Building (good) BVHs

Arsène Pérard-Gayot

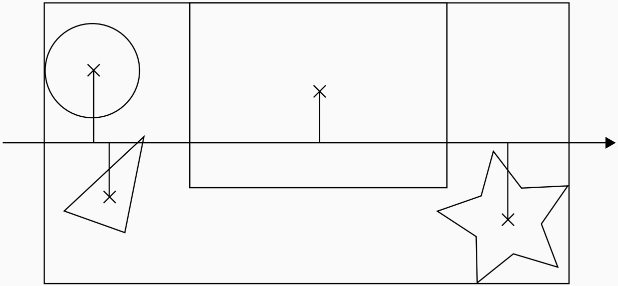
Building Strategies

- Naive (middle, median split) – *don't do that*
- Sweep SAH evaluation
- Binning SAH evaluation
- Sweep SAH evaluation + Spatial Splits
- Sweep SAH evaluation + Pre-splitting

BVH Construction

General Idea

- Split objects (triangles) in two *disjoint* sets: L and R
- Choosing L and R:
  - 2^{N-1} such partitions for N objects: *impractical*
  - Idea: Sort primitives according to centroid position



Sweep SAH

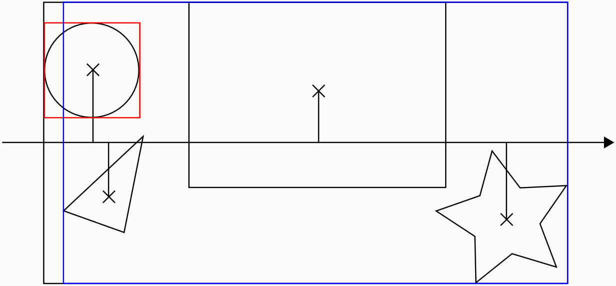
Reminder

$$SAH(P) = C_t + C_i \left(\frac{SA(L)}{SA(P)} N(L) + \frac{SA(R)}{SA(P)} N(R) \right)$$

Minimizing the SAH

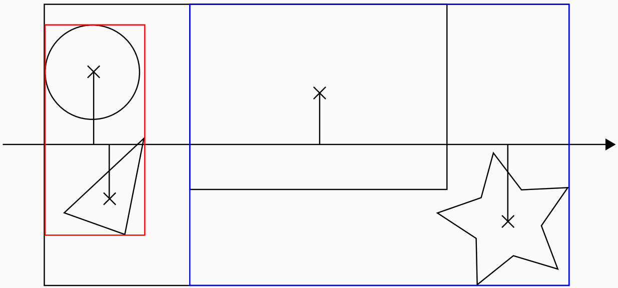
- Iterate through sorted primitives
- Select (L, R) with minimum SAH
- Omit common terms C_i and C_t
    - Irrelevant when minimizing

Sweep SAH



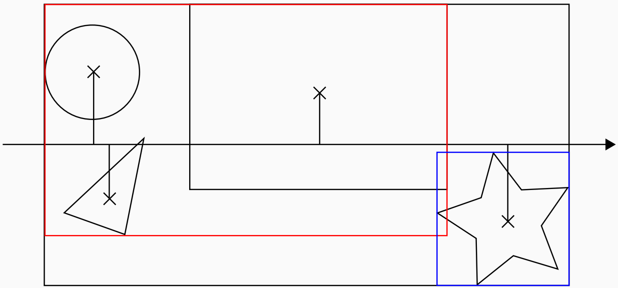
$$SAH(P) = C_t + C_i \left(\frac{SA(L)}{SA(P)} N(L) + \frac{SA(R)}{SA(P)} N(R) \right)$$

Sweep SAH



$$SAH(P) = C_t + C_i \left(\frac{SA(L)}{SA(P)} N(L) + \frac{SA(R)}{SA(P)} N(R) \right)$$

Sweep SAH



$$SAH(P) = C_t + C_i \left(\frac{SA(L)}{SA(P)} N(L) + \frac{SA(R)}{SA(P)} N(R) \right)$$

Efficient SAH evaluation

Important

$SA(L)$ can be incrementally computed by extending the bounding box of L at every step, *but not* $SA(R)$

Sweeping

- Two-step process: right-to-left and then left-to-right
- Right-to-left: compute and record $\frac{SA(R)}{SA(N)} N(R)$
- Left-to-right: compute full SAH using stored values

