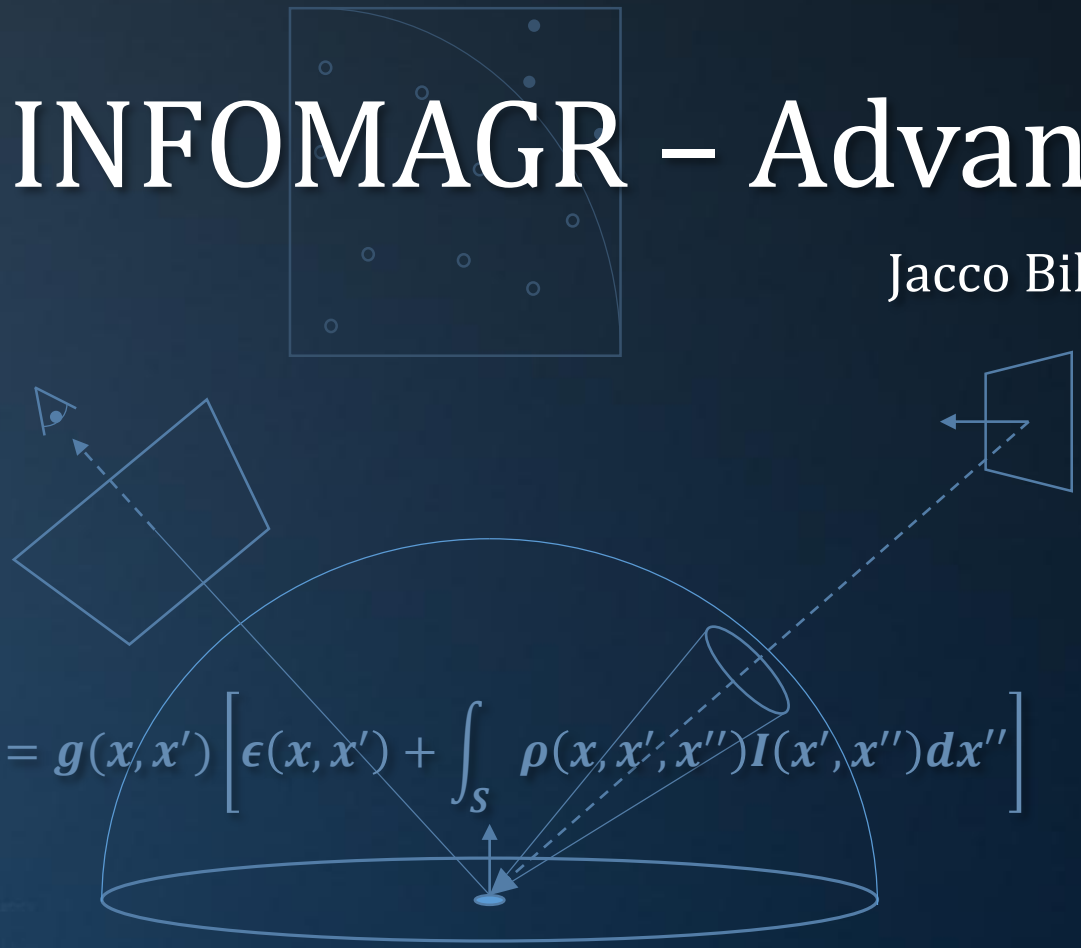
# INFOMAGR – Advanced Graphics

Jacco Bikker - February – April 2016

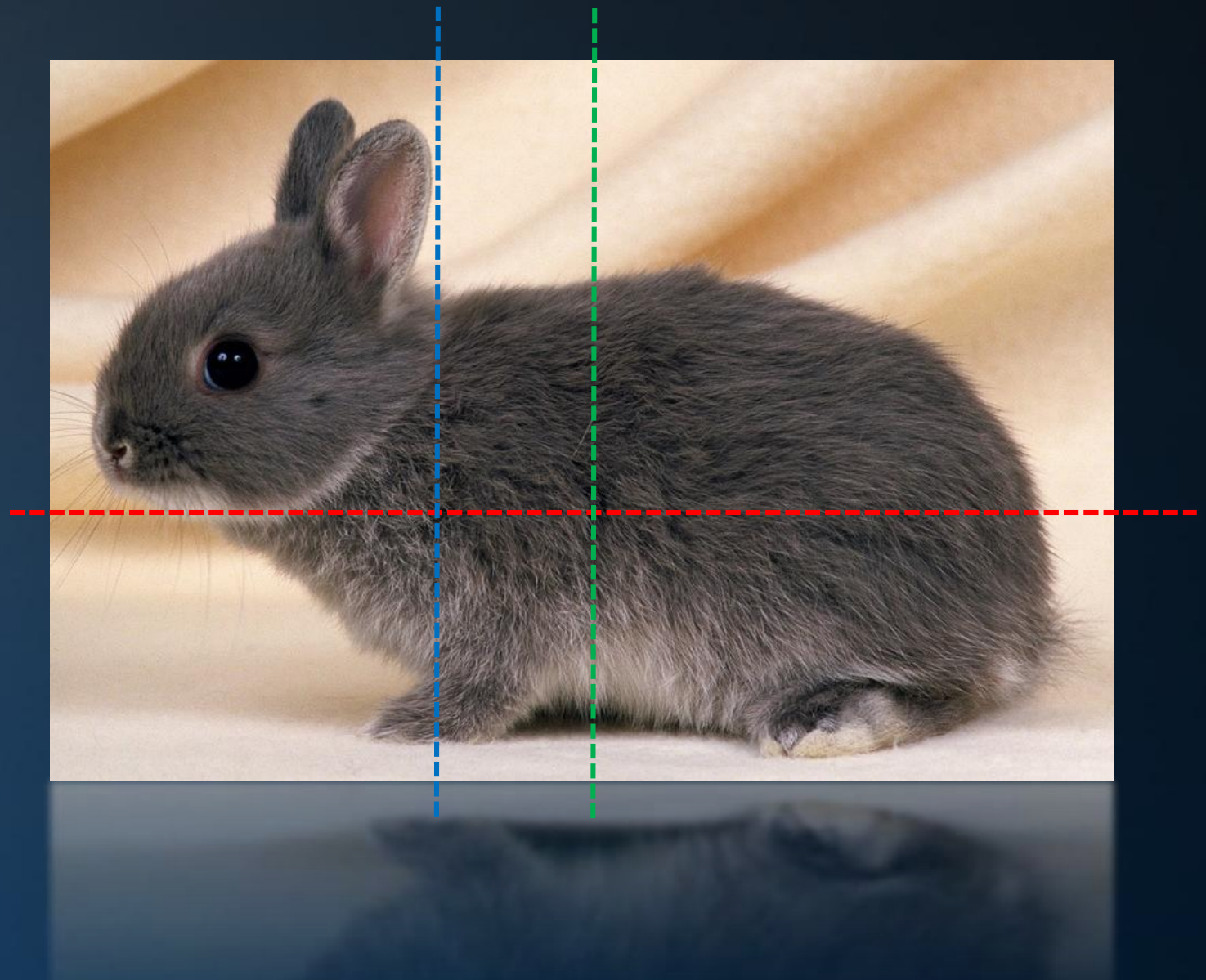



# Welcome!

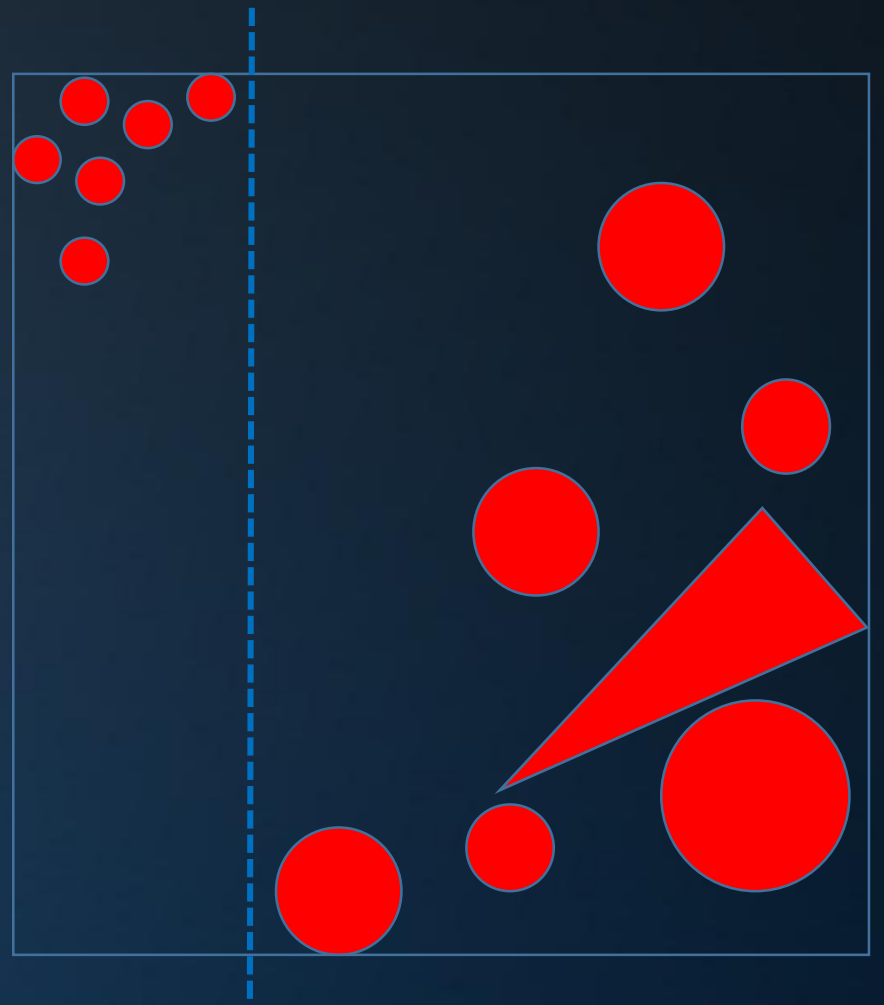
# Today’s Agenda:

- Building Better BVHs
- The Problem of Large Polygons
- Refitting
- Fast BVH Construction

# Better BVHs



# Better BVHs



# Better BVHs

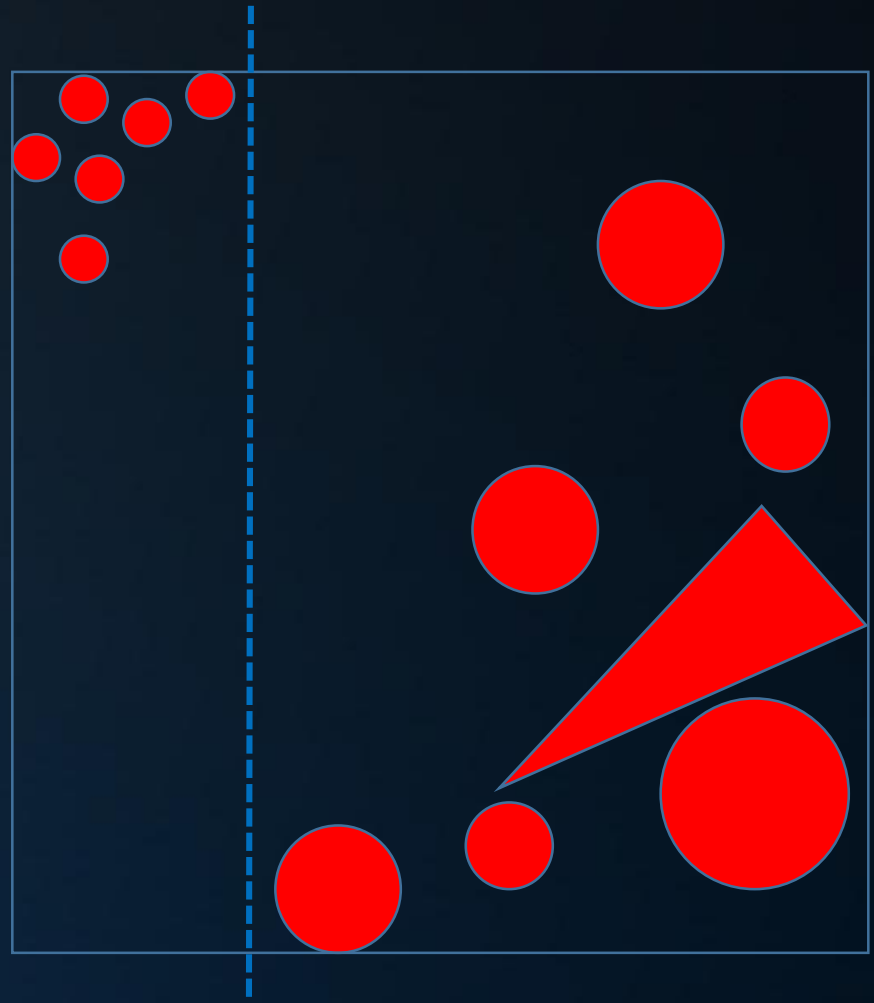
What Are We Trying To Solve?

