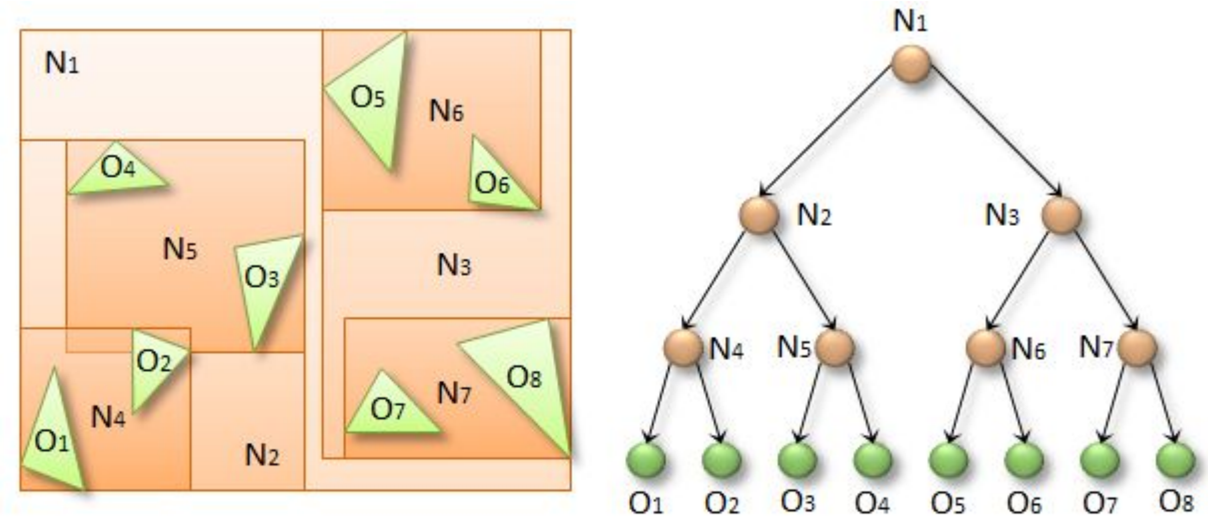
Fast Parallel Construction of High-Quality Bounding Volume Hierarchies

By Tero Karras and Timo Aila (NVIDIA, 2013)

Intro to Bounding volume hierarchies (BVH)



Building a BVH

Two important characteristics

- Build time, usually measured in seconds per build
- Efficiency, usually measured in rays per second

Surface area heuristic (SAH)

$$C_i \sum_{n \in I} \frac{A(n)}{A(\text{root})} + C_l \sum_{l \in L} \frac{A(l)}{A(\text{root})} + C_t \sum_{l \in L} \frac{A(l)}{A(\text{root})} N(l), \quad (1)$$

- SAH cost = expected cost of tracing a non-terminating ray through the scene
- C_i = 1.2, C_l = 0 and C_t = 1

Other methods for BVH construction

Realtime:

- LBVH
- HLBVH

Offline

- SweepSAH
- SBVH

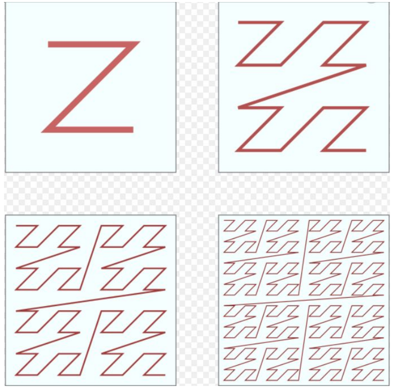
Linear BVH (LBVH)

- Sorts triangles along a space filling curve
- Partition them recursively so that each node represents a linear range of triangles

Very fast construction!

