MSBVH: An Efficient Acceleration Data Structure for Ray Traced Motion Blur

Leonhard Grünschloß Martin Stich
Sehera Nawaz Alexander Keller

August 6, 2011

Principles of Accelerated Ray Tracing

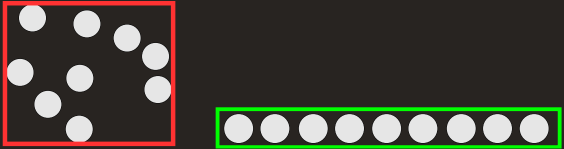
Hierarchical culling

- object list partitioning ⇒ BVH

Principles of Accelerated Ray Tracing

Hierarchical culling

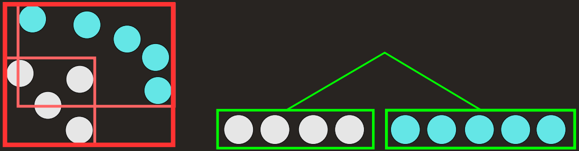
- object list partitioning ⇒ BVH



Principles of Accelerated Ray Tracing

Hierarchical culling

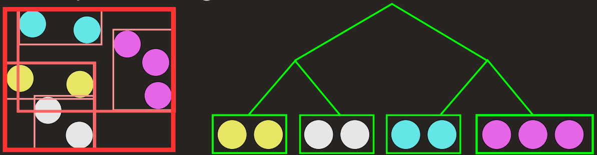
- object list partitioning ⇒ BVH



Principles of Accelerated Ray Tracing

Hierarchical culling

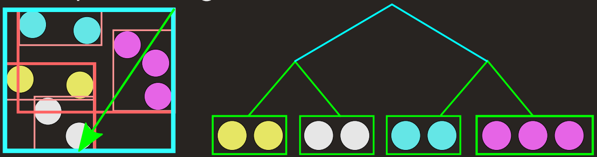
- object list partitioning ⇒ BVH



Principles of Accelerated Ray Tracing

Hierarchical culling

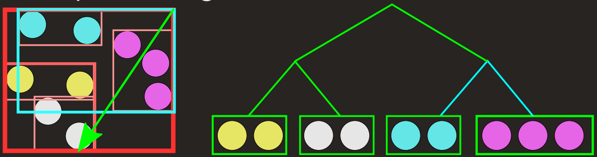
- object list partitioning ⇒ BVH



Principles of Accelerated Ray Tracing

Hierarchical culling

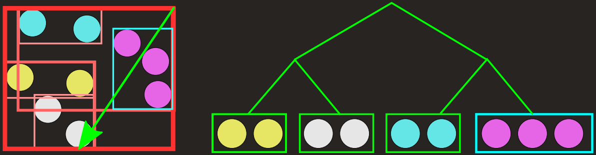
- object list partitioning ⇒ BVH



Principles of Accelerated Ray Tracing

Hierarchical culling

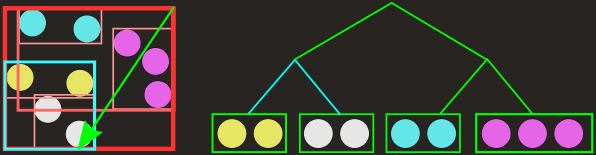
- object list partitioning ⇒ BVH



Principles of Accelerated Ray Tracing

Hierarchical culling

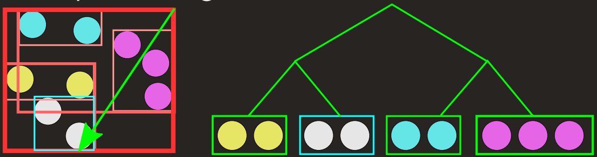
- object list partitioning ⇒ BVH



Principles of Accelerated Ray Tracing

Hierarchical culling

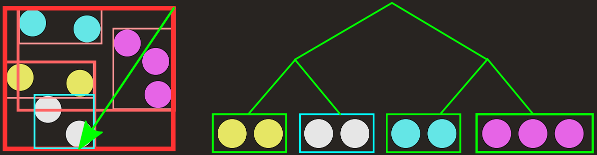
- object list partitioning ⇒ BVH



Principles of Accelerated Ray Tracing

Hierarchical culling

- object list partitioning ⇒ BVH

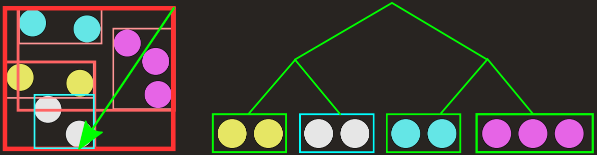


  - bounded memory, but overlapping bounding volumes

Principles of Accelerated Ray Tracing

Hierarchical culling

- object list partitioning ⇒ BVH



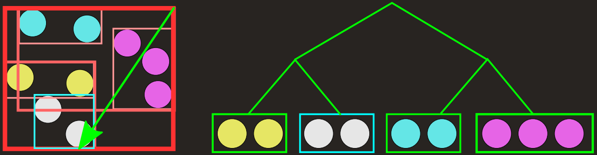
  - bounded memory, but overlapping bounding volumes