A BVH is used to reduce the number of ray/primitive intersections.

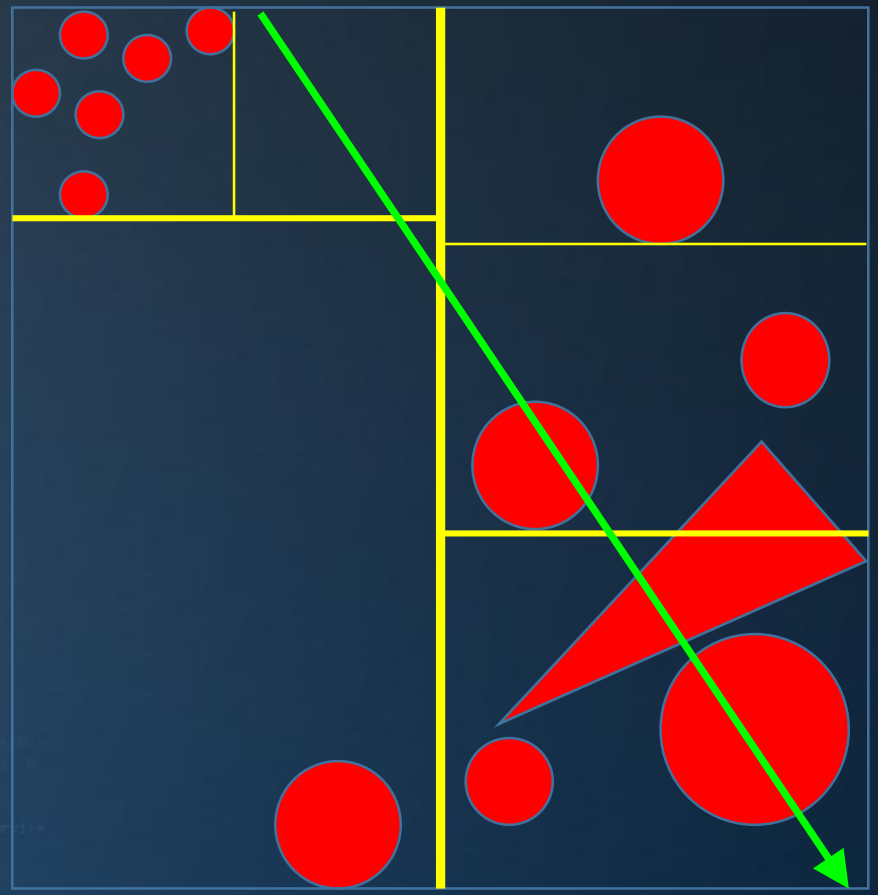
But: it introduces new intersections.

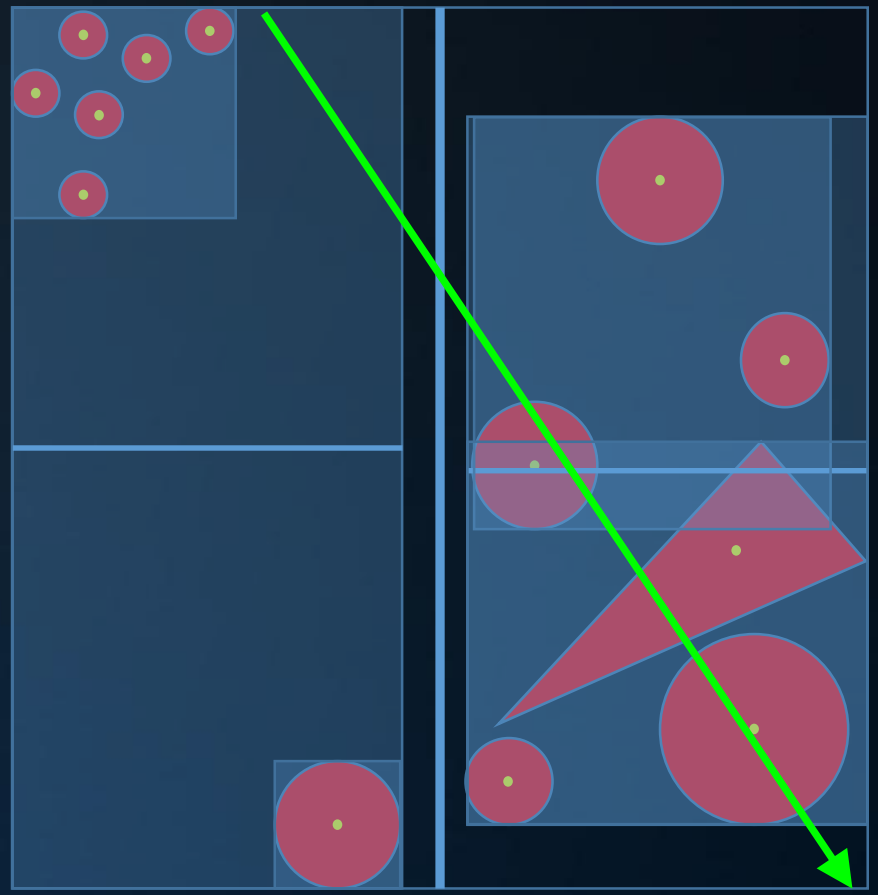
The ideal BVH minimizes:

- # of ray / primitive intersections
- # of ray / node intersections.



# Better BVHs





# Better BVHs

BVH versus kD-tree

The BVH better encapsulates geometry.

➔ This reduces the chance of a ray hitting a node.

➔ This is all about probabilities!

*What is the probability of a ray hitting a random triangle?*

*What is the probability of a ray hitting a random node?*

This probability is proportional to **surface area**.

# Better BVHs

## Optimal Split Plane Position

The ideal split minimizes the cost of a ray intersecting the resulting nodes.

This cost depends on:

- Number of primitives that will have to be intersected
- Probability of this happening

The cost of a split is thus:

$$A_{left} * N_{left} + A_{right} * N_{right}$$

# Better BVHs

## Optimal Split Plane Position

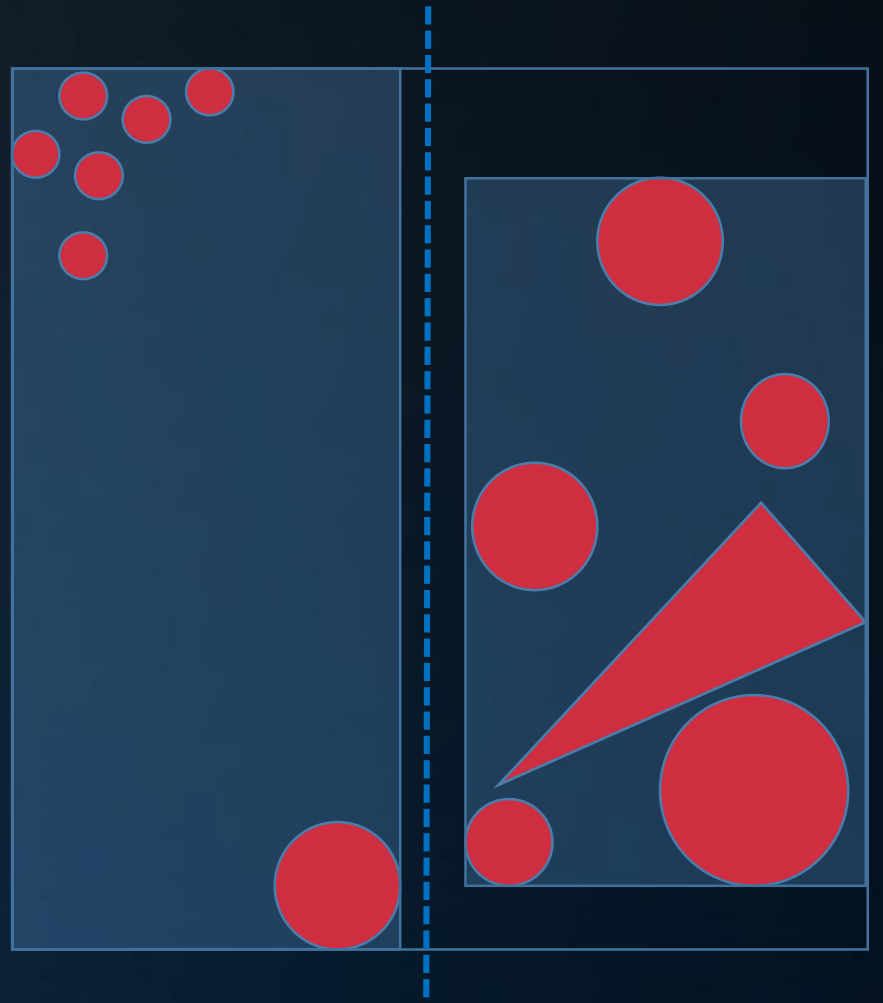
The ideal split minimizes the cost of a ray intersecting the resulting nodes.