Not very high performance! (Around 50% of SBVH)

HLBVH: Better performance, but still not great

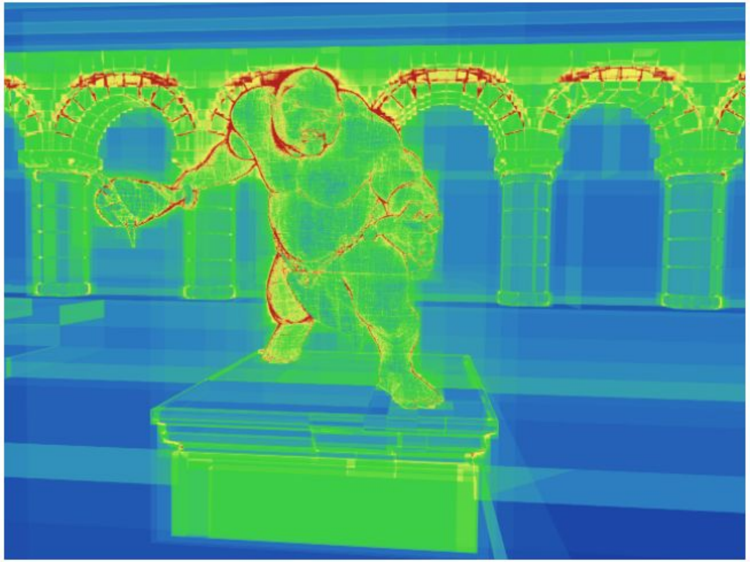


SBVH

- Top down approach
- Greedily selects splitting planes based on SAH cost
- Creates AABBs for each side of the plane
- Duplicates triangles that fall on both sides of the plane
- Incorporates triangle splitting

