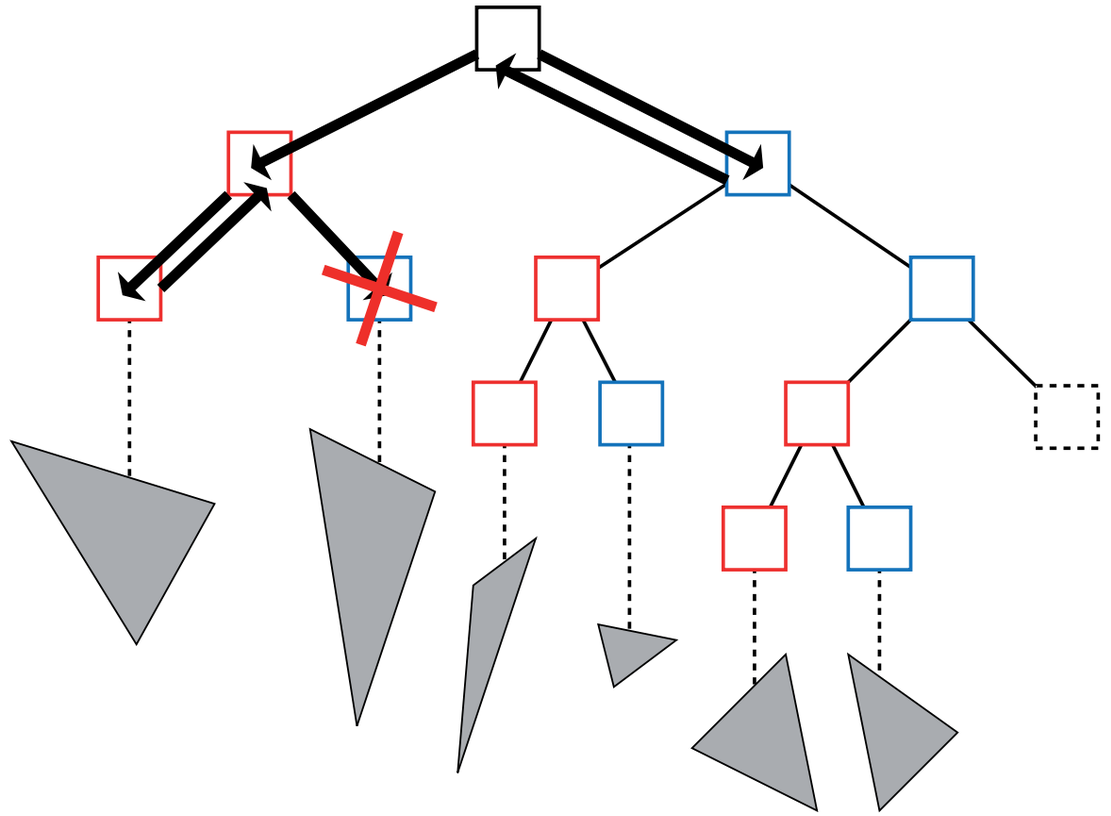
RAY TRAVERSAL



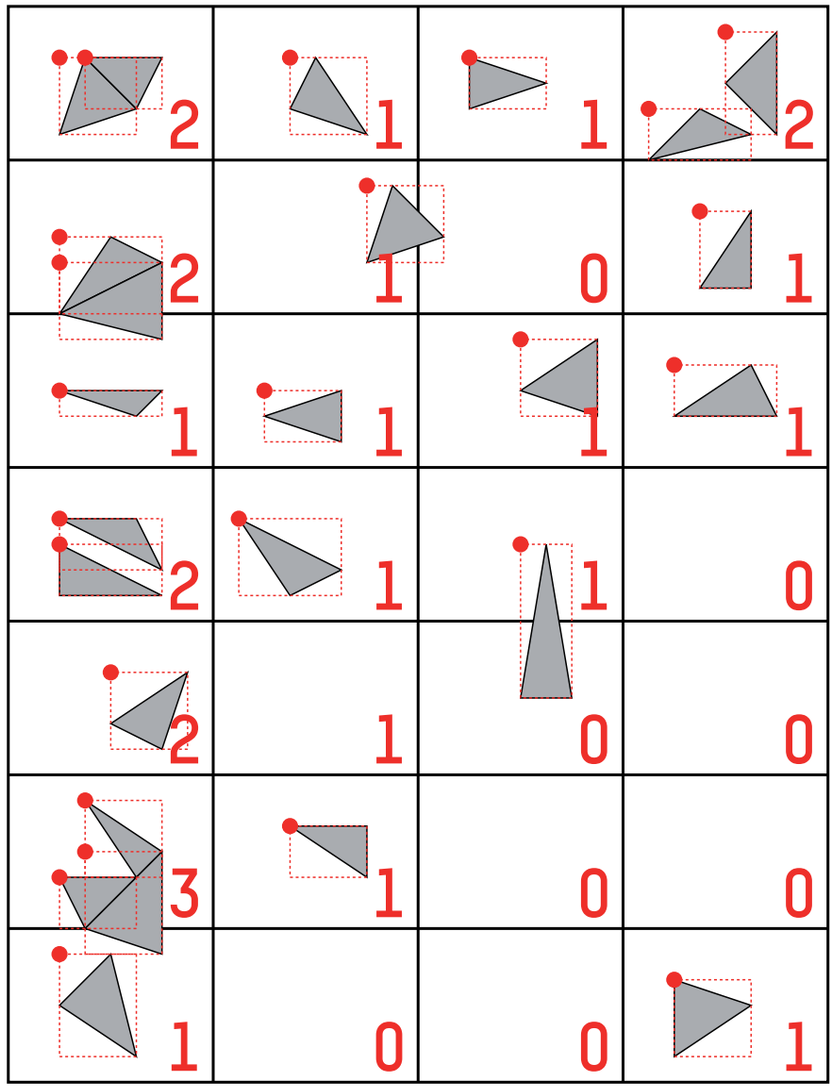
- children are spatially ordered
- directly access the child that is closer to the ray origin by the sign of the ray direction
- almost identical to that of a kd-tree
- the volume elements of a bounding interval hierarchy can overlap