This cost depends on:

- Number of primitives that will have to be intersected
- Probability of this happening

The cost of a split is thus:

$$A_{left} * N_{left} + A_{right} * N_{right}$$



# Better BVHs

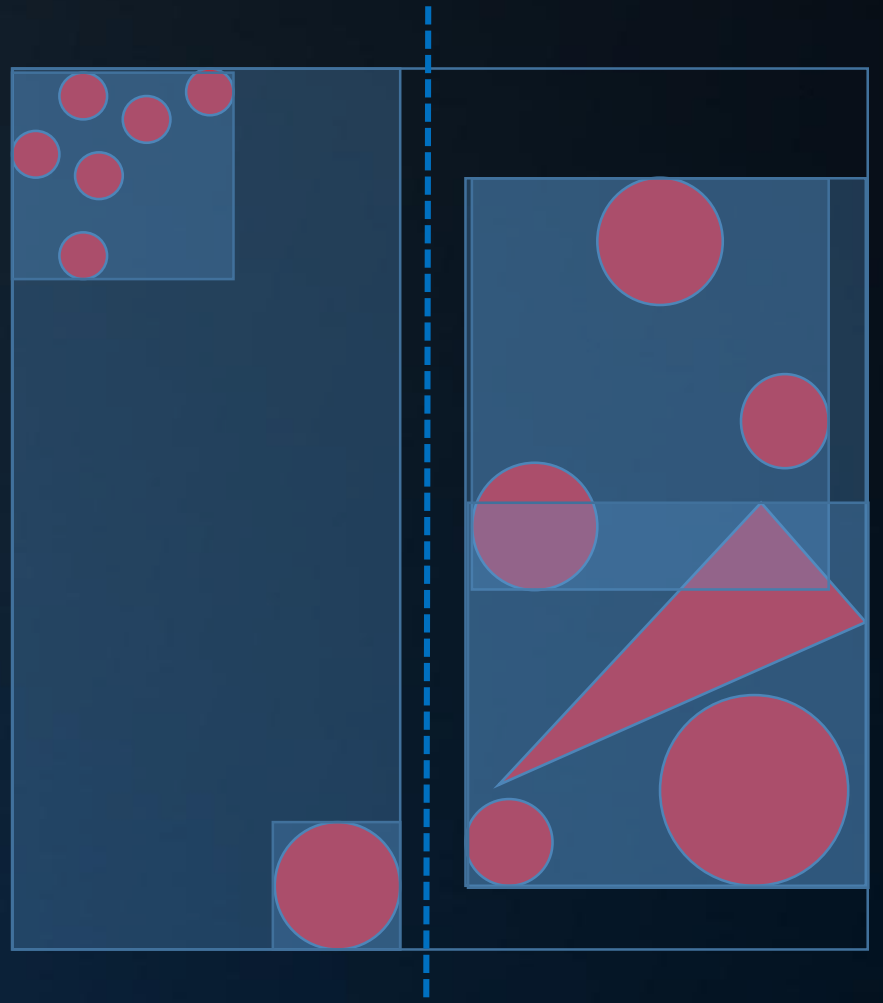
Optimal Split Plane Position

Or, more concisely:

$$A^0_{left} * \left(A^1_{left} * N^1_{left} + A^1_{right} * N^1_{right}\right)$$

$$+$$

$$A^0_{right} * \left(A^2_{left} * N^2_{left} + A^2_{right} * N^2_{right}\right)$$



# Better BVHs

Optimal Split Plane Position

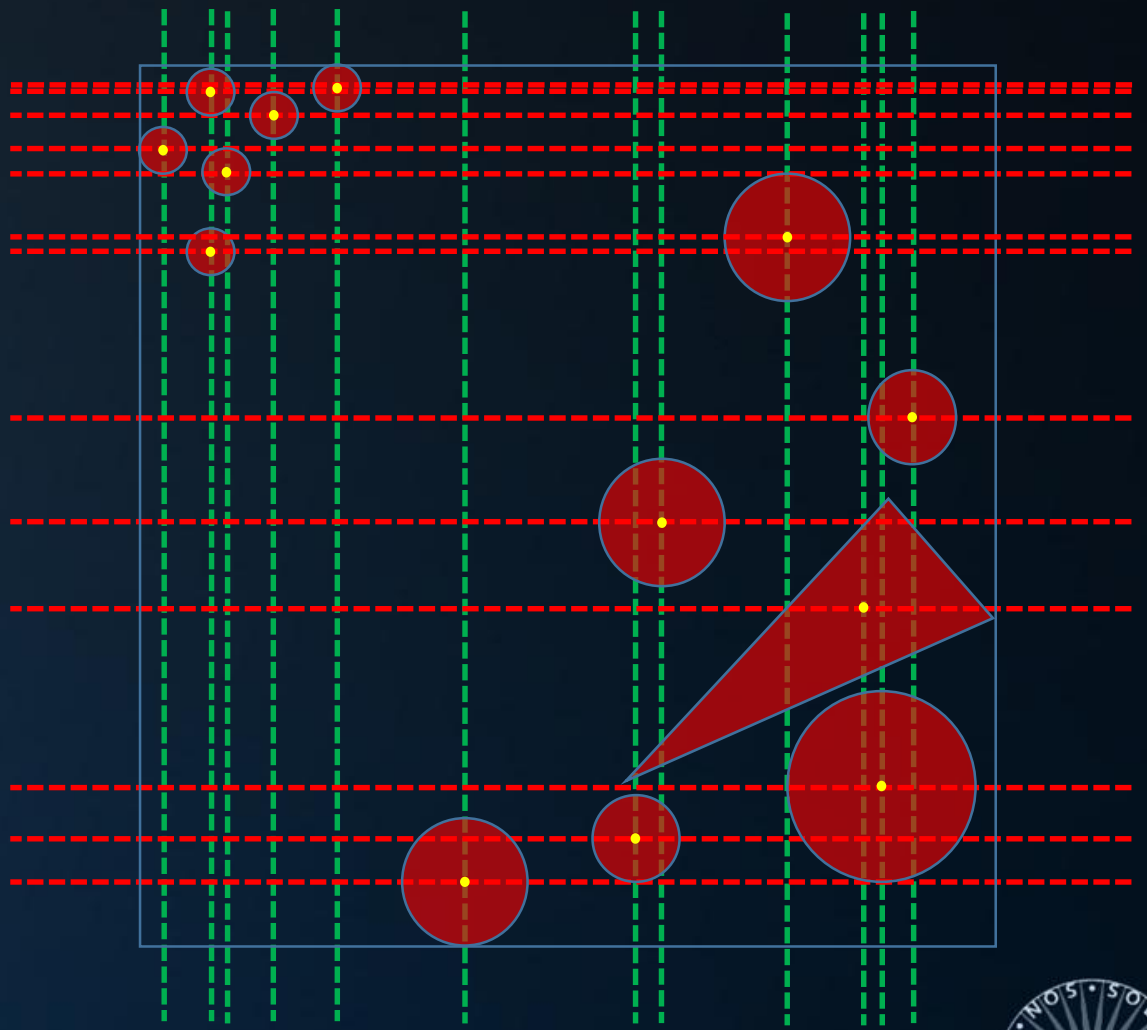
Which positions do we consider?

*Object subdivision may happen over x, y or z axis.*

*The cost function is constant between primitive centroids.*

➔ For N primitives: $3(N-1)$ possible locations

➔ For a 2-level tree: $(3(N-1))^2$ configurations



# Better BVHs

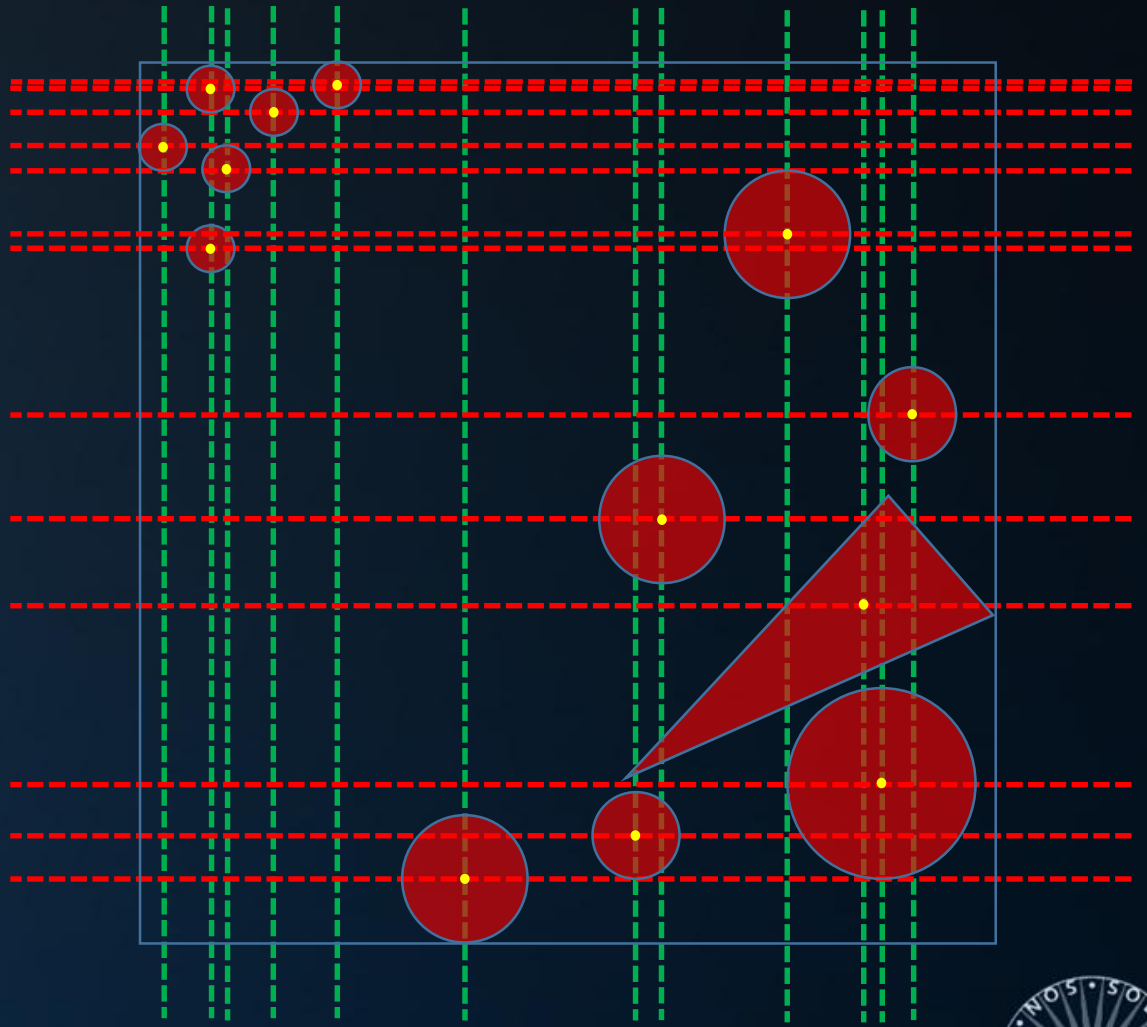
SAH and Termination

A split is ‘not worth it’ if it doesn’t yield a cost lower than the cost of the parent node, i.e.:

$$A_{left} * N_{left} + A_{right} * N_{right} \geq A * N$$

This provides us with a natural and optimal termination criterion.

(and it solves the problem of the Bad Artist)



# Better BVHs

Optimal Split Plane Position

Evaluating $(3(N-1))^2$ configurations?

Solution: apply the *surface area heuristic* (SAH) lazily*.

*: Heuristics for Ray Tracing using Space Subdivision, MacDonald & Booth, 1990.

