How do I test if a given BSP tree is optimal?

Asked 16 years, 2 months ago Modified 12 years ago Viewed 2k times

4

I have a polygon soup of triangles that I would like to construct a BSP tree for. My current program simply constructs a BSP tree by inserting a random triangle from the model one at a time until all the triangles are consumed, then it checks the depth and breadth of the tree and remembers the best score it achieved (lowest depth, lowest breadth).

By definition, the best depth would be log2(n) (or less if co-planar triangles are grouped?) where n is the number of triangles in my model, and the best breadth would be n (meaning no splitting has occurred). But, there are certain configurations of triangles for which this pinnacle would never be reached.

Is there an efficient test for checking the quality of my BSP tree? Specifically, I'm trying to find a way for my program to know it should stop looking for a more optimal construction.

algorithm 3d bsp-tree

Share edited Oct 17, 2012 at 13:52

Improve this question

Follow

tomdemuyt

4,582 ● 2 ● 34 ● 61

asked Oct 2, 2008 at 16:09

Martin

6,015 ● 7 ● 51 ● 79

3 Answers

Sorted by: Highest score (default)

3

Construction of an optimal tree is an NP-complete problem. Determining if a given tree is optimal is essentially the same problem.

From this BSP faq:

> The problem is one of splitting versus tree balancing. These are mutually exclusive requirements. You should choose your strategy for building a good tree based on how you intend to use the tree.

Share Improve this answer

Follow

answered Oct 2, 2008 at 16:24

Mitch Wheat

300k ● 44 ● 477 ● 550

In general, testing if a solution is optimal and finding an optimal solution do not have to belong to the same complexity class. Do you maybe know some reference that checking if a BSP tree is optimal is NP-complete? – axel22 Jan 18, 2013 at 9:35

2

Randomly building BSP trees until you chance upon a good one will be really, really inefficient.

Instead of choosing a tri at random to use as a split-plane, you want to try out several (maybe all of them, or maybe a random sampling) and pick one according to some heuristic. The heuristic is typically based on (a) how balanced the resulting child nodes would be, and (b) how many tris it would split.

You can trade off performance and quality by considering a smaller or larger sampling of tris as candidate split-planes.

But in the end, you can't hope to get a totally optimal tree for any real-world data so you might have to settle for 'good enough'.

Share Improve this answer Follow

answered Oct 2, 2008 at 16:20

Mike F

> Actually, even a single random BSP tree has asymptotically optimal complexity in the case of point sets. – Geoffrey Irving May 9, 2014 at 0:17

1

- Try to pick planes that (could potentially) get split by the most planes as splitting planes. Splitting planes can't be split.

- Try to pick a plane that has close to the same number of planes in front as in back.
- Try to pick a plane that doesn't cause too many splits.
- Try to pick a plane that is coplanar with a lot of other surfaces

You'll have to sample this criteria and come up with a scoring system to decide which one is most likely to be a good choice for a splitting plane. For example, the further off balance, the more score it loses. If it causes 20 splits, then penalty is -5 * 20 (for example). Choose the one that scores best. You don't have to sample every polygon, just search for a pretty good one.

Share Improve this answer

Follow

answered Dec 19, 2012 at 4:51

doug65536

6,744 ● 3 ● 44 ● 54

2 Building an optimal BSP tree is NP complete. But you can build a very good one with heuristics and partial searches for "good enough" partition planes. – doug65536 Jan 2, 2013 at 19:10