Ray Tracing for Games

Dr. Jacco Bikker - IGAD/BUAS, Breda, February 3

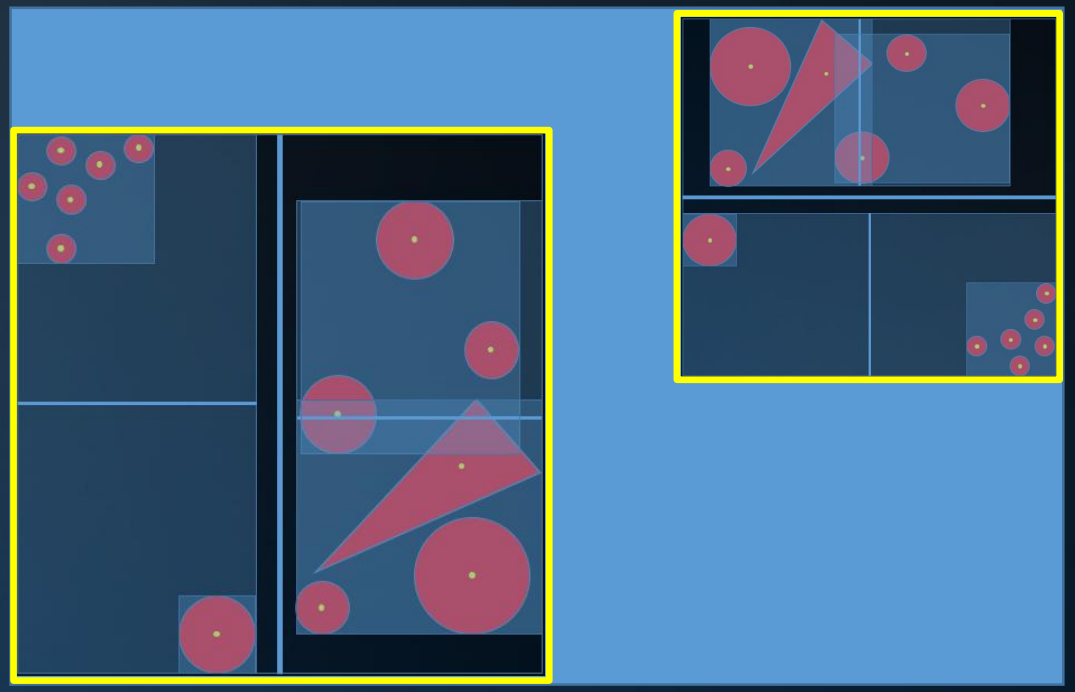
Welcome!

6

Agenda:

- Combining BVHs
- Refitting
- Rigid Motion

Combining BVHs



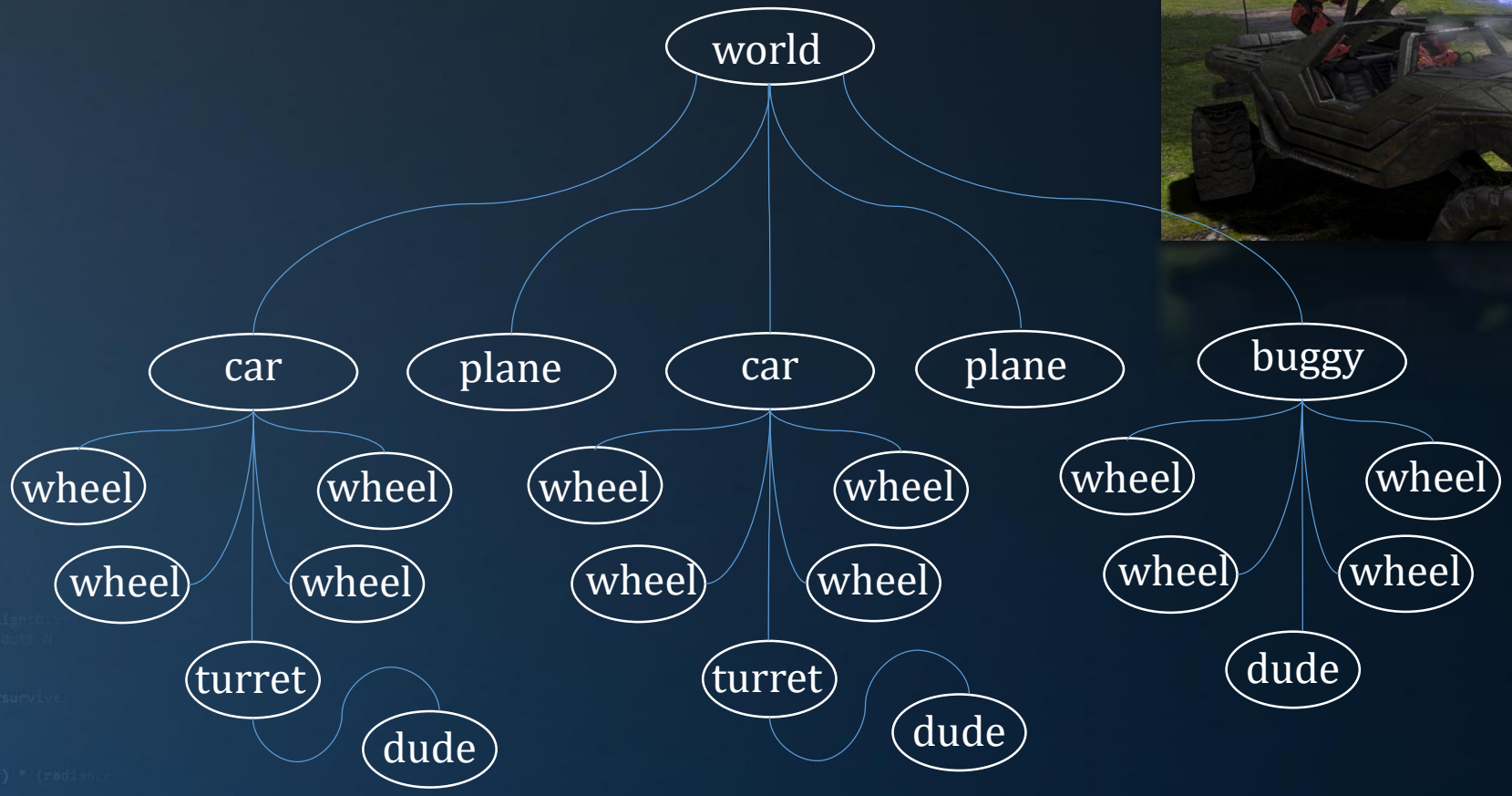
Combining BVHs

BVH #1: Car 1
BVH #2: Car 2
BVH #3: Track

Per frame:

- Rebuild BVH for each car
- Combine car 1 and car 2
- Combine track and cars
- Trace.

Scene Graph


world
car
plane
car
plane
buggy
wheel
wheel
wheel
wheel
wheel
wheel
wheel
wheel
wheel
wheel
wheel
wheel
turret
dude
turret
dude
dude

Scene Graph

If our application uses a scene graph, we can construct a BVH for each scene graph node.

The BVH for each node is built using an appropriate construction algorithm:

- High-quality SBVH for static scenery (offline)
- Fast binned SAH BVHs for dynamic scenery

The extra nodes used to combine these BVHs into a single BVH are known as the *Top-level BVH*.