# Better BVHs

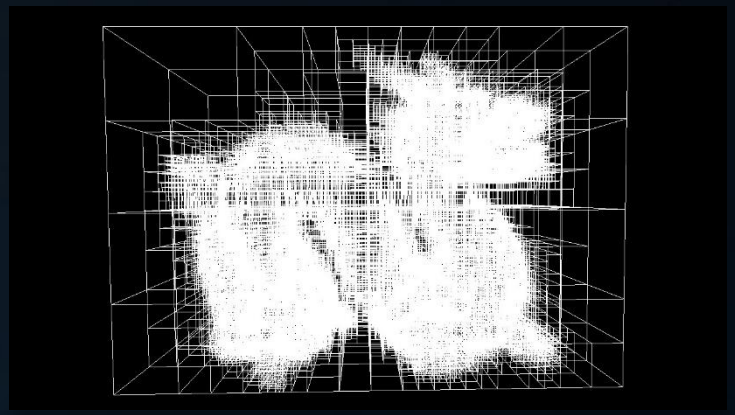
Optimal Split Plane Position

Comparing naïve versus SAH:

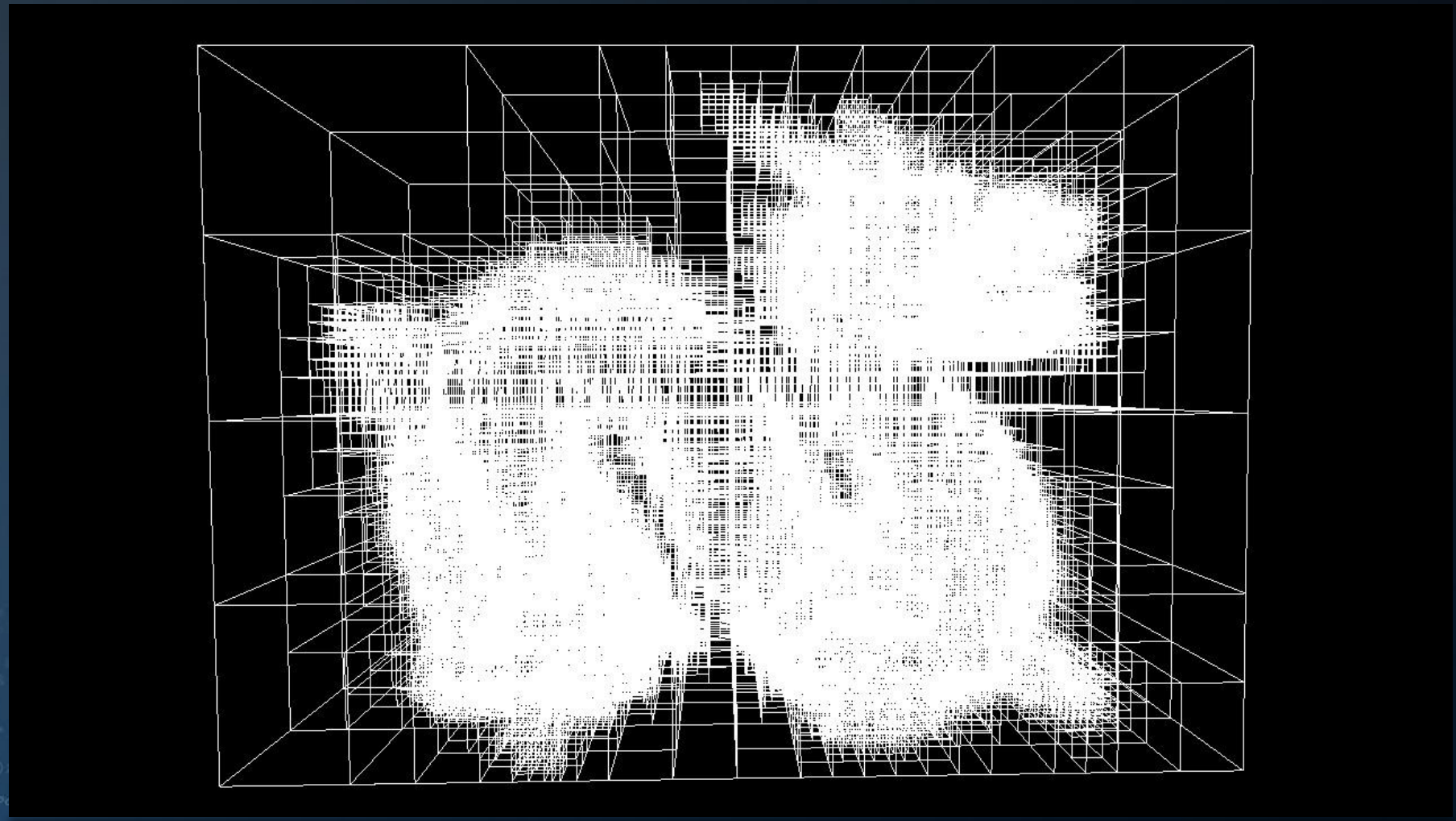
- SAH will cut #intersections in half;
- expect ~2x better performance.

SAH & kD-trees:

- Same scheme applies.