- spatial partitioning ⇒ *k*d-tree

Principles of Accelerated Ray Tracing

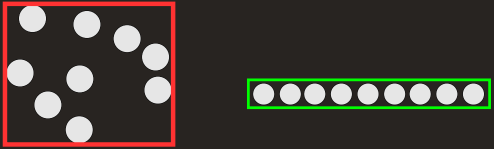
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

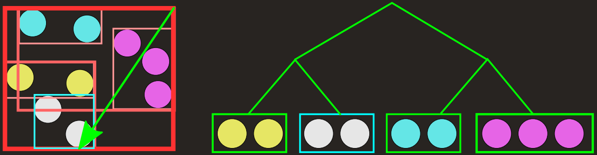
- spatial partitioning ⇒ *k*d-tree



Principles of Accelerated Ray Tracing

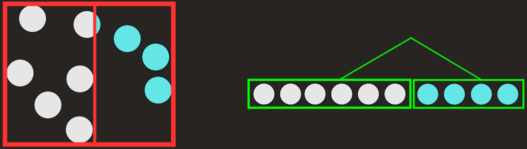
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

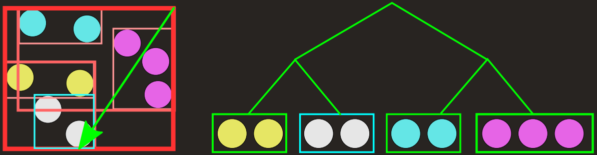
- spatial partitioning ⇒ *k*d-tree



Principles of Accelerated Ray Tracing

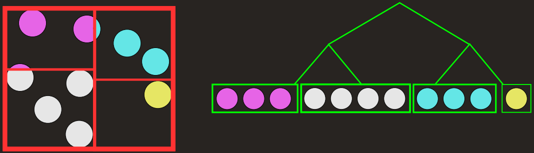
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

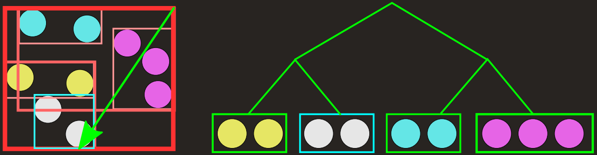
- spatial partitioning ⇒ *k*d-tree



Principles of Accelerated Ray Tracing

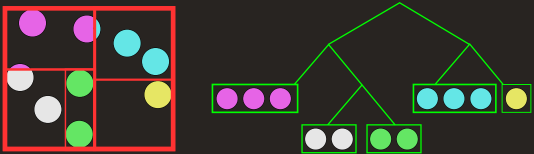
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

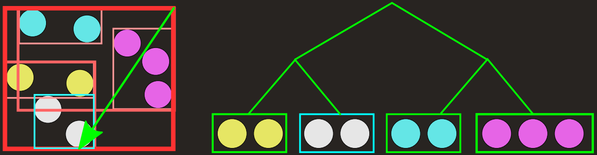
- spatial partitioning ⇒ *k*d-tree



Principles of Accelerated Ray Tracing

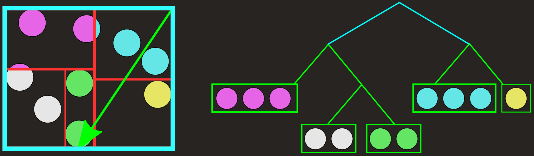
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

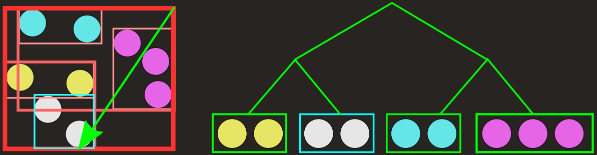
- spatial partitioning ⇒ *k*d-tree



Principles of Accelerated Ray Tracing

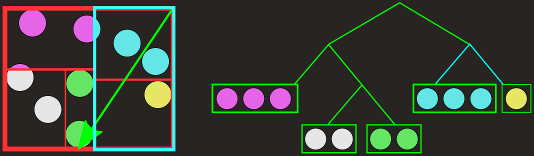
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

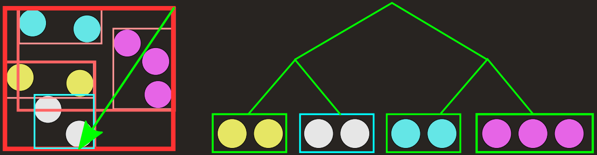
- spatial partitioning ⇒ *k*d-tree



Principles of Accelerated Ray Tracing

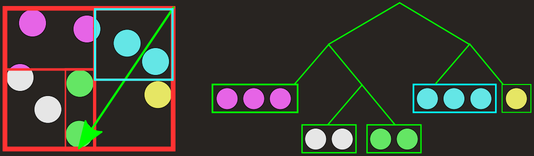
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

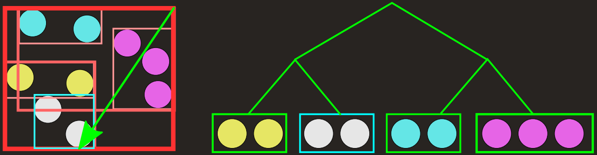
- spatial partitioning ⇒ *k*d-tree



Principles of Accelerated Ray Tracing

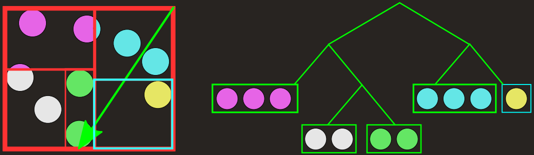
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