The Top-level BVH - Construction

Input: *list of axis aligned bounding boxes for transformed scene graph nodes*

Algorithm:

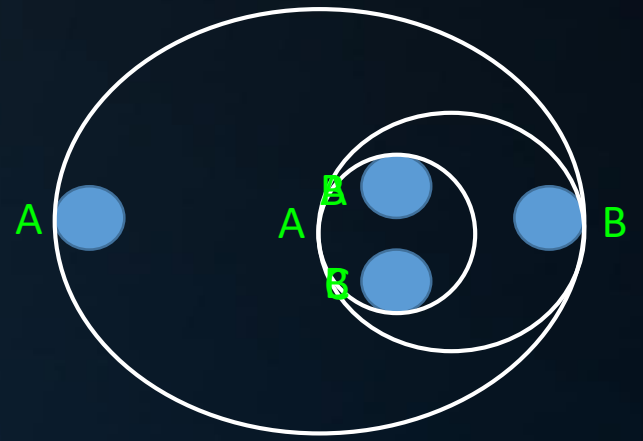
1. Find the two elements in the list for which the AABB has the smallest surface area
2. Create a parent node for these elements
3. Replace the two elements in the list by the parent node
4. Repeat until one element remains in the list.

Note: algorithmic complexity is $O(N^3)$.

The Top-level BVH – Faster Construction*

Algorithm:

```
Node A = list.GetFirst();
Node B = list.FindBestMatch( A );
while (list.size() > 1)
{
   Node C = list.FindBestMatch( B );
   if (A == C)
   {
      list.Remove( A );
      list.Remove( B );
      A = new Node( A, B );
      list.Add( A );
      B = list.FindBestMatch( A );
   }
   else A = B, B = C;
}
```



*: Fast Agglomerative Clustering for Rendering, Walter et al., 2008

Agenda:

- Combining BVHs
- Refitting
- Rigid Motion

Summary of BVH Characteristics

A BVH provides significant freedom.

- No need for a 1-to-1 relation between bounding boxes and primitives
- Bounding boxes may overlap
- Bounding boxes can be altered, as long as they fit in their parent box
- A BVH can be very bad but still valid

Some consequences / opportunities:

- We can rebuild part of a BVH
- We can combine two BVHs into one
- We can *refit* a BVH

Refitting

Q: What happens to the BVH of a tree model, if we make it bend in the wind?

A: Likely, only bounds will change; the topology of the BVH will be the same (or at least similar) in each frame.

Refitting:

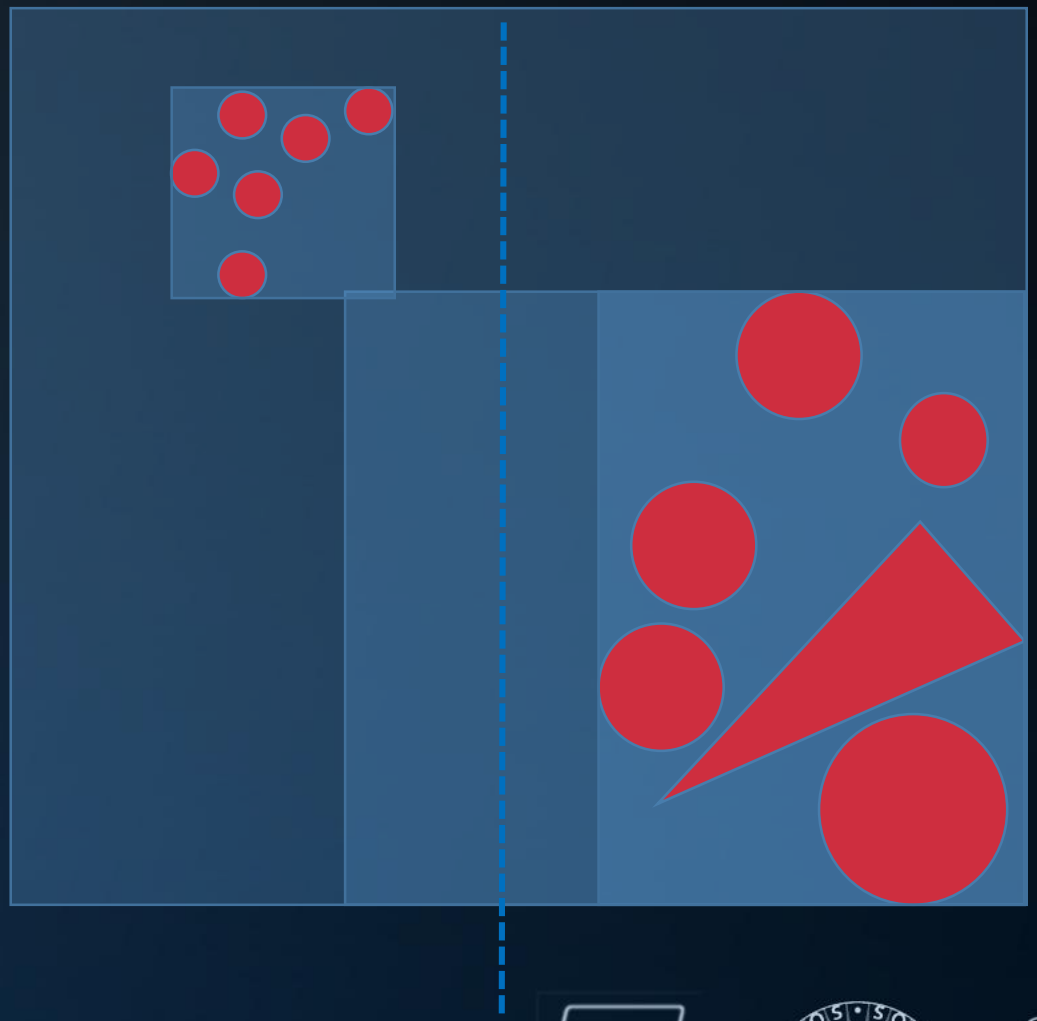
Updating the bounding boxes stored in a BVH to match changed primitive coordinates.