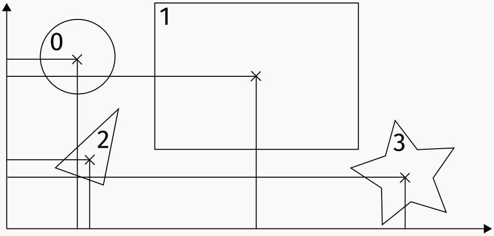
Sweep SAH: Algorithm

- For each $axis \in \{x, y, z\}$
  - Sort primitives according to projection of centroid on *axis*
  - Sweep from right to left to compute partial cost $\frac{SA(R)}{SA(N)} N(R)$
  - Sweep left to right to compute full cost
  - Choose split (L, R) with lowest cost
- Choose split (L, R) on axis with lowest SAH cost
- Compare lowest cost with cost of not splitting
  - i.e. $N(P)$ – number of primitives in the current node
- Terminate if the split is not beneficial

Sweep SAH: Implementation Notes

- Use *references*: Indices into the array of primitives
- Each node is a bounding box + range of references
- Create 3 arrays of references
  - Initially filled with $0..N$
  - Sort according to projection of centroid on $\{x, y, z\}$
- Each split partitions the 3 arrays of references
  - Use a *stable* (i.e *order preserving*) partitioning algorithm!
  - No need to sort references again

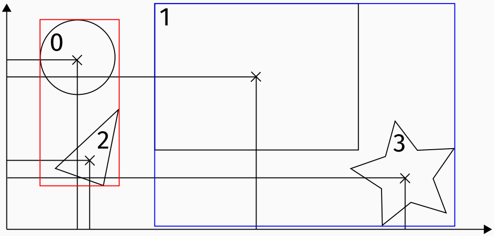
Sweep SAH: Example



Initial State

- References on X: 0, 2, 1, 3
- References on Y: 3, 2, 1, 0
- Root node reference range 0..3 (both ends included)

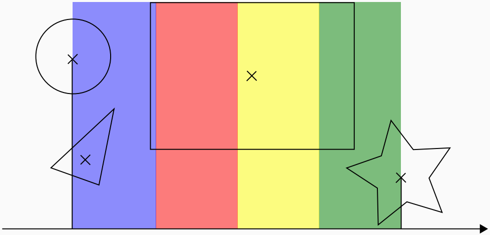
Sweep SAH: Example



Example Partition

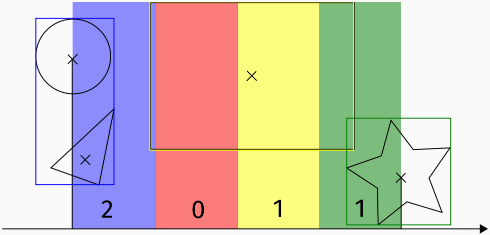
- Partition on X: 0, 2, 1, 3
- References on X are already partitioned
- Partition references on Y
  - Fill boolean array with 1 for red references, 0 otherwise: 1, 0, 1, 0
  - Perform a stable partition of 3, 2, 1, 0 according to flags: 2, 0, 3, 1
- Set the bounding box and range of L and R
  - Ranges: L 0..1, R 2..3

Binning SAH



- When full sweep is too slow
- Compute min and max *centroid* bounds
- Create N (typically small, e.g. 16) equally sized bins on $\{x, y, z\}$

Binning SAH



- When full sweep is too slow
- Compute min and max *centroid* bounds
- Create N (typically small, e.g. 16) equally sized bins on $\{x, y, z\}$
- Put primitives in bins according to their centroid projection
  - Record number of primitives and bounding box per bin
- Sweep *bins* instead of primitives

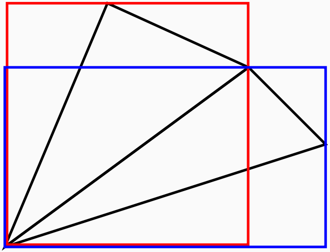
Binning SAH continued

- Produces lower quality trees
- Very fast
- Simple implementation
- Good performance/quality compromise