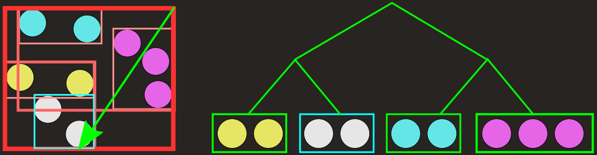
- spatial partitioning ⇒ *k*d-tree



Principles of Accelerated Ray Tracing

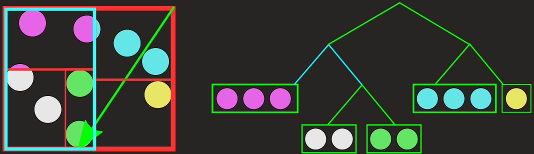
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

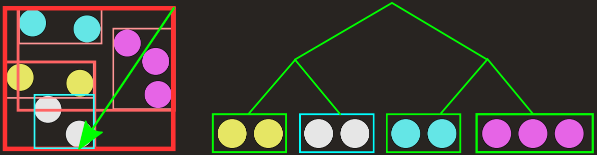
- spatial partitioning ⇒ *k*d-tree



Principles of Accelerated Ray Tracing

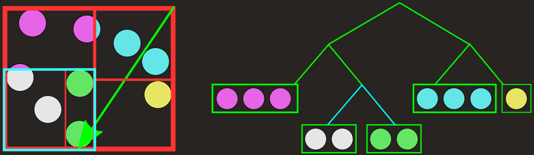
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

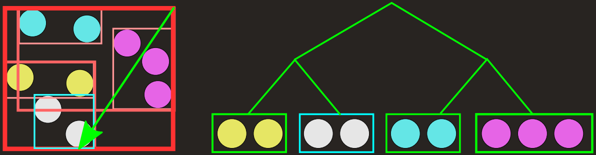
- spatial partitioning ⇒ *k*d-tree



Principles of Accelerated Ray Tracing

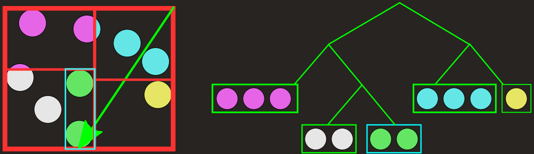
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

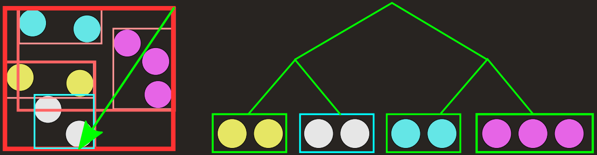
- spatial partitioning ⇒ *k*d-tree



Principles of Accelerated Ray Tracing

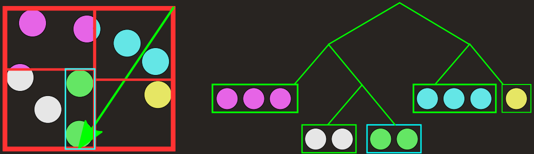
Hierarchical culling

- object list partitioning ⇒ BVH



  - bounded memory, but overlapping bounding volumes

- spatial partitioning ⇒ *k*d-tree



  - nodes do not overlap, but reference duplication

SBVH

Best of both worlds

- object list partitioning whenever overlap is small
- spatial partitioning otherwise

SBVH

Best of both worlds

- object list partitioning whenever overlap is small
- spatial partitioning otherwise

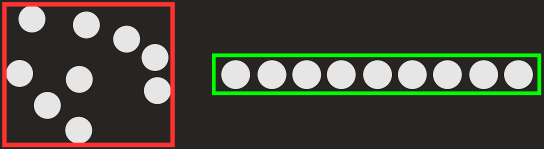
- use spatial splits to build BVH with reference duplication

SBVH

Best of both worlds

- object list partitioning whenever overlap is small
- spatial partitioning otherwise

- use spatial splits to build BVH with reference duplication

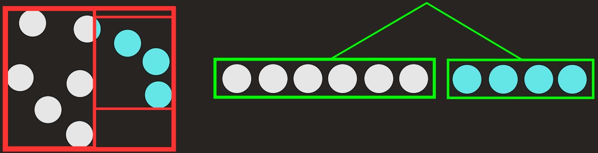


SBVH

Best of both worlds

- object list partitioning whenever overlap is small
- spatial partitioning otherwise

- use spatial splits to build BVH with reference duplication

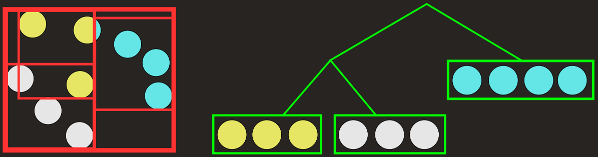


SBVH

Best of both worlds

- object list partitioning whenever overlap is small
- spatial partitioning otherwise

- use spatial splits to build BVH with reference duplication

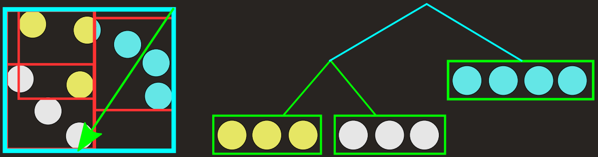


SBVH

Best of both worlds

- object list partitioning whenever overlap is small
- spatial partitioning otherwise