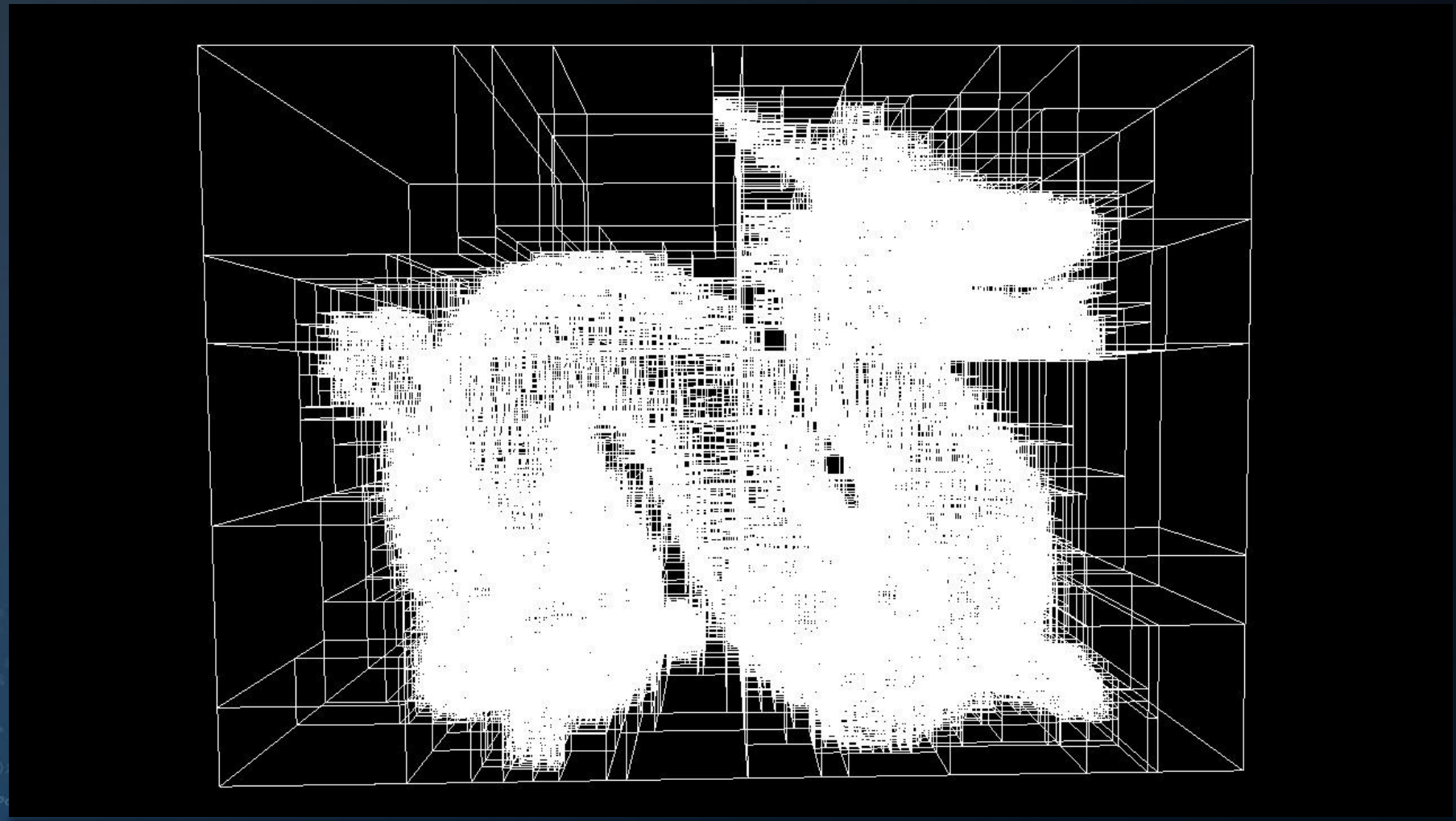


# Better BVHs



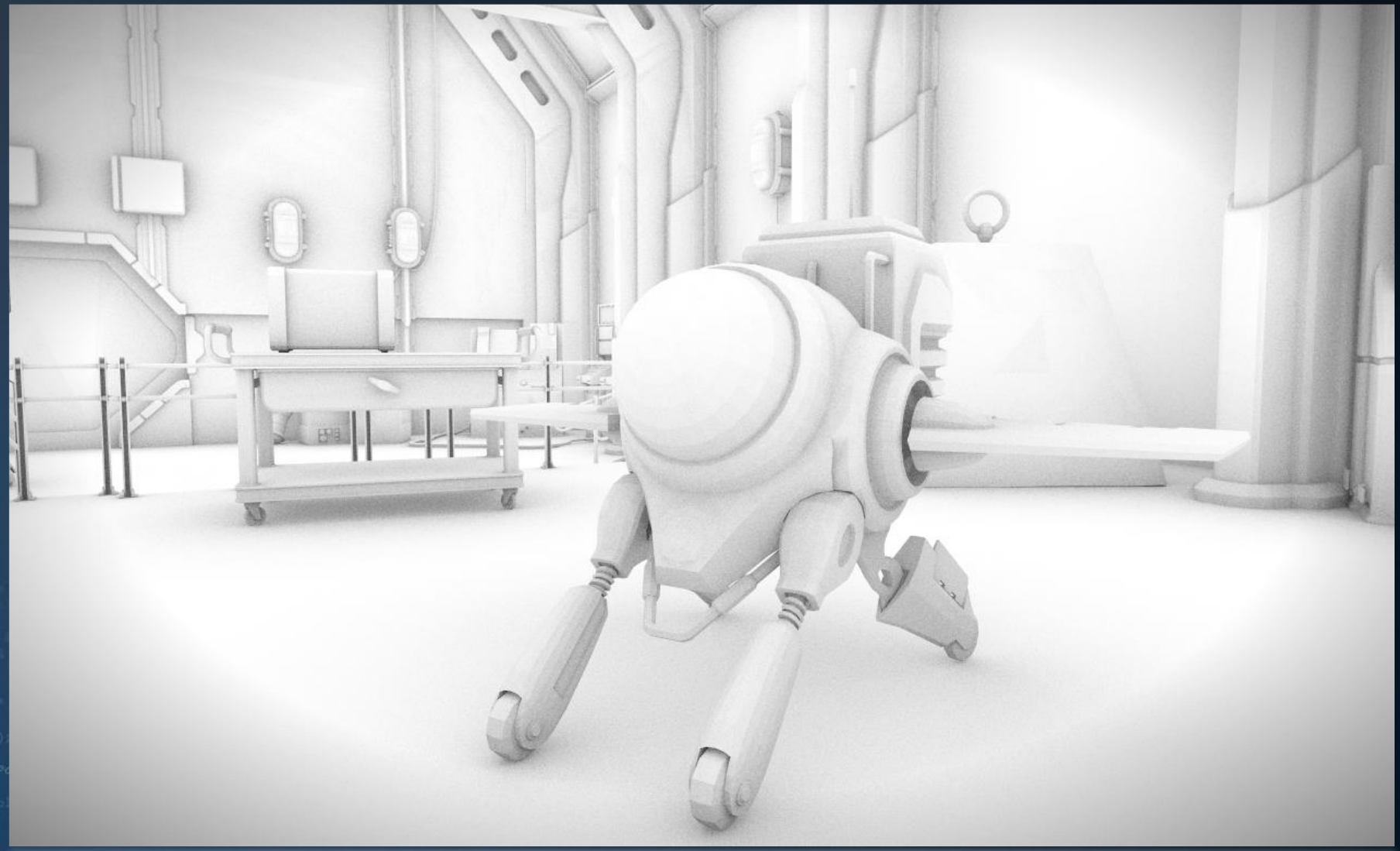
**Median Split**

# Better BVHs

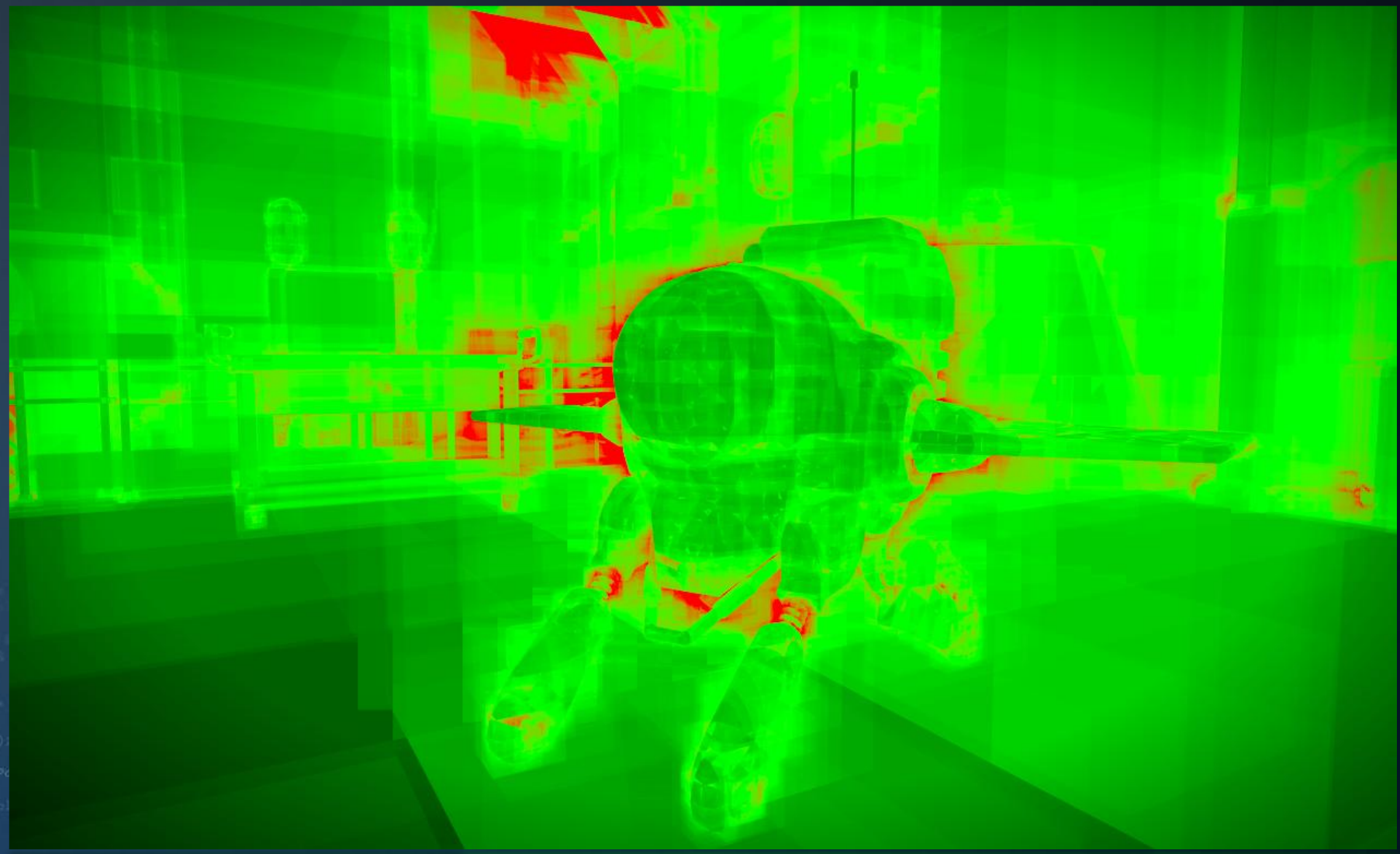


**Surface Area Heuristic**

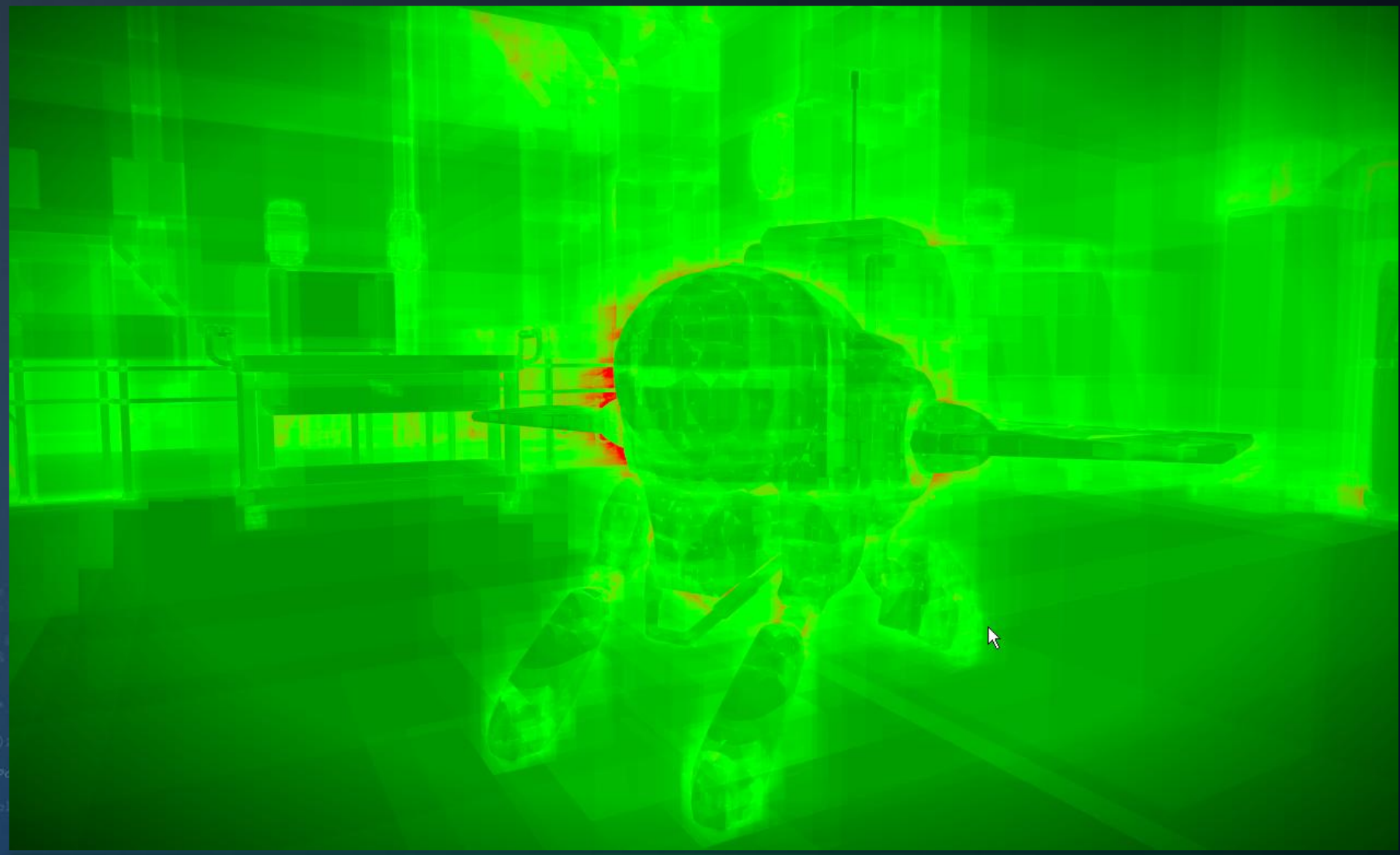
# Better BVHs



# Better BVHs



# Better BVHs



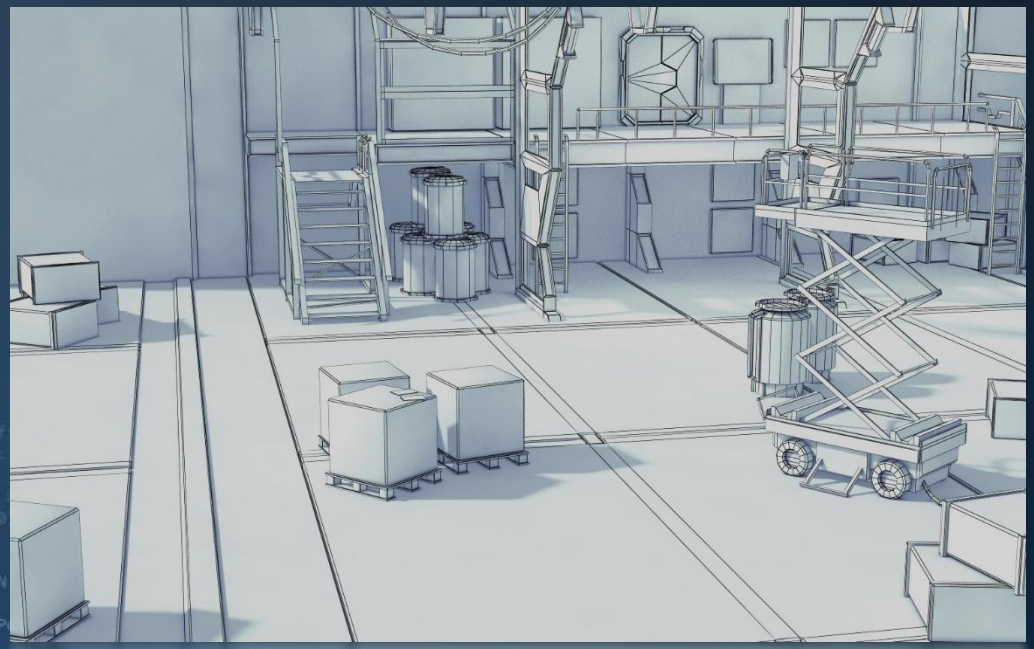
# Today’s Agenda:

- Building Better BVHs
- The Problem of Large Polygons
- Refitting
- Fast BVH Construction

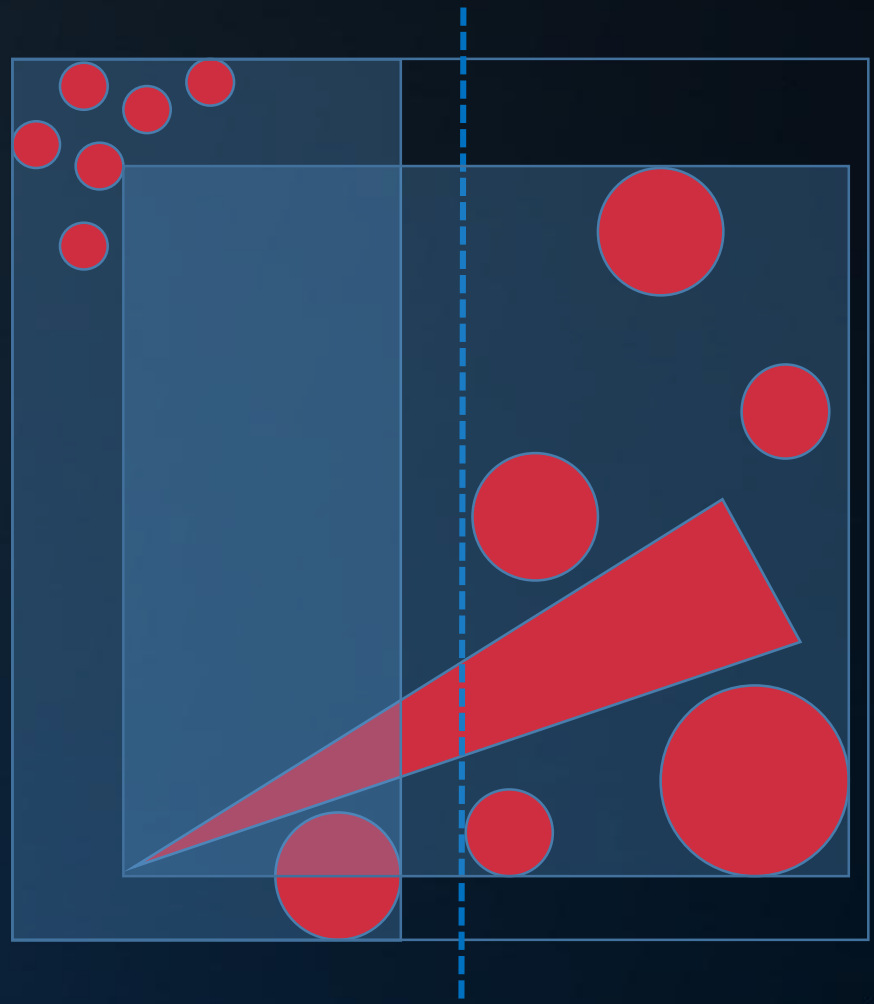
# Splitting

Problematic Large Polygons

Large polygons lead to poor BVHs.