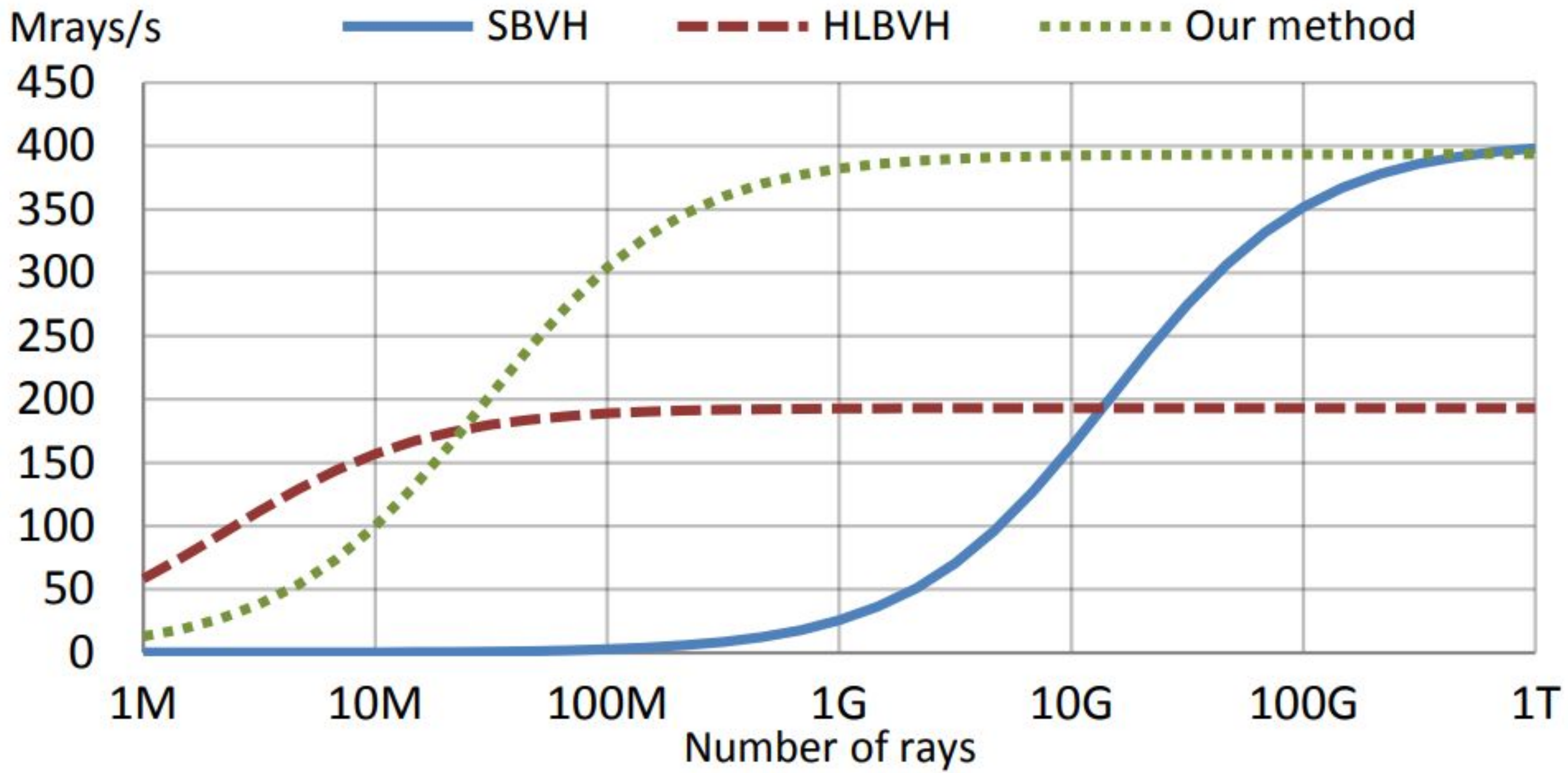
Slow build time

Best known performance!



The papers algorithm

- Builds the BVH in real time
- High performance (within 10% of SBVH)



Performance averaged over 20 test scenes

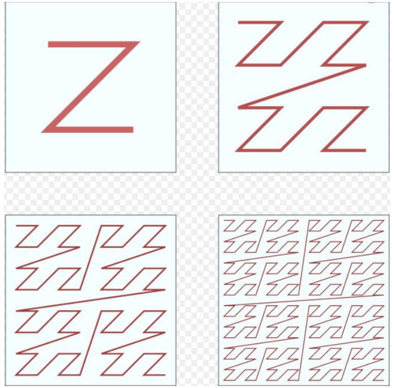
How did they do it?

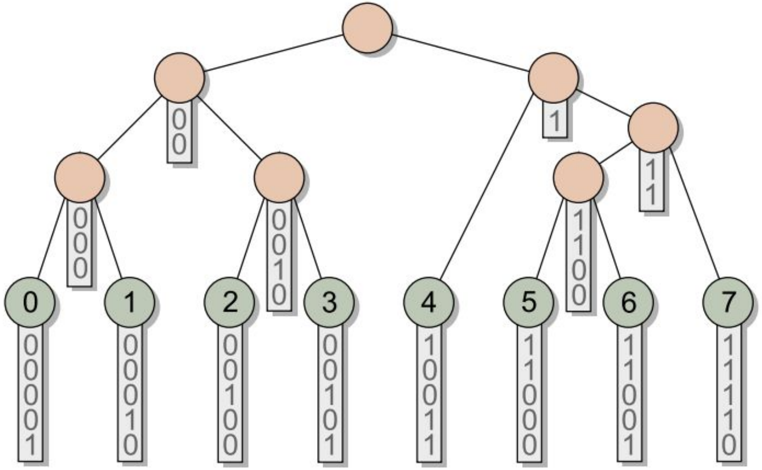
- Treelet optimization
- Triangle splitting

	No splits	30% splits
Triangle splitting		0.4 ms
Initial BVH construction	5.4 ms	6.6 ms
Optimization	17.0 ms	21.4 ms
Post-processing	1.2 ms	1.6 ms