- use spatial splits to build BVH with reference duplication

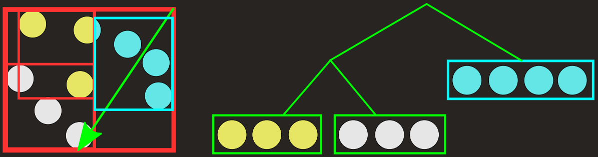


SBVH

Best of both worlds

- object list partitioning whenever overlap is small
- spatial partitioning otherwise

- use spatial splits to build BVH with reference duplication

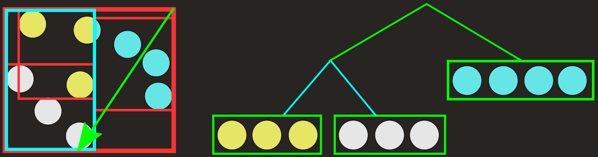


SBVH

Best of both worlds

- object list partitioning whenever overlap is small
- spatial partitioning otherwise

- use spatial splits to build BVH with reference duplication

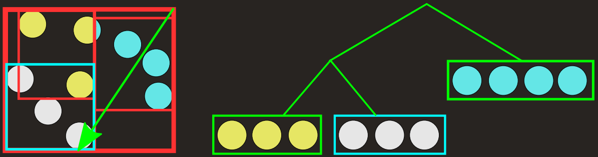


SBVH

Best of both worlds

- object list partitioning whenever overlap is small
- spatial partitioning otherwise

- use spatial splits to build BVH with reference duplication

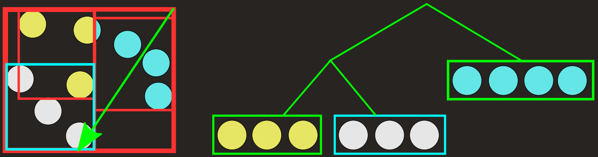


SBVH

Best of both worlds

- object list partitioning whenever overlap is small
- spatial partitioning otherwise

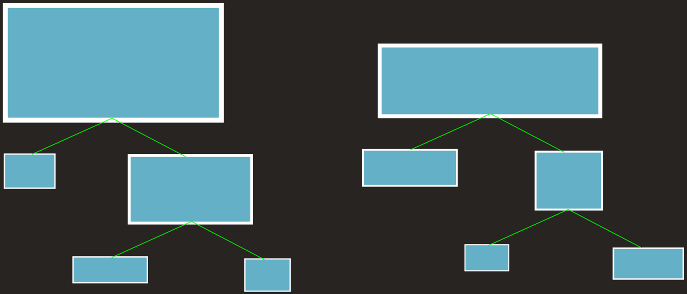
- use spatial splits to build BVH with reference duplication



How to support motion blur?

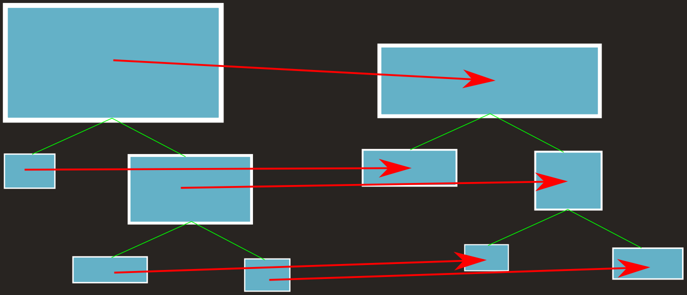
Multiple BVHs Sharing Identical Topology

Convex combination of bounding boxes yields conservative BVH



Multiple BVHs Sharing Identical Topology

Convex combination of bounding boxes yields conservative BVH

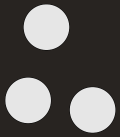


Multiple BVHs Sharing Identical Topology

Example: linear interpolation at leaf level

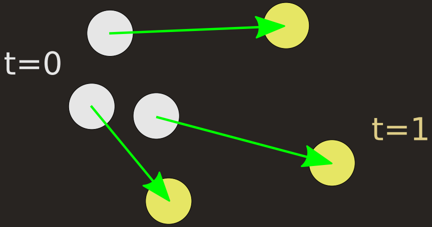
Multiple BVHs Sharing Identical Topology

Example: linear interpolation at leaf level



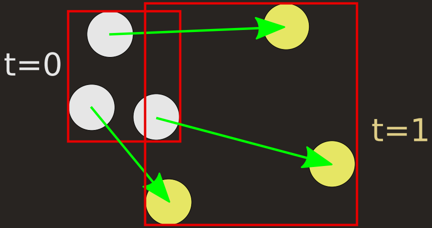
Multiple BVHs Sharing Identical Topology

Example: linear interpolation at leaf level



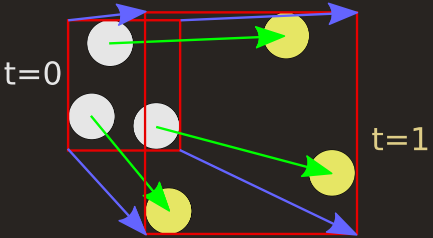
Multiple BVHs Sharing Identical Topology