Refitting

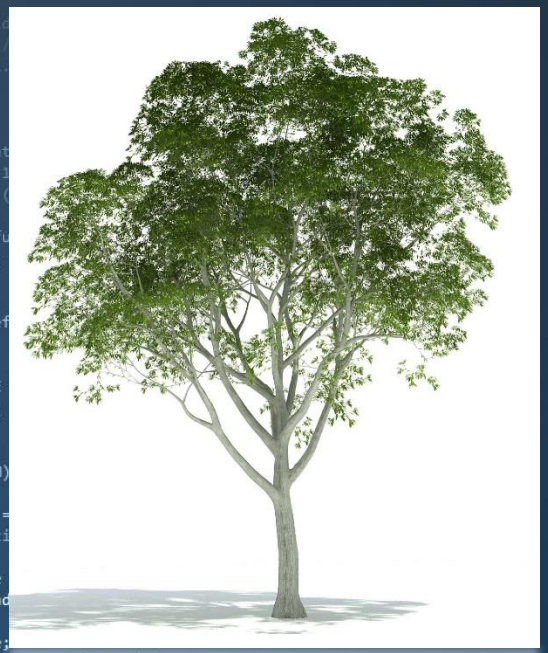
Updating the bounding boxes stored in a BVH to match changed primitive coordinates.