Initial BVH construction

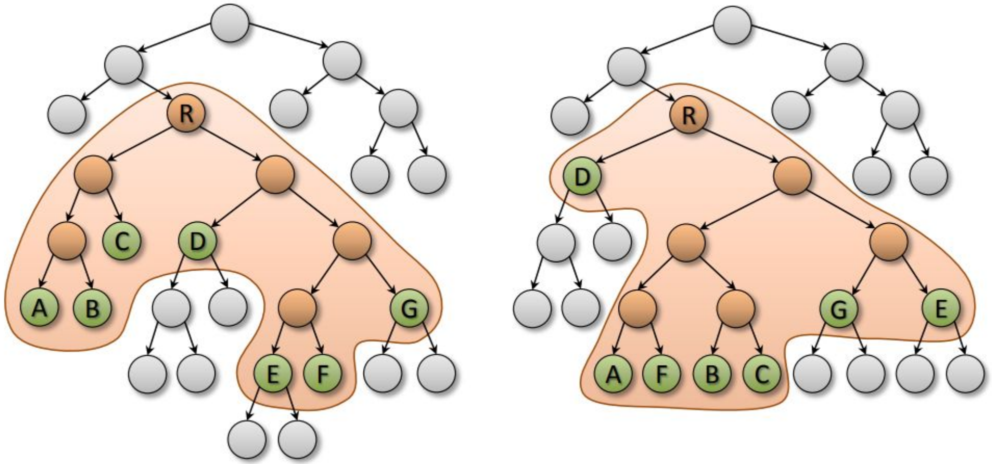
- Uses a method developed by Karras in 2012
- Assigns a 60-bit Morton code to each leaf node
- Creates a binary radix tree





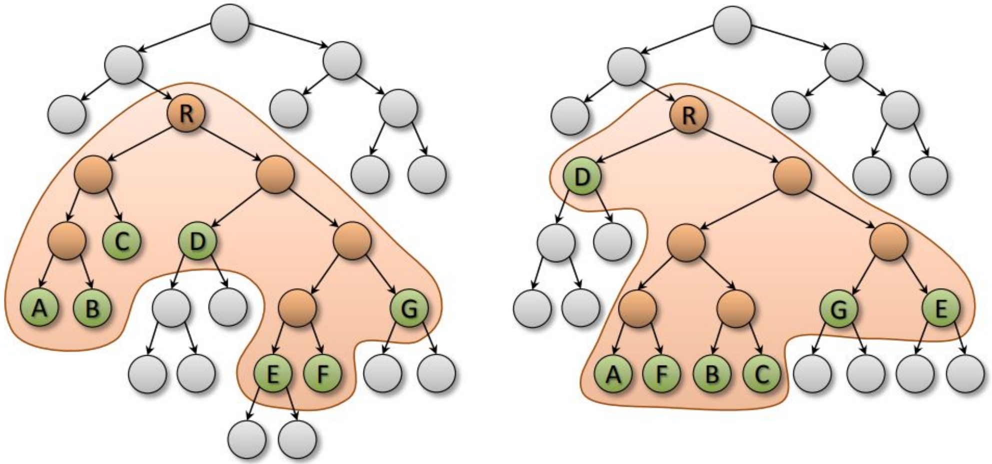
Optimization

- Treelet optimization
- A immediate subtree of a node
- Find the best configuration of that subtree