Example: linear interpolation at leaf level



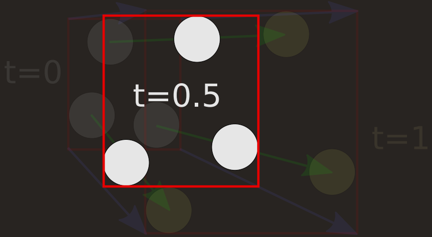
Multiple BVHs Sharing Identical Topology

Example: linear interpolation at leaf level



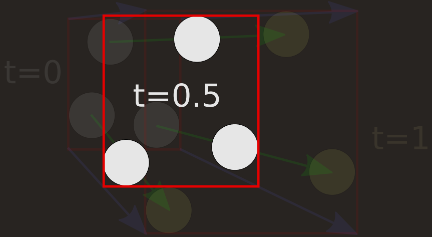
Multiple BVHs Sharing Identical Topology

Example: linear interpolation at leaf level



Multiple BVHs Sharing Identical Topology

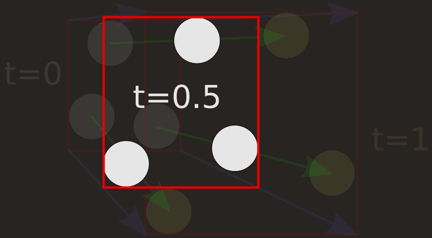
Example: linear interpolation at leaf level



- acceptable memory overhead

Multiple BVHs Sharing Identical Topology

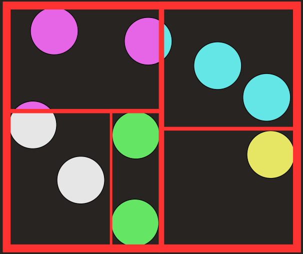
Example: linear interpolation at leaf level



- acceptable memory overhead
- allows for very tight bounding boxes for every ray time *t*

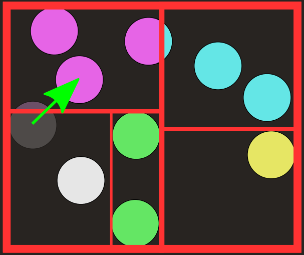
Interpolation and Spatial Splits

Can a *k*d-tree be interpolated?



Interpolation and Spatial Splits

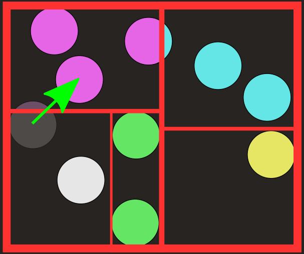
Can a *k*d-tree be interpolated?



- objects can move across split planes
  - thus node references change!

Interpolation and Spatial Splits

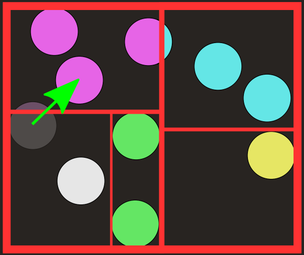
Can a *k*d-tree be interpolated?



- objects can move across split planes
  - thus node references change!
- hierarchy over convex hulls is inefficient

Interpolation and Spatial Splits

Can a *k*d-tree be interpolated?



- objects can move across split planes
  - thus node references change!
- hierarchy over convex hulls is inefficient
- splitting along time-axis requires lots of memory

Our Contribution

Extend the SBVH to handle motion blur (MSBVH)

- by computing multiple bounding volumes per node
- using classic bounding volume interpolation traversal

Our Contribution

Extend the SBVH to handle motion blur (MSBVH)

- by computing multiple bounding volumes per node
- using classic bounding volume interpolation traversal
    - *which includes spatial splits*

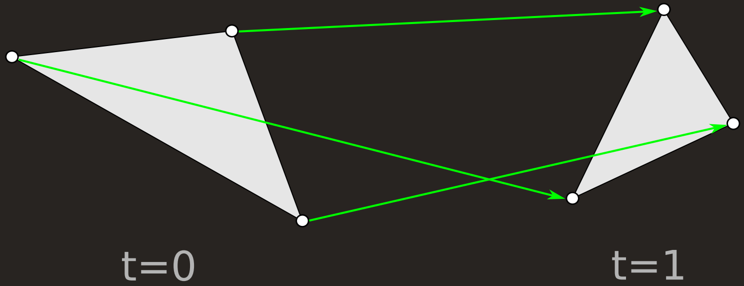
Our Contribution

Extend the SBVH to handle motion blur (MSBVH)

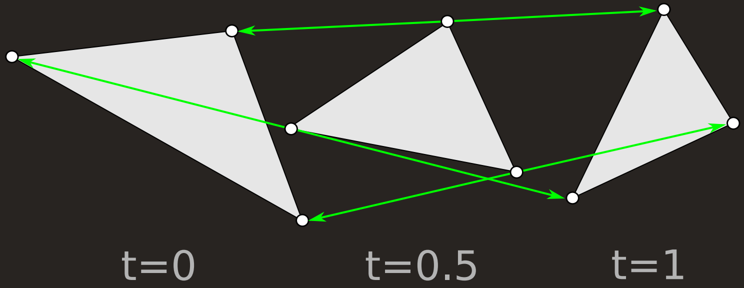
- by computing multiple bounding volumes per node
- using classic bounding volume interpolation traversal
  - *which includes spatial splits*
- memory-efficient (MSBVH)
- reduced bounding volume overlap (MSBVH)