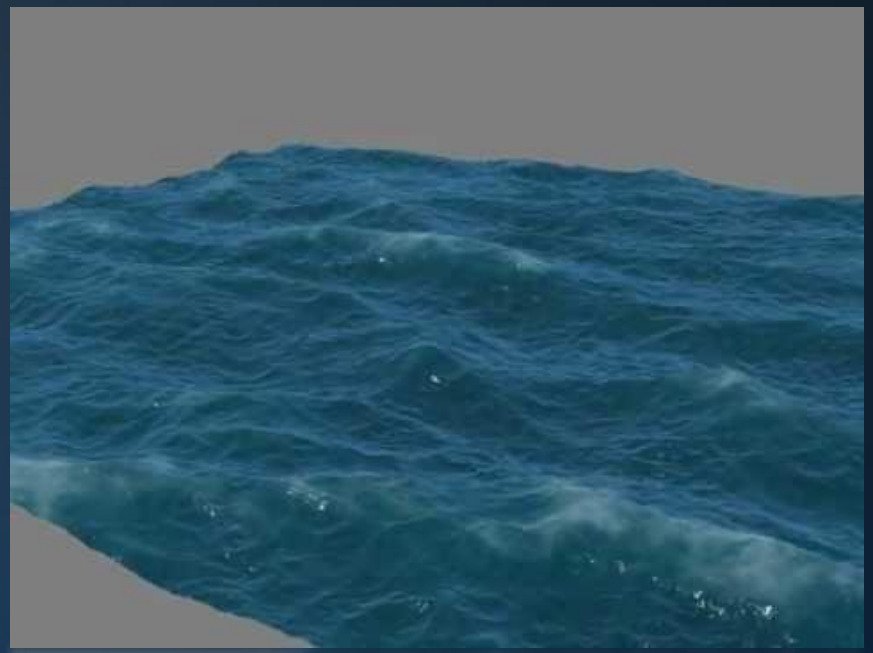
Algorithm:

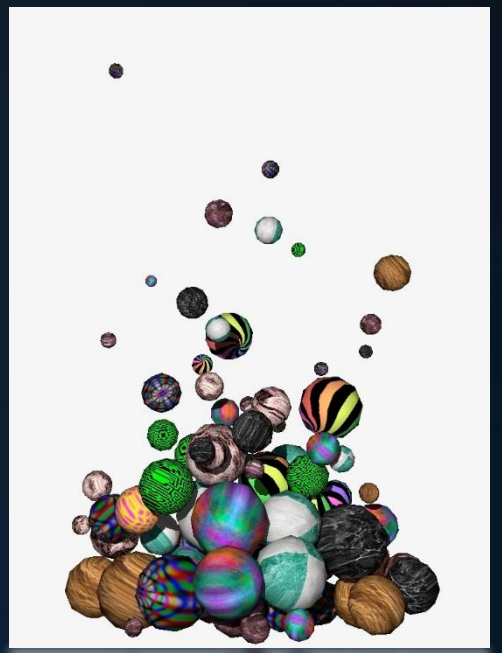
1. For each leaf, calculate the bounds over the primitives it represents
2. Update parent bounds



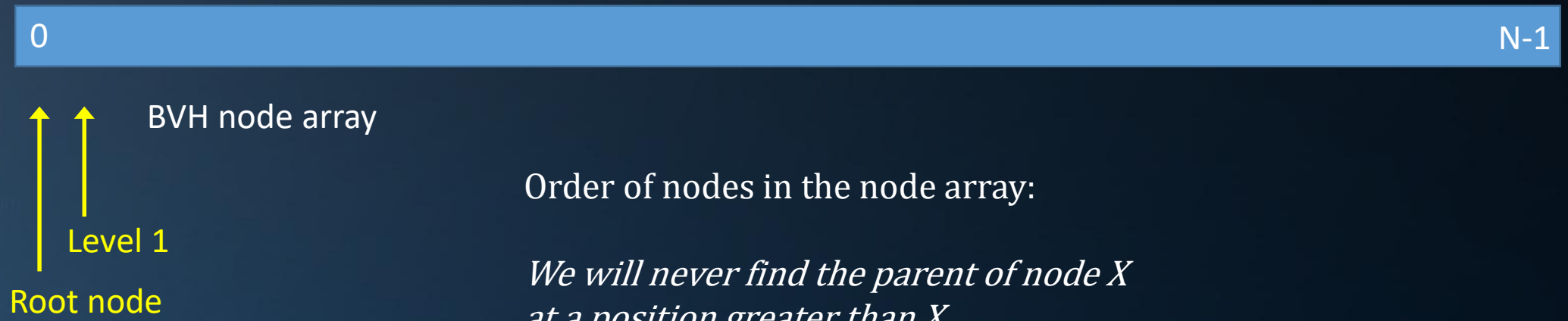
Refitting - Suitability







Refitting – Practical



Order of nodes in the node array:

We will never find the parent of node X at a position greater than X.