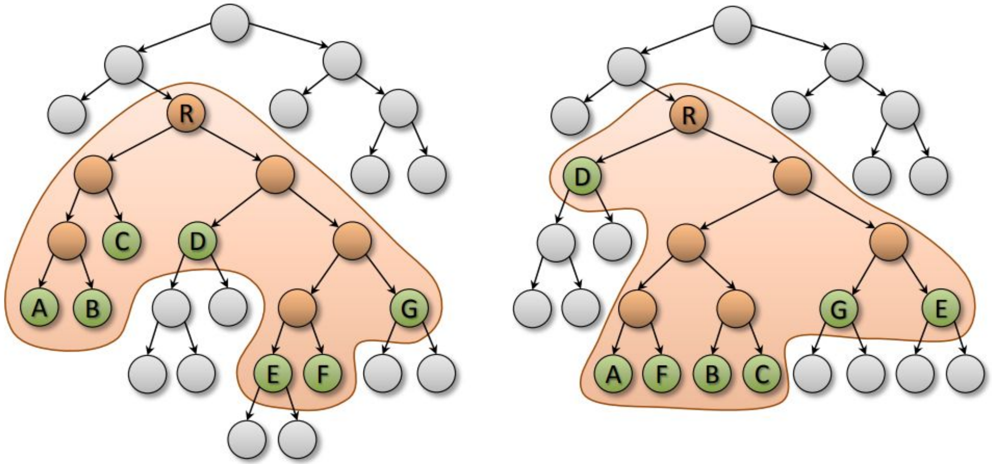
Treelet formation

- Select root-nodes in a bottom up algorithm that avoids overlapping treelets
- Select the nodes of the root with the largest surface area



Naive implementation

- Recursively try every possible subtree with the given nodes
- 7 leaf nodes results in 1.15 million recursive function calls



Dynamic programming

- Represent a subset of leaf-nodes as a bitstring

- 1000000 -> Just leaf node 1
- 1100000 -> Internal node containing node 1 and node 2

- Calculate SAH cost for all subsets with 1 leaf-node, then 2, up to 7
- Final result is stored in c_opt[1111111]

// Calculate surface area for each subset
for $\bar{s} = 1$ **to** $2^n - 1$ **do**
    $a[\bar{s}] \leftarrow \textsc{Area}(\textsc{UnionOfAABBs}(L, \bar{s}))$
end for
// Initialize costs of individual leaves
for $i = 0$ **to** $n - 1$ **do**
    $c_{\text{opt}}[2^i] \leftarrow C(L_i)$
end for
// Optimize every subset of leaves
for $k = 2$ **to** n **do**
    for each $\bar{s} \in [1, 2^n - 1]$ with k set bits **do**
        // Try each way of partitioning the leaves
        $(c_{\bar{s}}, \bar{p}_{\bar{s}}) \leftarrow (\infty, 0)$
        for each $\bar{p} \in \{\text{partitionings of } \bar{s}\}$ **do**
            $c \leftarrow c_{\text{opt}}[\bar{p}] + c_{\text{opt}}[\bar{s} \text{ XOR } \bar{p}]$ // $S \setminus P$
            if $c < c_{\bar{s}}$ **then** $(c_{\bar{s}}, \bar{p}_{\bar{s}}) \leftarrow (c, \bar{p})$
        end for
        // Calculate final SAH cost (Equation 2)
        $t \leftarrow \textsc{TotalNumTriangles}(L, \bar{s})$
        $c_{\text{opt}}[\bar{s}] \leftarrow \min\left((C_i \cdot a[\bar{s}] + c_{\bar{s}}), (C_t \cdot a[\bar{s}] \cdot t)\right)$
        $\bar{p}_{\text{opt}}[\bar{s}] \leftarrow \bar{p}_{\bar{s}}$
    end for
end for

Post processing

- Every leaf node contains one triangle
- Collapse leaf nodes into nodes with multiple triangles
- Make sure there are no overlap between threads

$$C(n) = \min \begin{cases} C_i A(n) + C(n_l) + C(n_r) & (n \in I) \\ C_t A(n) N(n) & (n \in L) \end{cases} \qquad (2)$$

Triangle splitting

Why?