(far more common than you'd think)



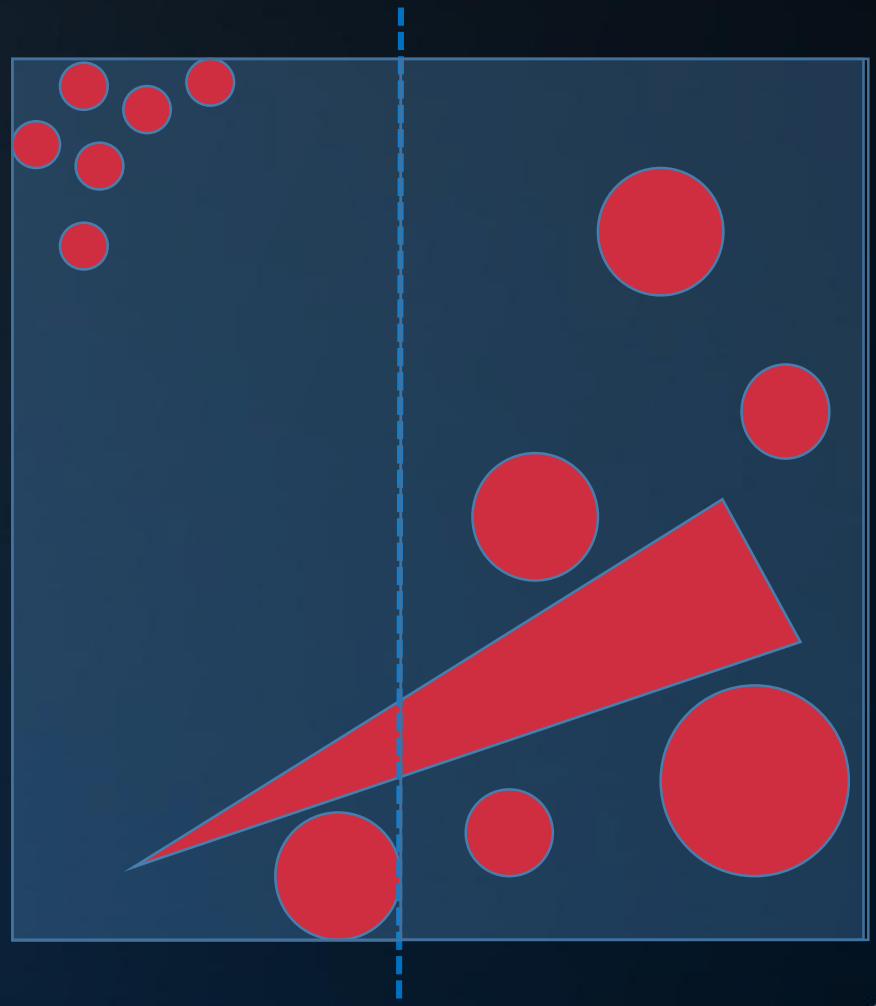
# Splitting

## Problematic Large Polygons

Large polygons lead to poor BVHs.

Using the spatial splits in kD-trees, this is far less of an issue:

The triangle will simply be assigned to each subspace.



# Splitting

## Problematic Large Polygons

Large polygons lead to poor BVHs.

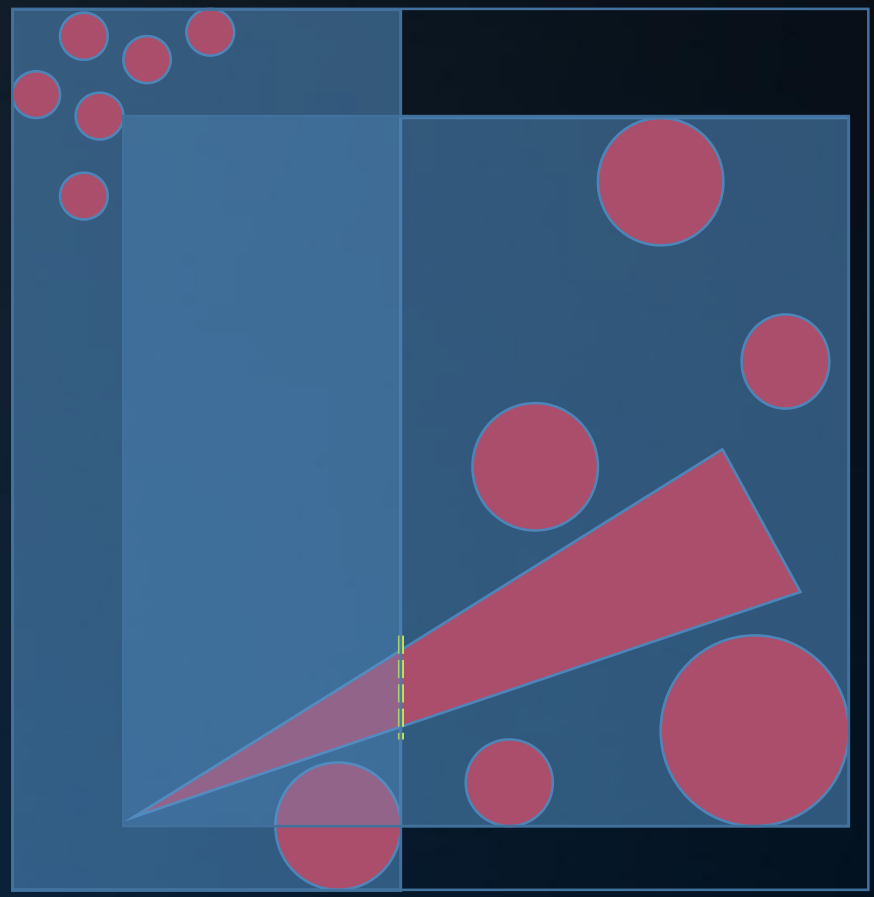
Using the spatial splits in kD-trees, this is far less of an issue:

The triangle will simply be assigned to each subspace.

Solution 1: split large polygons*.

Observations:

1. A polygon can safely reside in multiple leafs;
2. The bounds of a leaf do not have to include the entire polygon.



*: Early Split Clipping for Bounding Volume Hierarchies, Ernst & Greiner, 2007

# Splitting

## Early Split Clipping

Observations:

1. A polygon can safely reside in multiple leafs;
2. The bounds of a leaf do not have to include the entire polygon.
3. BVH construction only uses primitive bounding boxes.

Algorithm:

Prior to BVH construction, we recursively subdivide any polygon with a surface area that exceeds a certain threshold.

## Issues:

- Threshold parameter
- Individual polygons are split, regardless of surrounding geometry
- Primitives may end up multiple times in the same leaf

(some of these issues are resolved in: The Edge Volume Heuristic - Robust Triangle Subdivision for Improved BVH Performance, Dammertz & Keller, 2008)

# Splitting

Spatial Splits for BVHs

$$C_{split} = A_{left} * N_{left} + A_{right} * N_{right} < A * N$$

Observation: spatial splits are not limited to kD-trees.

But: spatial splits tend to increase the cost of a split.

Idea:

1. Determine cost of optimal object partition;
2. Determine cost of optimal spatial split;
3. Apply spatial split if cost is lower than object partition*.

*: Spatial Splits in Bounding Volume Hierarchies, Stich et al., 2009

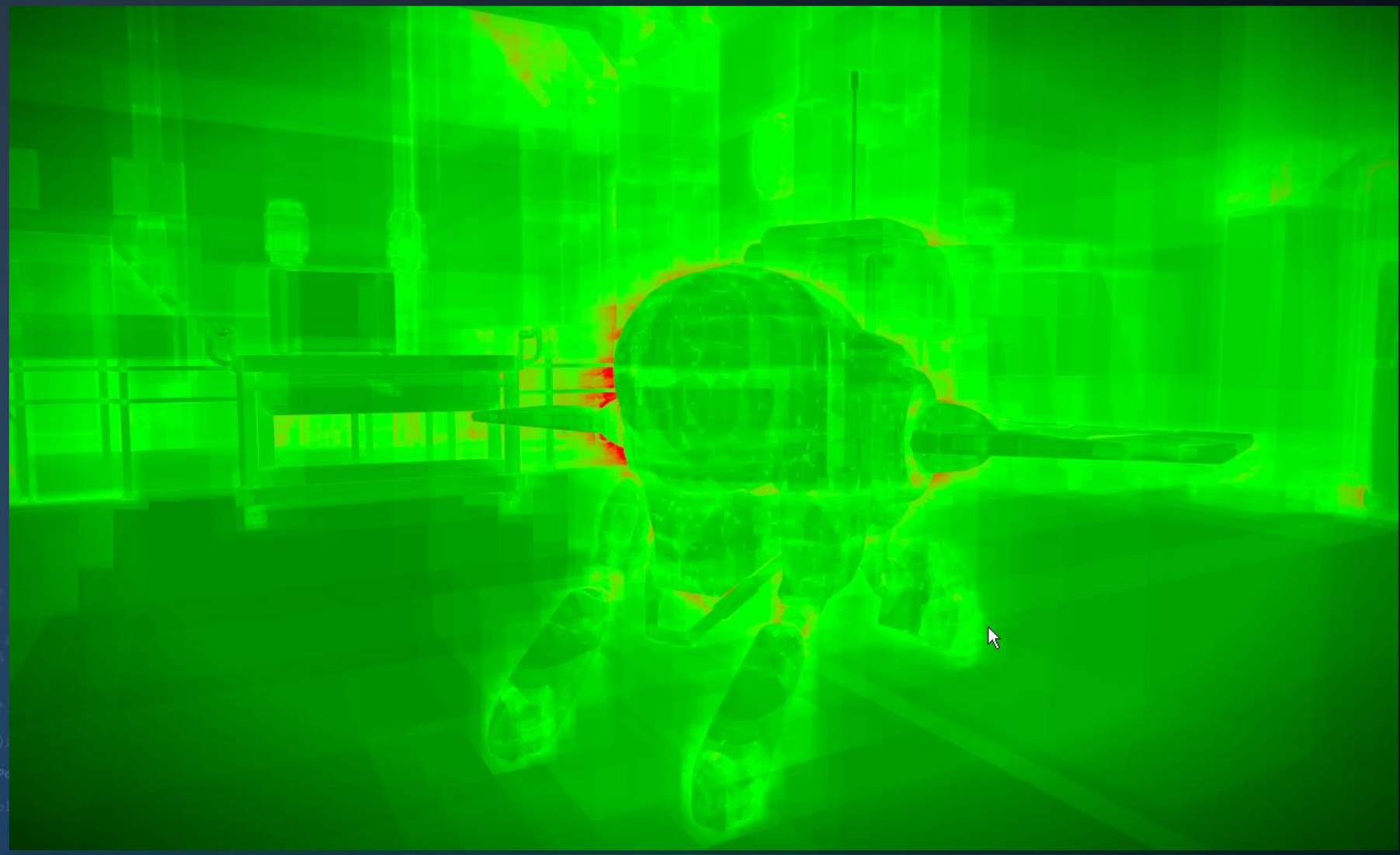
# Splitting

State of the Art: SBVH

Summary: high quality bounding volume hierarchies can be obtained by combining the surface area heuristic and spatial splits.