RAY TRAVERSAL





APPROXIMATE SORTING



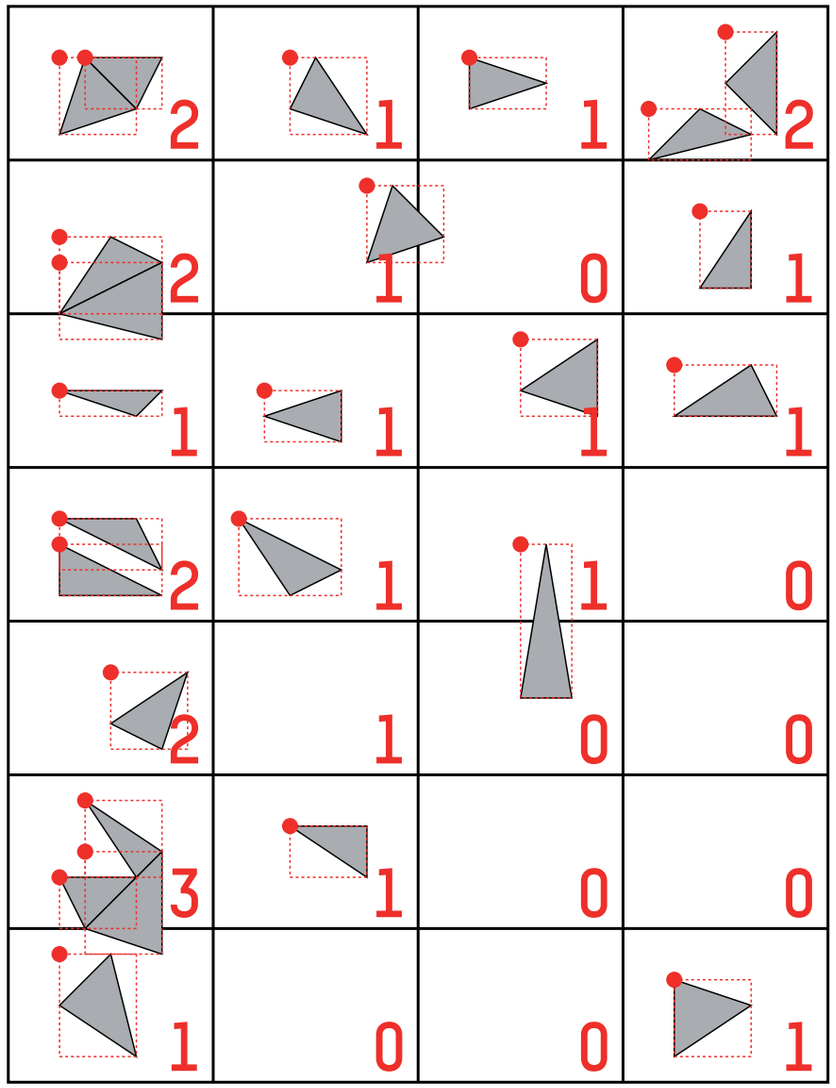
- bucket sorting preprocess

$$\text{GRID RESOLUTION} = \frac{\text{SCENE BOUNDING BOX}}{\text{AVERAGE OBJECT SIZE}}$$

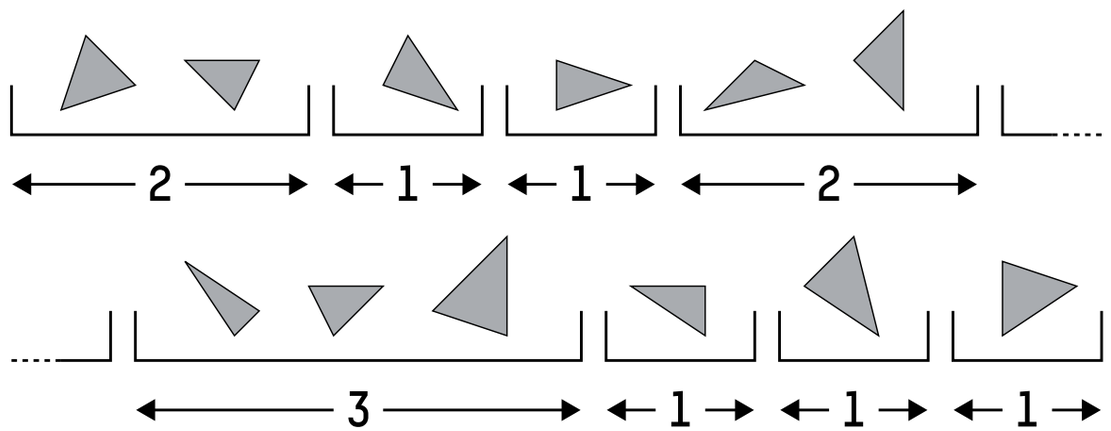
- uniform distribution highly unlikely, results in scaling factor
- initialize counter to zero
- for every object increment the counter

$$\sum \text{counter} = \text{number of objects}$$

APPROXIMATE SORTING



- global object index array is allocated
- using same point of every object sort it into the buckets
- for each bucket we compute the BB