Spatial Splits

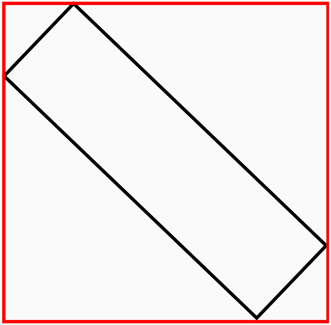
Problems

- Bounding box overlap

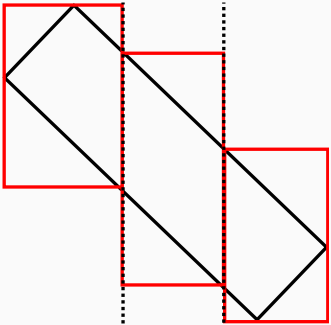


Problems

- Non axis-aligned geometry



Spatial Splits



Split Primitives

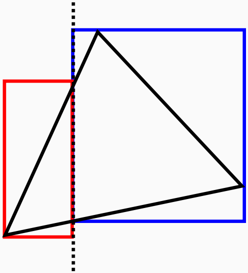
- Either as a pre-pass, before BVH construction, or
- Adaptively during BVH construction

Spatial Split BVH (SBVH)

Splitting Algorithm

- Compute SAH cost of object split (e.g. using Sweep SAH)
- Compute SAH cost of spatial split
- If the spatial split is beneficial, use it
- Otherwise, use object split

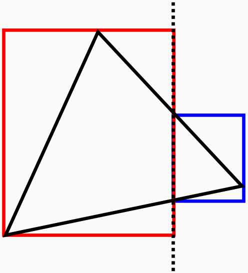
Finding Good Spatial Splits



Spatial Binning

- The SAH cost of spatial splits changes *inside* a primitive
  - Looking at bounding box extrema is not enough

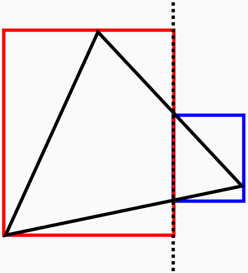
Finding Good Spatial Splits



Spatial Binning

- The SAH cost of spatial splits changes *inside* a primitive
  - Looking at bounding box extrema is not enough

Finding Good Spatial Splits

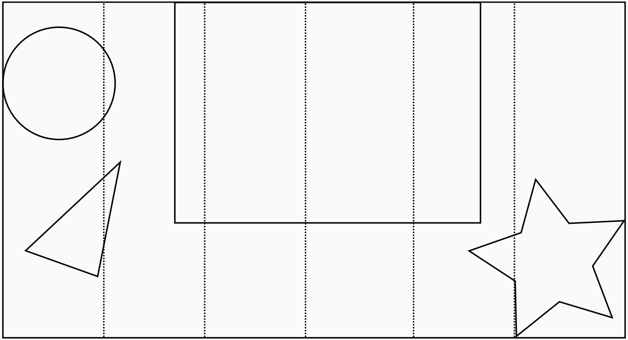


Spatial Binning

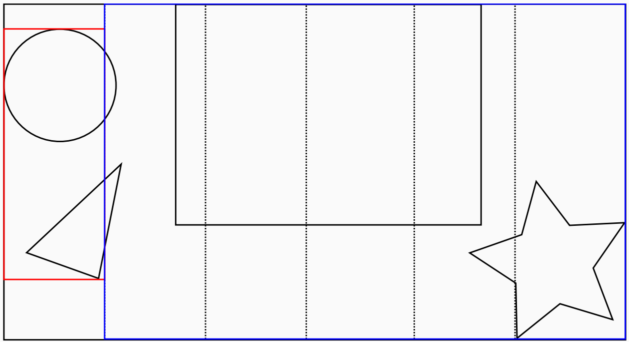
- The SAH cost of spatial splits changes *inside* a primitive
  - Looking at bounding box extrema is not enough
- Use spatial bins