Compared to a regular SAH BVH, spatial splits improve the BVH by ~25% (see paper for scenes and figures).

# Better BVHs



# Better BVHs

# Today’s Agenda:

- Building Better BVHs
- The Problem of Large Polygons
- Refitting
- Fast BVH Construction

# Refitting

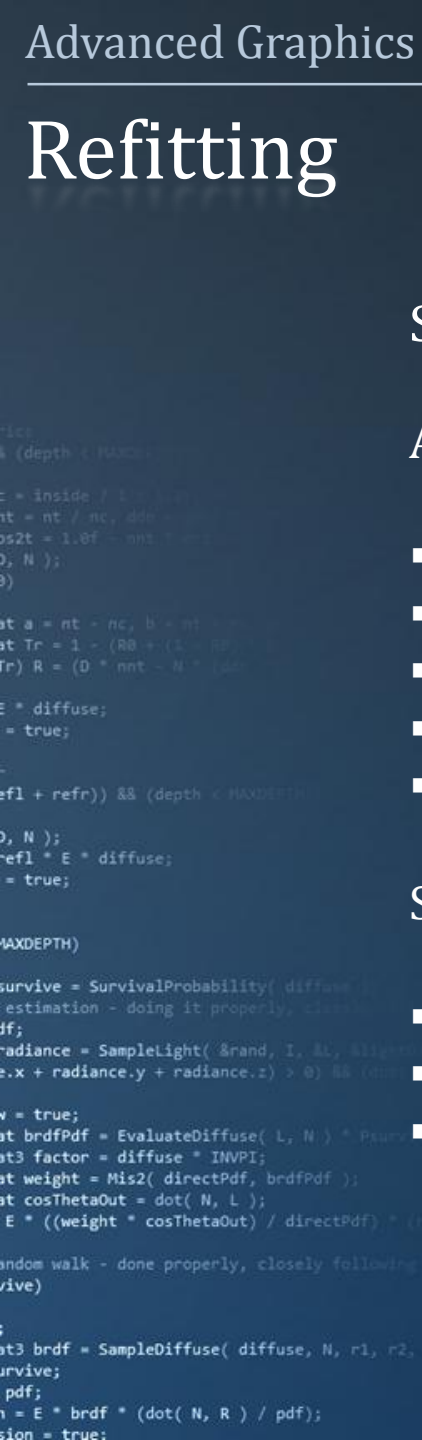
Summary of BVH Characteristics

A BVH provides significant freedom compared to e.g. a kD-tree:

- No need for a 1-to-1 relation between bounding boxes and primitives
- Primitives may reside in multiple leafs
- 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

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.*

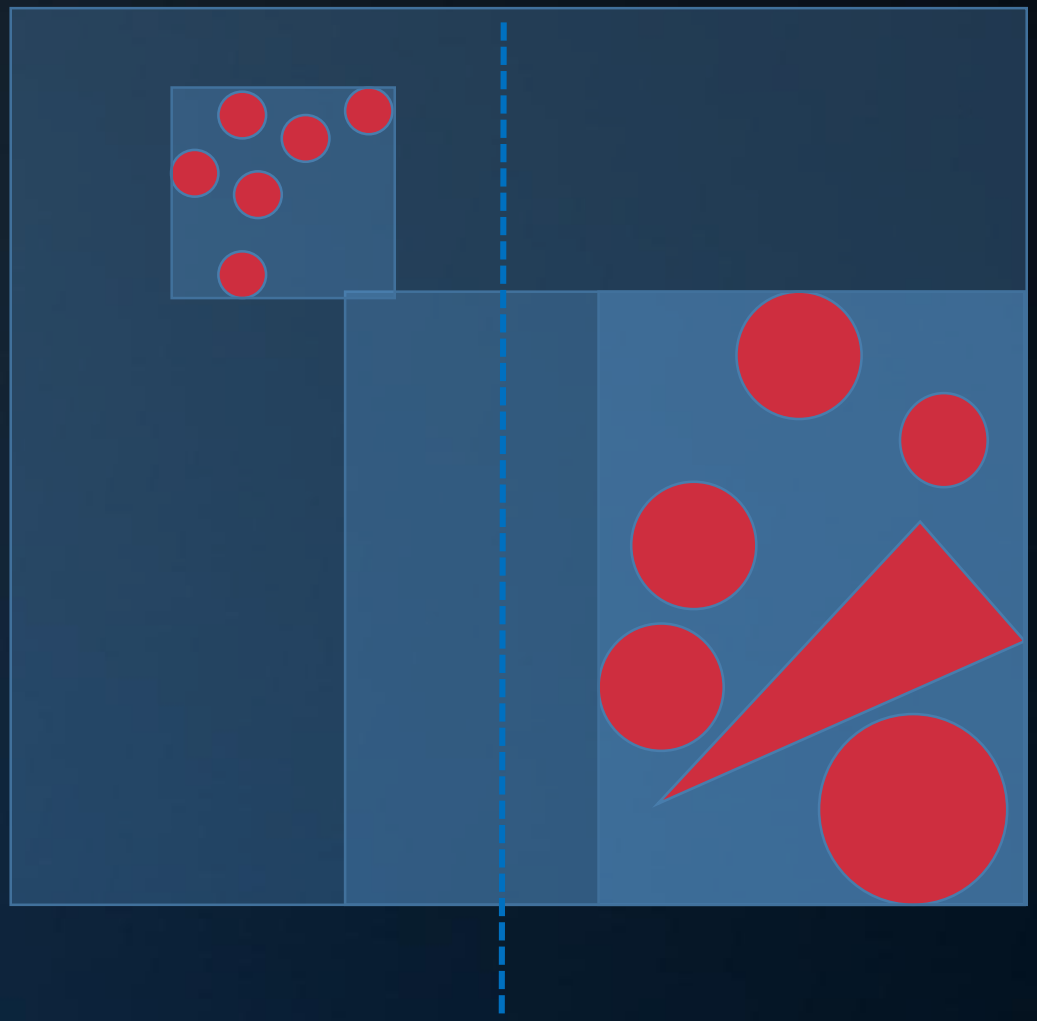
# Refitting

## Refitting

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

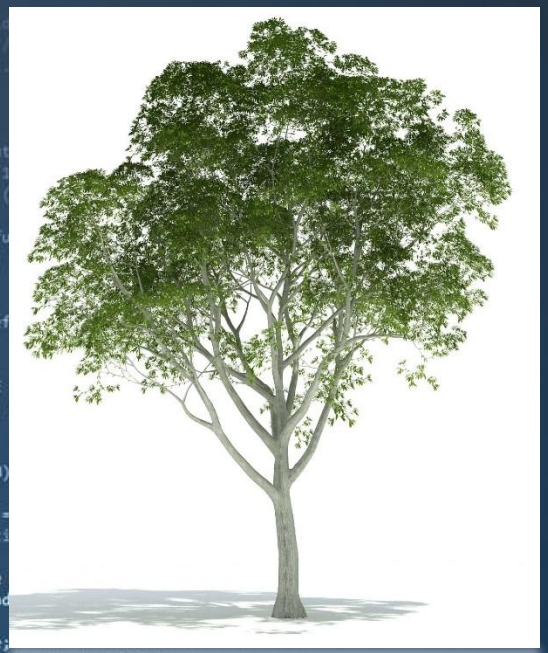
Algorithm:

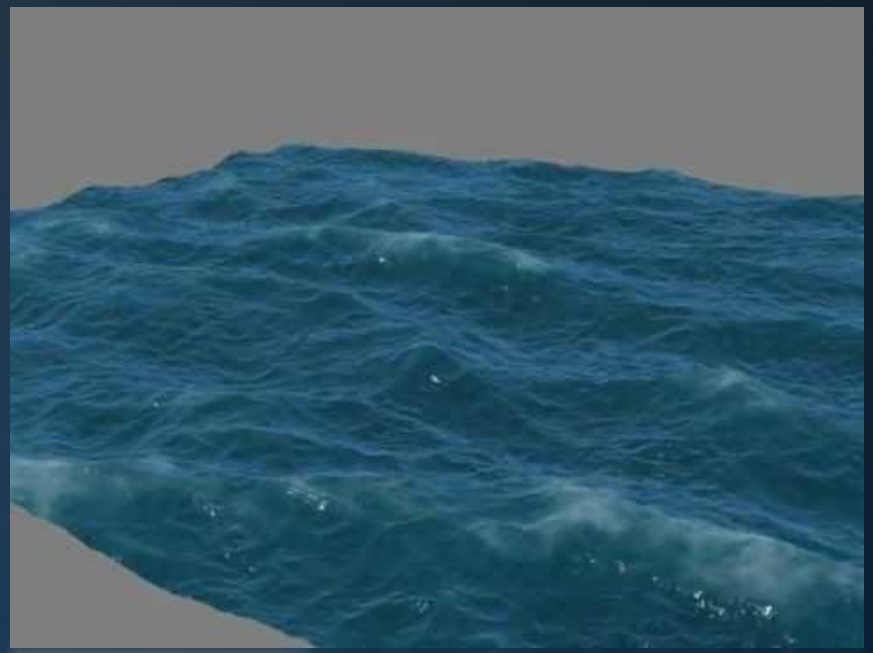
1. For each leaf, calculate the bounds over the primitives it represents
2. Update parent bounds