- sorting the BB speeds up construction



COMPARISON

› bounding interval hierarchy with/without using the bucket sort preprocess
(640x480, Pentium 4HT 2.8 GHz)

Stanford Buddha	BIH + Bucket	BIH
Triangles	1,087,716	dto.
fps	94%	100%
Construction (msec)	765	1,703

Stanford Dragon	BIH + Bucket	BIH
Triangles	871,414	dto.
fps	93%	100%
Construction (msec)	657	1,390

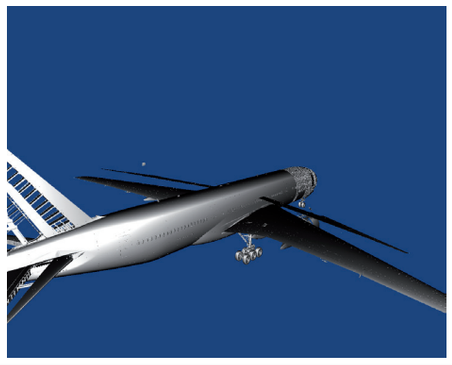
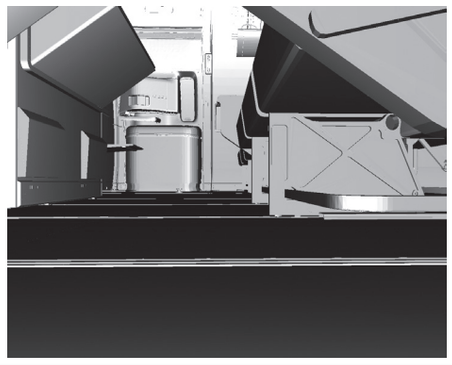
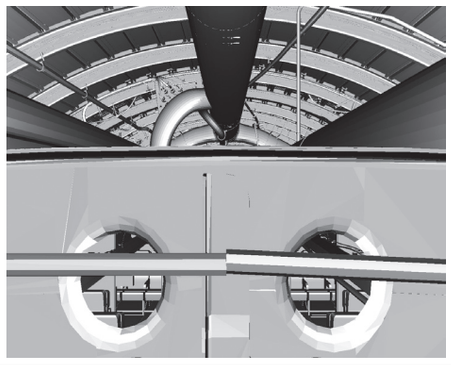
Stanford Thai Statue	BIH + Bucket	BIH
Triangles	10,000,000	dto.
fps	94%	100%
Construction (msec)	7,812	17,484