Spatial Binning

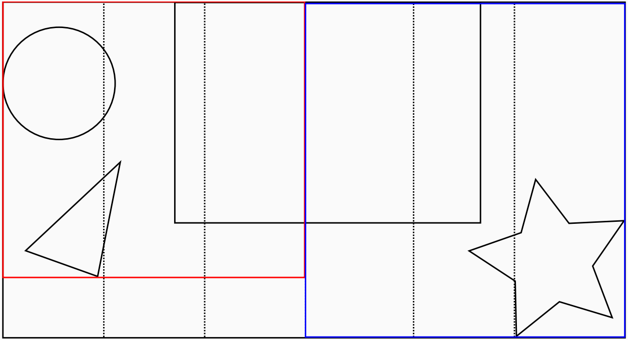
Example



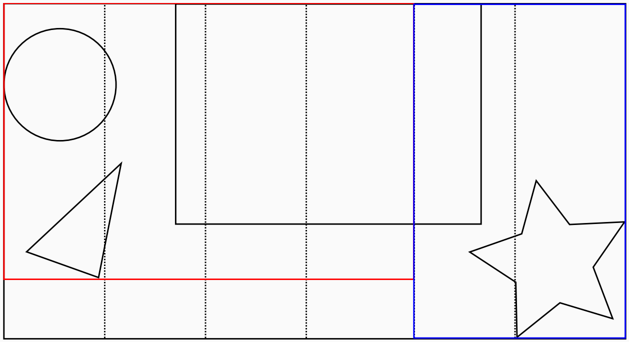
Example



Example

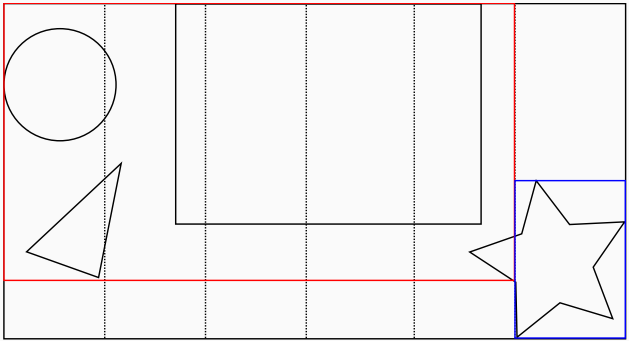


Example



Spatial Binning

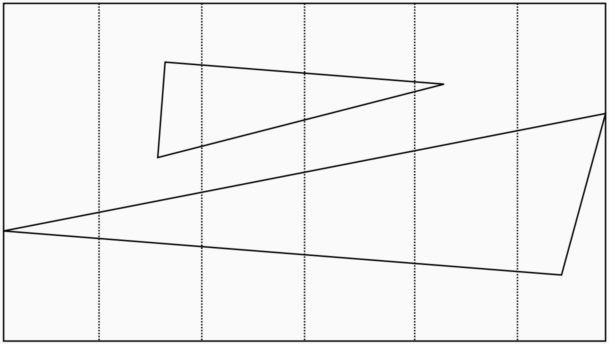
Example



Spatial Binning: Details

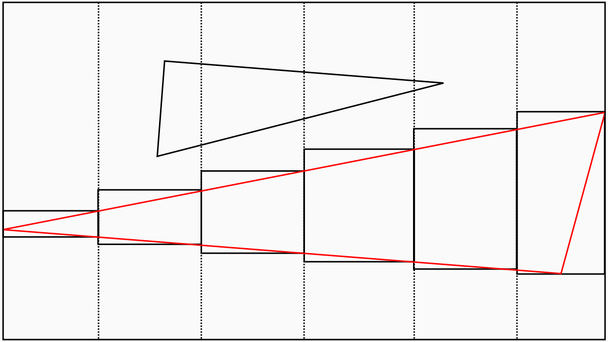
- For each bin that the primitive spans
  - Split the primitive according to the bin
  - Extend the bin with the bounding box after splitting
- Record the number of primitive *entries* and *exits* per bin
  - Increase the entry count of the first bin touched by the primitive
  - Increase the exit count of the last bin touched by the primitive
- Sweep the bins from right to left
  - Accumulate the bounding boxes of the bins in one array
- Sweep the primitives from left to right to compute the cost
  - Number of primitives in L and R from entry and exit counts
  - $N(L) = \sum_{b \in Bins(L)} Entry(b)$
  - $N(R) = N_{total} - \sum_{b \in Bins(L)} Exit(b)$
  - Both $N(L)$ and $N(R)$ only depend on the bins in L

Spatial Binning: Details



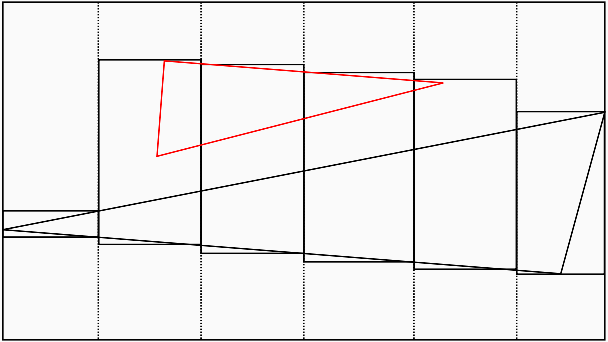
entry	0	0	0	0	0	0
exit	0	0	0	0	0	0

Spatial Binning: Details



entry	1	0	0	0	0	0
exit	0	0	0	0	0	1

Spatial Binning: Details



entry	1	1	0	0	0	0
exit	0	0	0	0	1	1

SBVH Performance

- Performance between $+5$ and $+40\%$ w.r.t Full Sweep SAH
- Number of references between $+3$ and $+30\%$
- In extreme cases, can reach $2\times$ performance *and* $2\times$ references
- Costly spatial split evaluation
  - Spatial splits typically restricted to the top of the tree
  - Measure overlap between L and R after object split: $\frac{SA(L \cap R)}{SA(S)}$
    where S is the bounding box of the entire scene
  - Split if greater than user parameter α