Note: we assume the hierarchy is rebuilt per frame

Algorithm

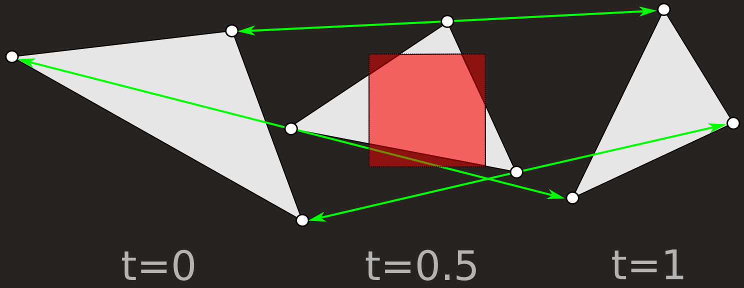


Algorithm



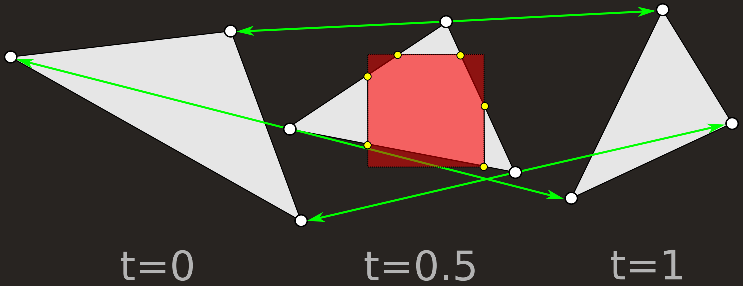
1. Build the SBVH for $t = 0.5$ to determine topology

Algorithm



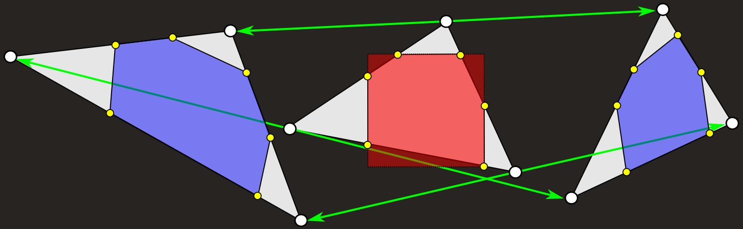
1. Build the SBVH for $t = 0.5$ to determine topology
2. Compute partial primitives in leaf nodes

Algorithm



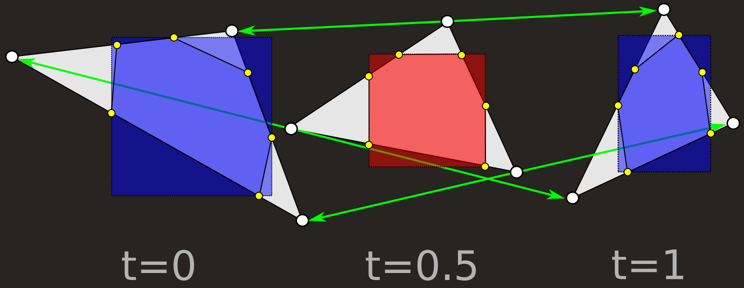
1. Build the SBVH for $t = 0.5$ to determine topology
2. Compute partial primitives in leaf nodes

Algorithm



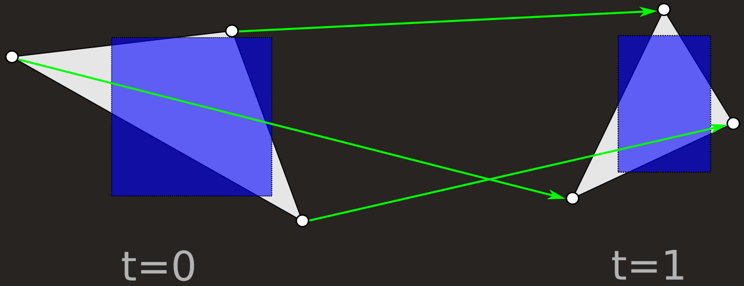
1. Build the SBVH for $t = 0.5$ to determine topology
2. Compute partial primitives in leaf nodes
3. Compute corresponding bounds for $t = 0$ and $t = 1$

Algorithm



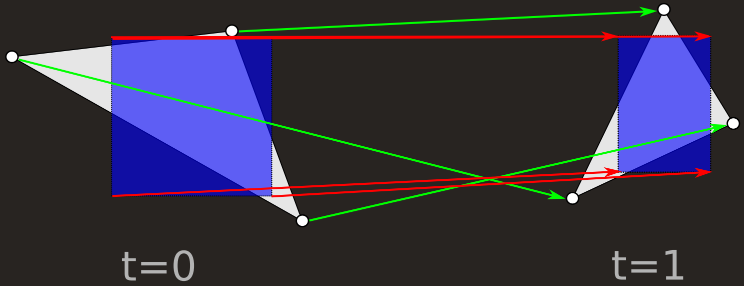
1. Build the SBVH for $t = 0.5$ to determine topology
2. Compute partial primitives in leaf nodes
3. Compute corresponding bounds for $t = 0$ and $t = 1$