CONSTRUCTION ON DEMAND

- „constructing only, where rays traverse“
- object sorting is done in place, only a flag is required to mark volume elements that have not yet been subdivided
- subdivision routine is called upon traversal of a ray if the flag is set
- simple optimization is to subdivide a node completely, if all objects contained in it fit into the cache
- on demand construction removes the classic separation of traversal and construction routines

COMPARISON

› Total rendering times including on demand tree construction for the Boeing 777 data set (349,569,456 triangles amounting 12,584,500,416 bytes)
(1280x1024 pixels, single core of an Opteron 875 2.2 GHz 32GB)

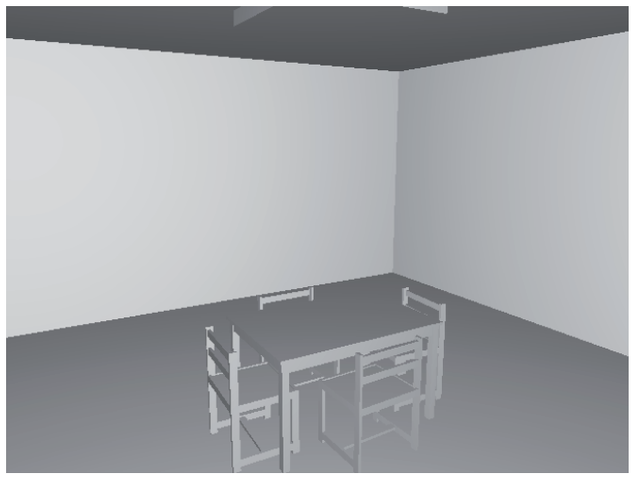
Boeing	View 1	View 2	View 3	View 4
Acc. Data	326,447,848	12,748,120	15,471,692	259,261,404
fps	0.26	0.13	0.13	0.38
Total	8 min	133 sec	153 sec	270 sec
				

CONCLUSION

„The bounding interval hierarchy is an object partitioning scheme that benefits from the efficient traversal techniques of spatial partitioning schemes.“ **[C. Wächter, A. Keller]**

- construction matches split planes to object BB (number of inner nodes six times the number of objects)
- object references exactly matches the number of objects in the scene
- uses bounding box information and minimum/maximum operations in the canonical coordinate system » numerically robust
- does not need a mailbox unit

COMPARISON



- the new technique and state-of-the-art kd-tree implementation, using a very simple shader and 2x2 (SSE accelerated) ray bundles
  (640x480 pixels, measured on a Pentium 4HT 2.8 GHz)

Shirley Scene 6	InView	kd	BIH	on demand
Triangles	1,380	804	dto.	dto.
Triangle memory	66,240	28,944	dto.	dto.
Acc. Data memory	115,312	55,188	12,828	11,972
fps	5.02	11.17	11.99	n.a.
Time to image (msec)	199	89	83	87

COMPARISON

› the new technique and state-of-the-art kd-tree implementation, using a very simple shader and 2x2 (SSE accelerated) ray bundles
(640x480 pixels, measured on a Pentium 4HT 2.8 GHz)

Stanford Dragon	InView	kd	BIH	on demand
Triangles	863,334	871,414	dto.	dto.
Triangle memory	41,440,032	31,370,904	dto.	dto.
Acc. Data memory	26,207,404	24,014,264	13,466,176	5,175,936
fps	2.49	5.92	5.98	n.a.
Time to image (msec)	44,500	3,106	1,557	1,102

Thank you.