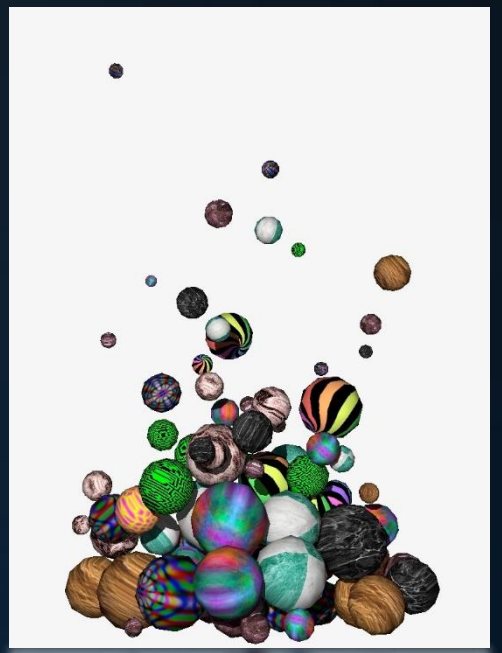


# Refitting

## Refitting - Suitability

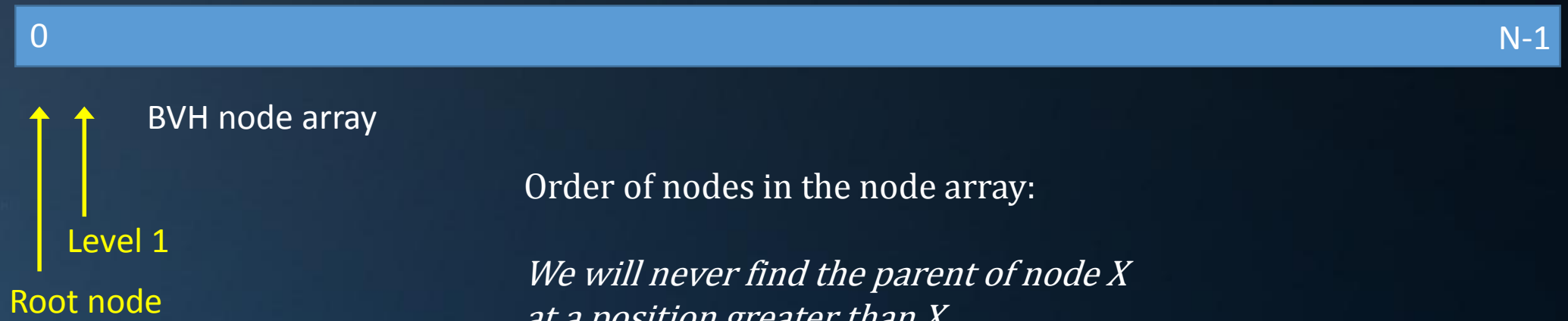






# Refitting

## 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();
```

# Today’s Agenda:

- Building Better BVHs
- The Problem of Large Polygons
- Refitting
- Fast BVH Construction

# Binning

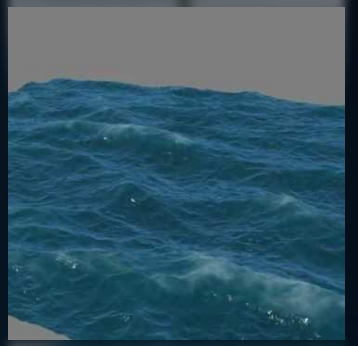
Rapid BVH Construction

Refitting allows us to update hundreds of thousands of primitives in real-time. But what if topology changes significantly?

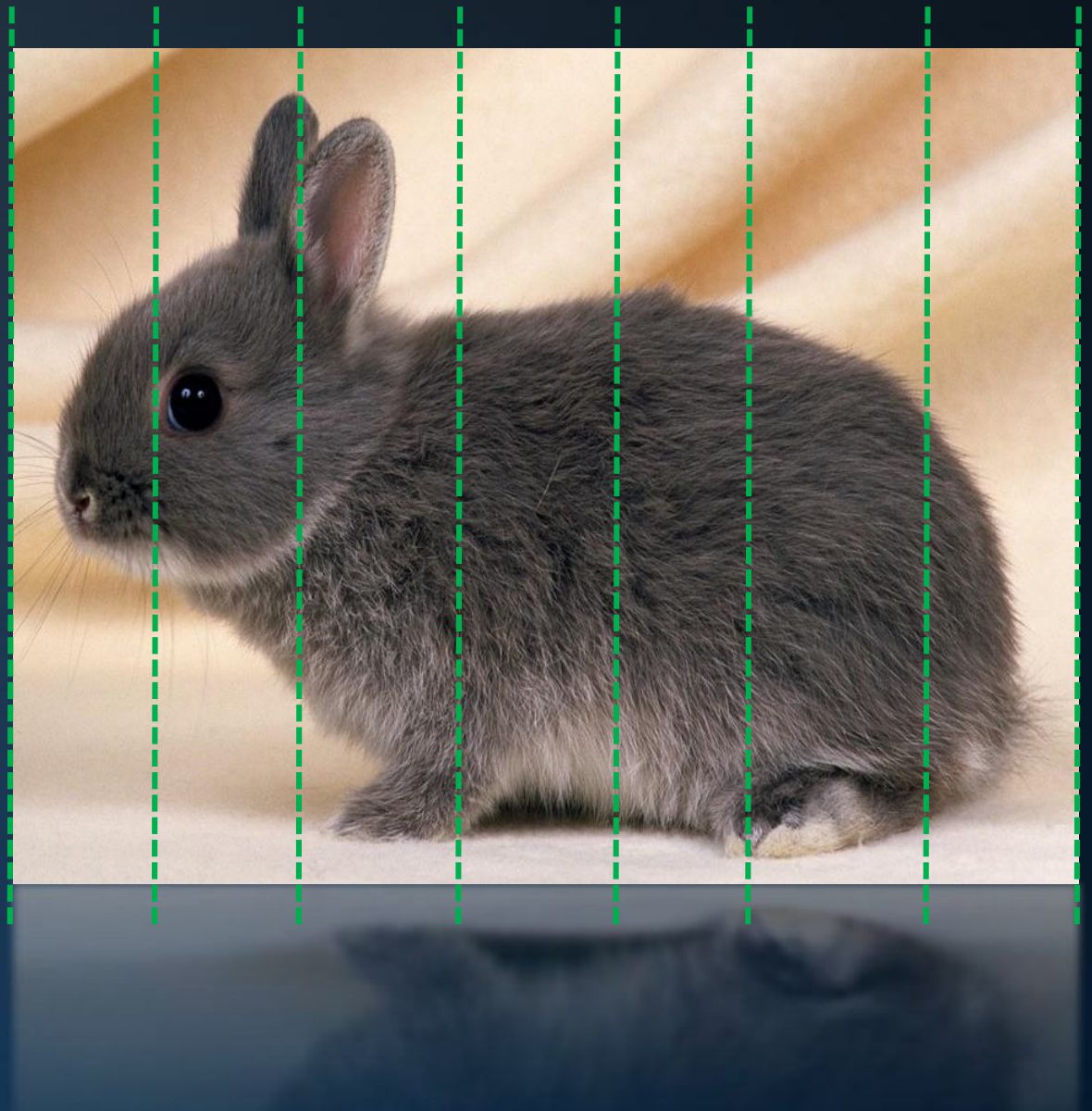
Rebuilding a BVH requires $3NlogN$ split plane evaluations.

Options:

1. Do not use SAH (significantly lower quality BVH)
2. Do not evaluate all 3 axes (minor degradation of BVH quality)
3. Make split plane selection independent of $N$



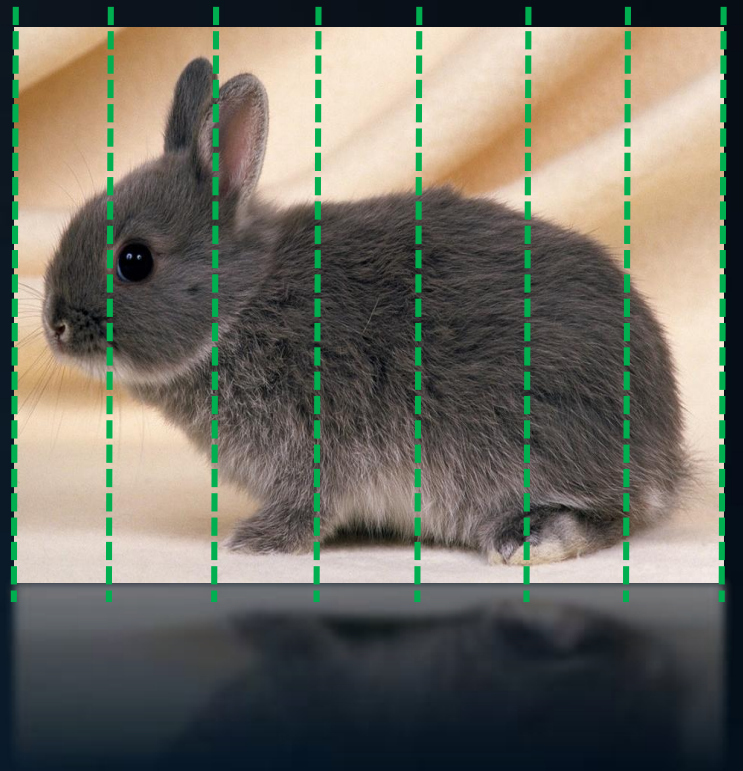
# Binning



# Binning

Binned BVH Construction*

Binned construction:
*Evaluate SAH at N discrete intervals.*



*: On fast Construction of SAH-based Bounding Volume Hierarchies, Wald, 2007

# Binning

## Binned BVH Construction

Detailed algorithm:

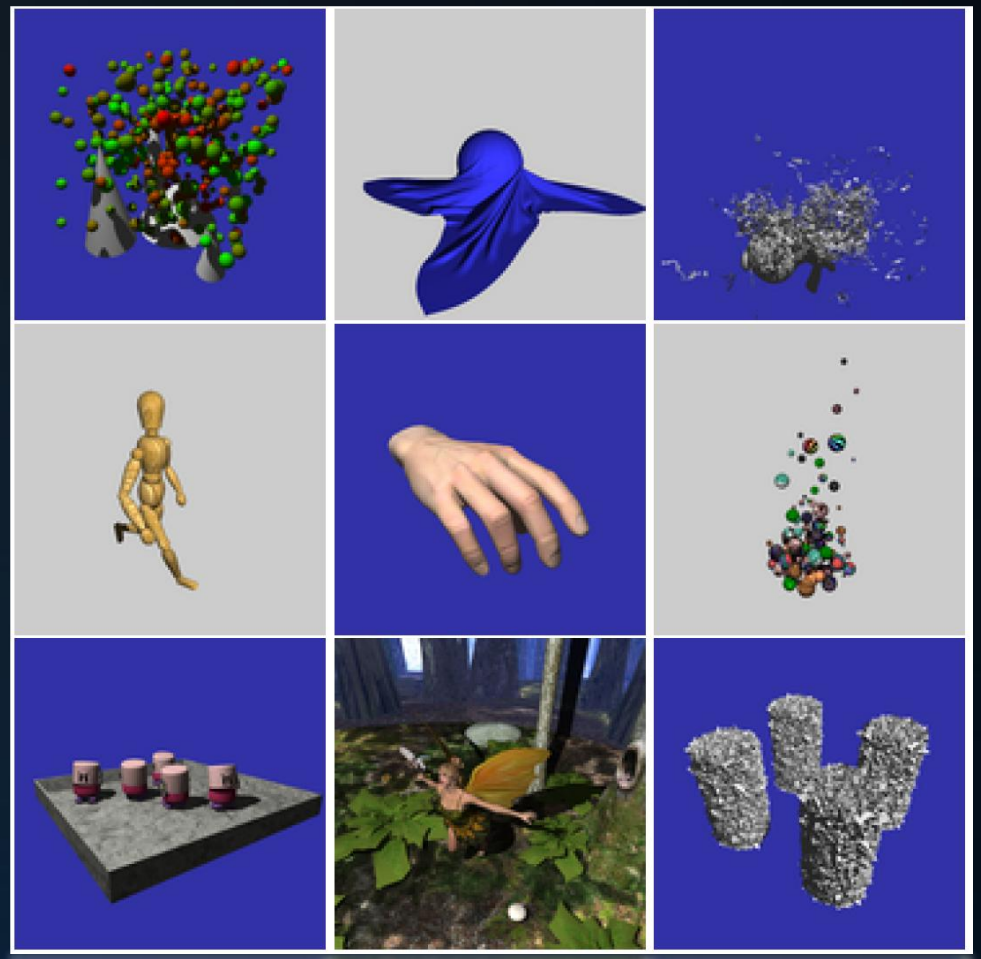
1. Calculate spatial bounds
2. Calculate object centroid bounds
3. Calculate intervals (efficiently and accurately!)
4. Populate bins
5. Sweep: evaluate cost, keep track of counts
6. Use best position

# Binning

Binned BVH Construction

Performance evaluation:

1.9s for 10M triangles (8 cores @ 2.6Ghz).



# Binning

# INFOMAGR – Advanced Graphics

Jacco Bikker - February – April 2016

# END of “The Perfect BVH”

next lecture: “Real-time Ray Tracing”