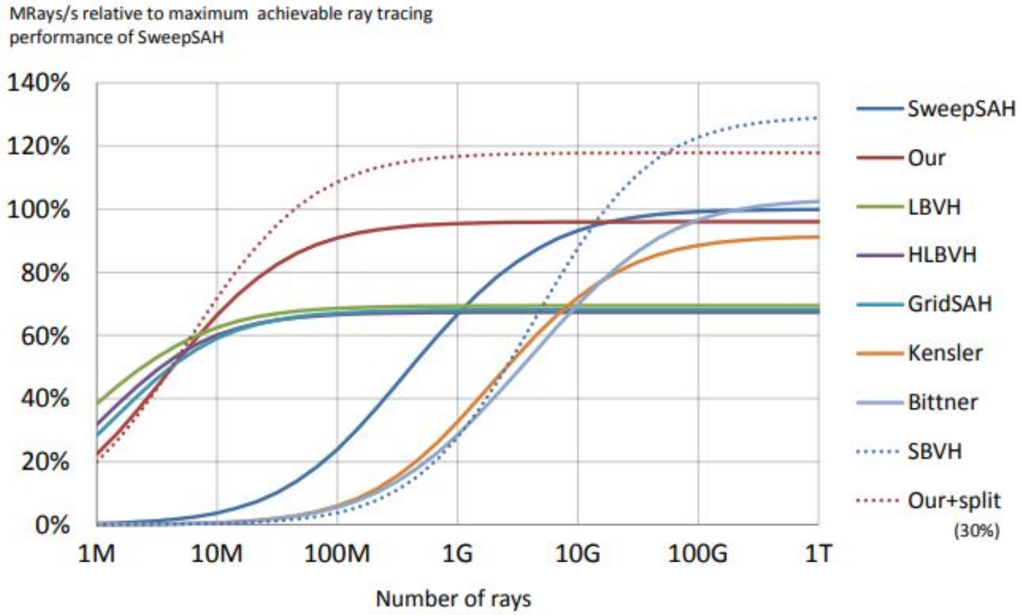
- Reduce surface area
- Reduce overlap between subtrees

What is new?

$$p_t = \left(X^i \cdot (A_{\text{aabb}} - A_{\text{ideal}})\right)^Y,$$

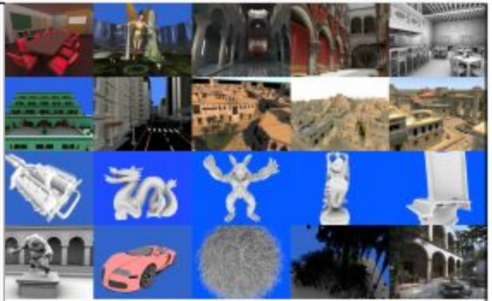
- Experimentally found priority formula
- Cap splitting at certain percentage (usually 30%)

Results



MRays/s relative to maximum achievable ray tracing performance of SweepSAH
140%
120%
100%
80%
60%
40%
20%
0%
1M
10M
100M
1G
10G
100G
1T
Number of rays
SweepSAH
Our
LBVH
HLBVH
GridSAH
Kensler
Bittner
SBVH
Our+split
(30%)

Results (No splitting)



Builder	AVERAGE OF 20 SCENES relative to SweepSAH		
	Perf %	**SAH** %	**Build** %
SweepSAH	100.0	100.0	100.00
Our	96.0	94.4	1.03
LBVH	69.4	131.5	0.28
HLBVH	67.4	129.3	0.57
GridSAH	68.1	129.1	0.70
Kensler	91.5	99.6	552.26
Bittner	103.3	87.7	1083.24

Builder	AVERAGE OF 20 SCENES relative to SBVH		
SBVH	**Perf**	**SAH**	**Build**
Our (no splits)	%	%	%
Our (10% splits)	100.0	100.0	100.00
Our (30% splits)	76.5	111.5	0.14
Our (50% splits)	87.2	108.8	0.15
Our (match SBVH)	91.0	108.0	0.17
ESC (match SBVH)	92.5	108.3	0.20
EVH (match SBVH)	90.7	107.4	0.16