Algorithm



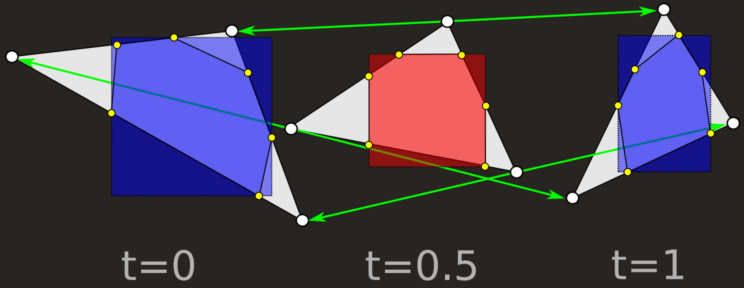
1. Build the SBVH for $t = 0.5$ to determine topology
2. Compute partial primitives in leaf nodes
3. Compute corresponding bounds for $t = 0$ and $t = 1$
4. Propagate bounds to the parent nodes

Algorithm



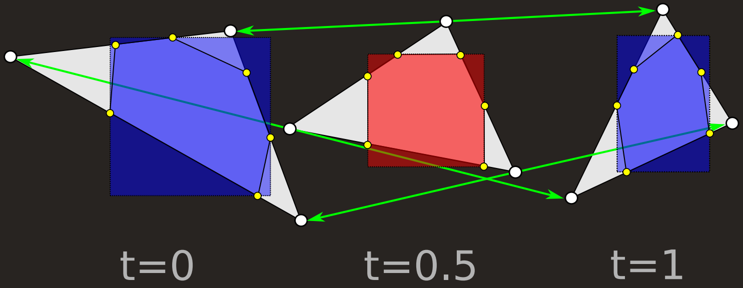
1. Build the SBVH for $t = 0.5$ to determine topology
2. Compute partial primitives in leaf nodes
3. Compute corresponding bounds for $t = 0$ and $t = 1$
4. Propagate bounds to the parent nodes
5. Interpolate these bounds during traversal

Triangles and AABB-Hierarchies under Linear Motion



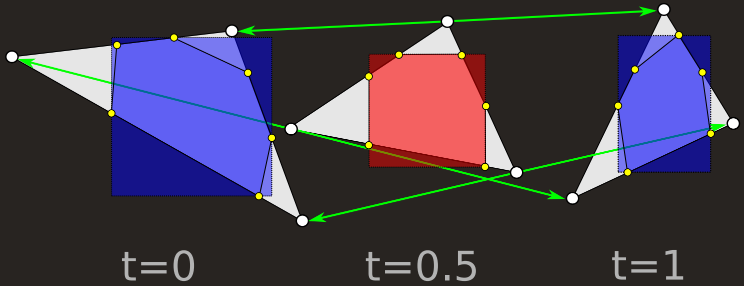
1. Use Sutherland-Hodgman to clip against leaf AABB
2. Results in barycentric coordinates of polygon vertices

Triangles and AABB-Hierarchies under Linear Motion



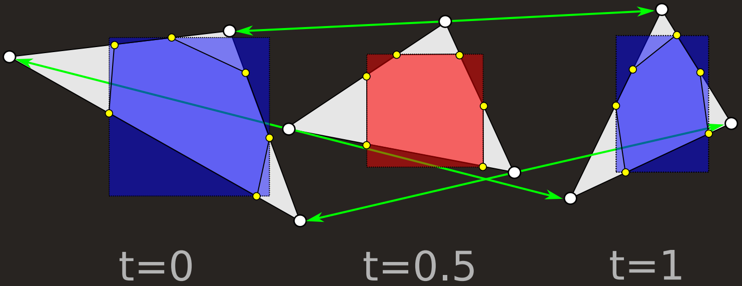
1. Use Sutherland-Hodgman to clip against leaf AABB
2. Results in barycentric coordinates of polygon vertices
3. Compute transformed polygon for $t = 0$ and $t = 1$

Triangles and AABB-Hierarchies under Linear Motion



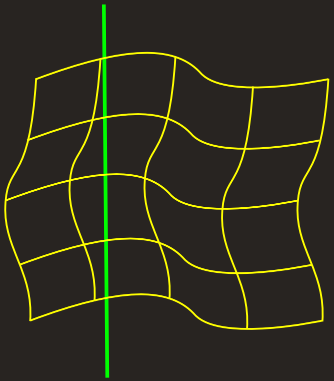
1. Use Sutherland-Hodgman to clip against leaf AABB
2. Results in barycentric coordinates of polygon vertices
3. Compute transformed polygon for $t = 0$ and $t = 1$
4. Bound the transformed polygon

Triangles and AABB-Hierarchies under Linear Motion

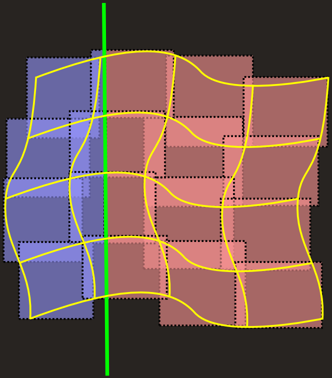


1. Use Sutherland-Hodgman to clip against leaf AABB
2. Results in barycentric coordinates of polygon vertices
3. Compute transformed polygon for $t = 0$ and $t = 1$
4. Bound the transformed polygon
5. No extra storage necessary