Therefore:

```
for( int i = N-1; i >= 0; i-- )
   nodeArray[i].AdjustBounds();
```

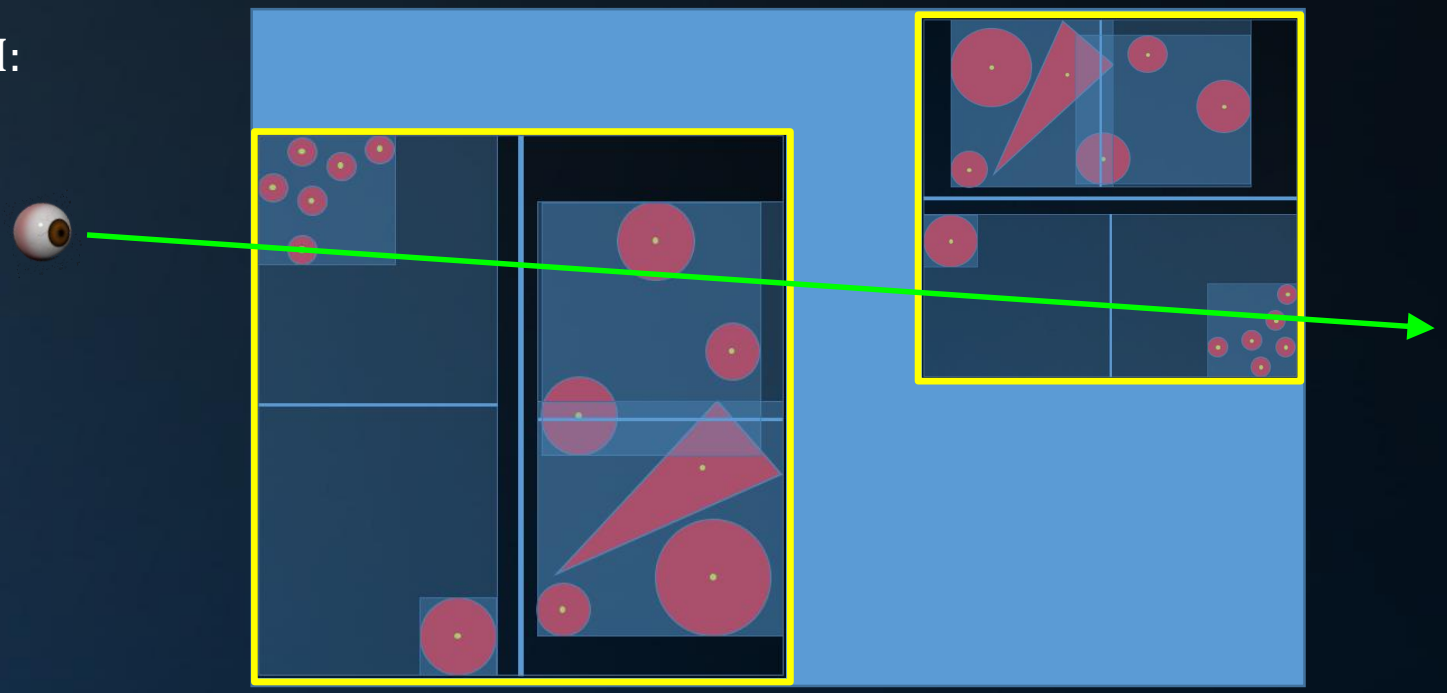
Agenda:

- Combining BVHs
- Refitting
- Rigid Motion

Rigid Motion

Applying rigid motion to a BVH:

1. Refit the top-level BVH
2. Refit the affected BVH



Rigid Motion

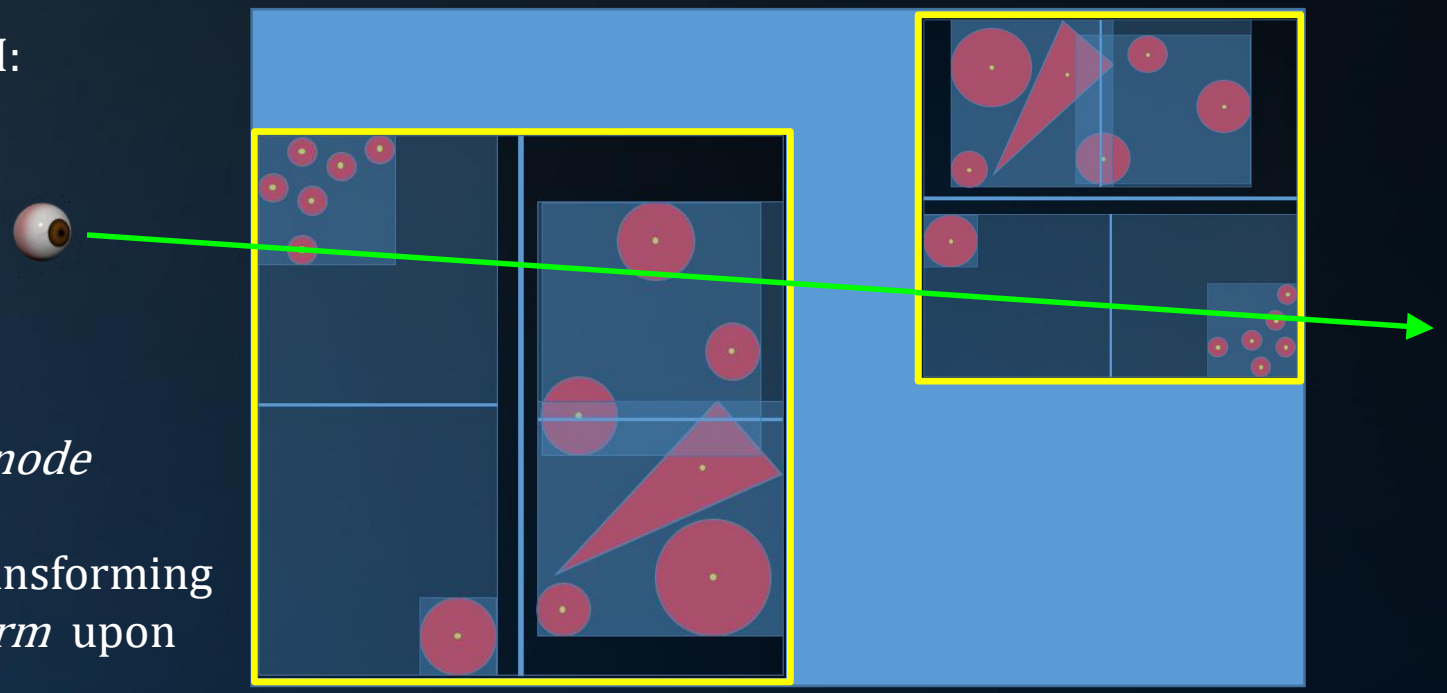
Applying rigid motion to a BVH:

1. Refit the top-level BVH
2. Refit the affected BVH

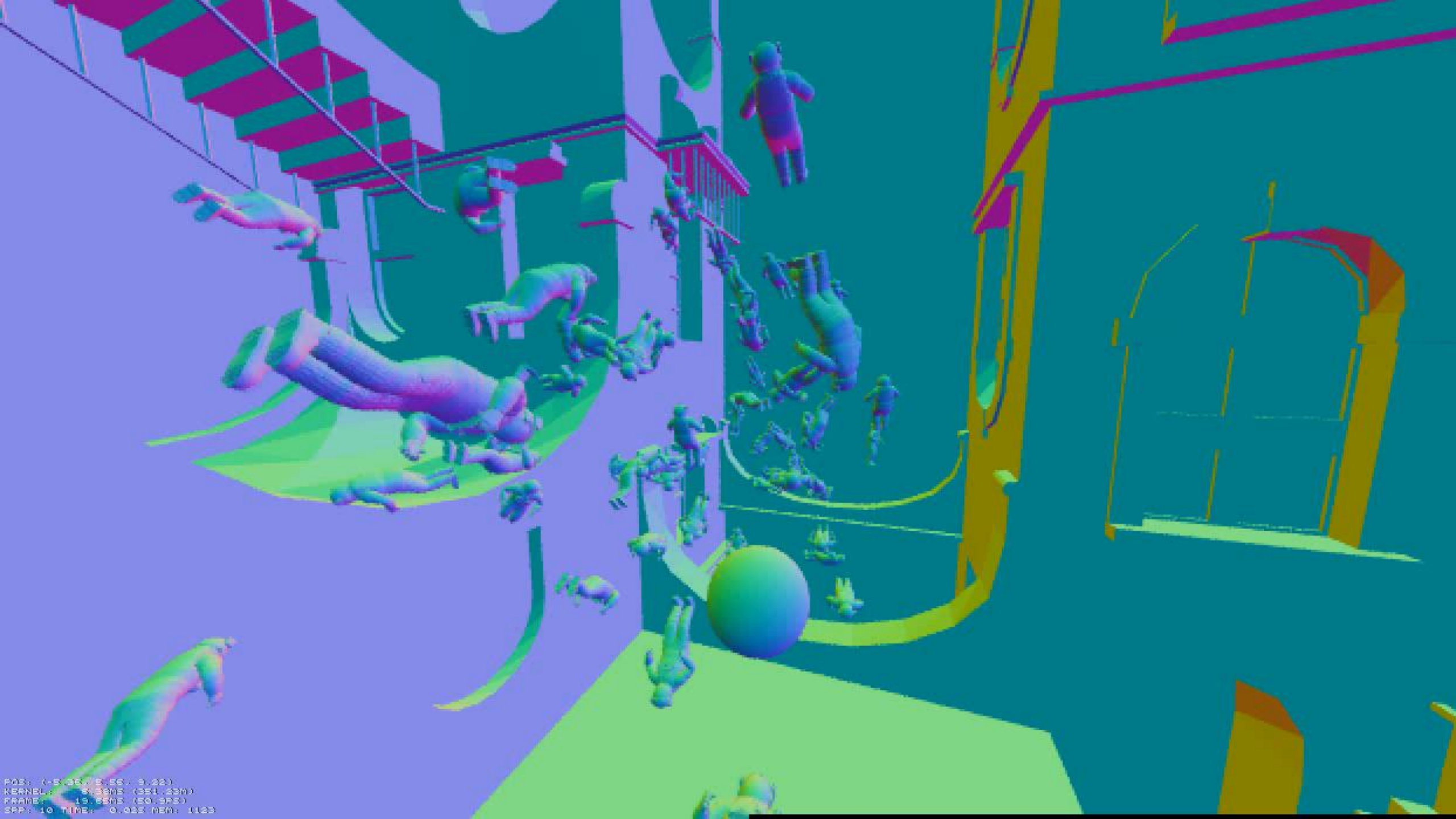
or:

2. *Transform the ray, not the node*

Rigid motion is achieved by transforming the rays by the *inverse transform* upon entering the sub-BVH.



(this obviously does not only apply to translation)


POS: (-5.36, 5.56, 9.22)
KERNEL: 8.38MS (351.23M)
FRAME: 19.68MS (50.9FS)
SPP: 10 TIME: 0.02S MEM: 1123

Agenda:

- Combining BVHs
- Refitting
- Rigid Motion

End of PART 6.