Pre-splitting

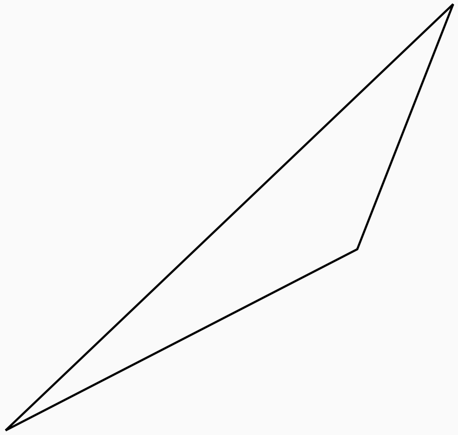
- Simple idea: Perform splitting *before* building the BVH
- Almost no modification to existing BVH builder
- *But* only local knowledge
  - Looking at each primitive independently

Pre-splitting: Edge Volume Heuristic

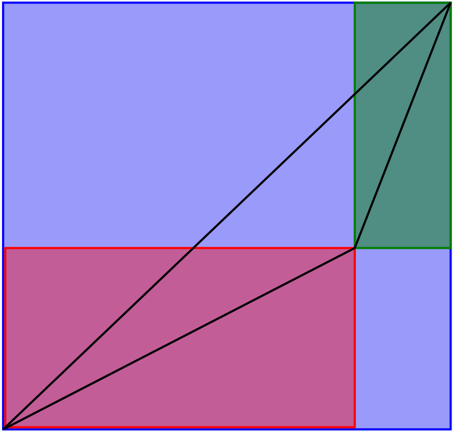
- Economical heuristic to split problematic triangles
- Compute the volume of the bounding box of each edge
  - Select edge with largest volume
  - If volume is above threshold, split edge in the middle
  - Recurse on resulting triangles
- Threshold derived from total scene volume: $\frac{V(S)}{2^t}$ with $t = 14$
- Watertight: Avoid cracks at shared edges, numerically robust
- Only affects *really bad* triangles

Pre-splitting: Edge Volume Heuristic

Example

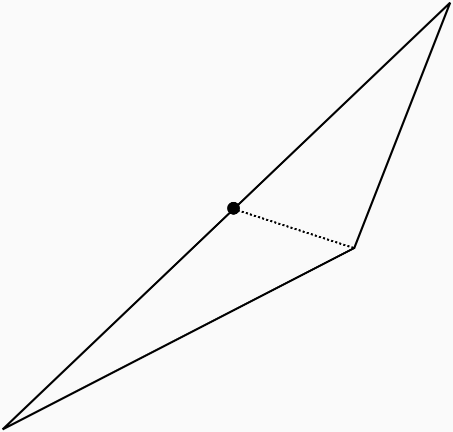


Example



Pre-splitting: Edge Volume Heuristic

Example



Pre-splitting: Edge Volume Heuristic

- Performance identical for non-problematic scenes
- Improved performance in pathological cases
  - Rotated objects, very long and thin triangles
- Highly parallelizable
- Optimization: Remove duplicate references in BVH leaves

Summary

- Building good BVHs is hard
- Still active research topic
- Binning SAH construction is a good compromise between:
    - Traversal performance
    - Build times
    - Ease of implementation
- See references

References

- Holger Dammertz and Alexander Keller.
  Edge volume heuristic - robust triangle subdivision for improved BVH performance.
  In *Proc. 2008 IEEE/EG Symposium on Interactive Ray Tracing*.
- Martin Stich, Heiko Friedrich, and Andreas Dietrich.
  Spatial splits in bounding volume hierarchies.
  In *Proc. High-Performance Graphics 2009*.
- Ingo Wald.
  On fast construction of SAH-based bounding volume hierarchies.
  In *Proc. 2007 IEEE Symposium on Interactive Ray Tracing*.