Clipping Displaced Subdivision Surfaces

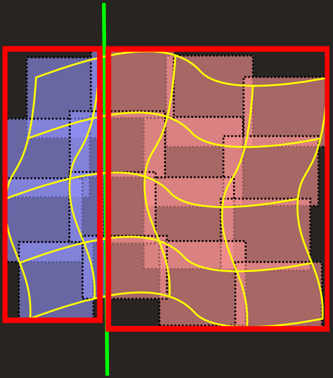


Clipping Displaced Subdivision Surfaces



1. Subdivide along surface parametrization
2. Bound individual elements, e.g. using interval arithmetic

Clipping Displaced Subdivision Surfaces



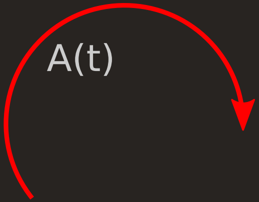
1. Subdivide along surface parametrization
2. Bound individual elements, e.g. using interval arithmetic
3. Clip resulting bounding boxes
4. The union conservatively bounds the clipped primitive

Extensions

- two-level hierarchy: animated instances

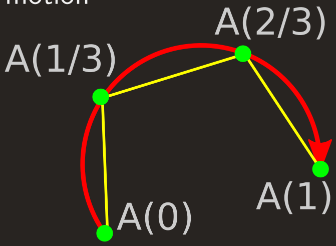
Extensions

- two-level hierarchy: animated instances
- interpolate transformation matrix *elements* to force linear motion



Extensions

- two-level hierarchy: animated instances
- interpolate transformation matrix *elements* to force linear motion

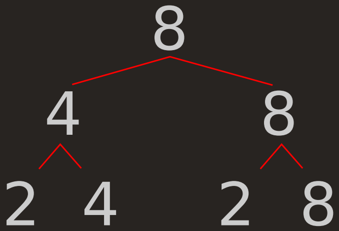


Extensions

- two-level hierarchy: animated instances
- interpolate transformation matrix *elements* to force linear motion
- multiple motion segments

Extensions

- two-level hierarchy: animated instances
- interpolate transformation matrix *elements* to force linear motion
- multiple motion segments
  - restricted to powers of two for propagation up the hierarchy



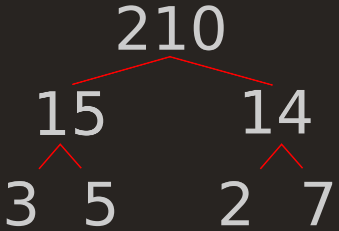
8

4 8

2 4 2 8

Extensions

- two-level hierarchy: animated instances
- interpolate transformation matrix *elements* to force linear motion
- multiple motion segments
  - restricted to powers of two for propagation up the hierarchy

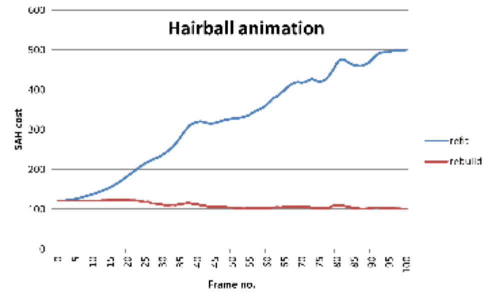


Extensions

- two-level hierarchy: animated instances
- interpolate transformation matrix *elements* to force linear motion
- multiple motion segments
    - restricted to powers of two for propagation up the hierarchy
- higher-order interpolation

Extensions

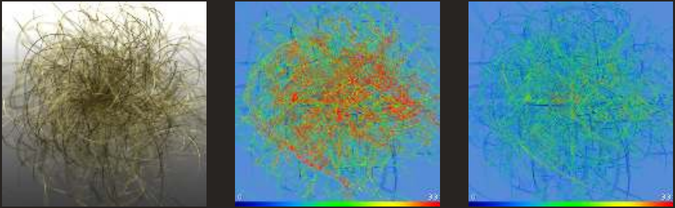
- two-level hierarchy: animated instances
- interpolate transformation matrix *elements* to force linear motion
- multiple motion segments
  - restricted to powers of two for propagation up the hierarchy
- higher-order interpolation
- refitting over multiple frames



Results

BVH traversal with linear interpolation

- reduced SAH cost
- significantly less intersection tests

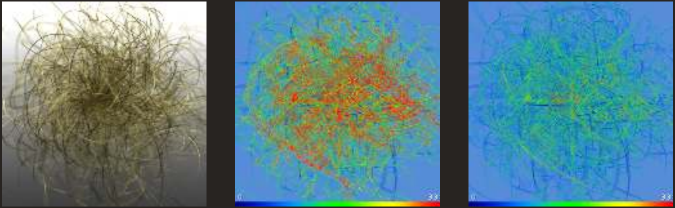


⇒ Video

Results

BVH traversal with linear interpolation

- reduced SAH cost
- significantly less intersection tests



- often less traversal steps
- about 20% rendering speed-up for many scenes

Summary

In practice, works well for single frames

- helps well whenever SBVH helps
- increased build times (between BVH and *k*d-tree)
- prototype implemention in OptiX

Summary

In practice, works well for single frames

- helps well whenever SBVH helps
- increased build times (between BVH and *k*d-tree)
- prototype implemention in OptiX
- spatial splits only avoid overlap for $t = 0.5$
    - topology determined for $t = 0.5$
    - problematic for incoherent motion

Weta Digital is hiring!

http://wetafx.co.nz/siggraph2011