KD Tree Path Tracer Optimization

Rony Edde

- Shooting rays through a complex scene with a large number of polygons can be heavy and exponentially slow even on the GPU. Axis Aligned Bounding boxes help improve performance, hover this still doesn't solve the problem when rays intersect a complex object with millions of triangles.

KD Tree Path Tracer Optimization

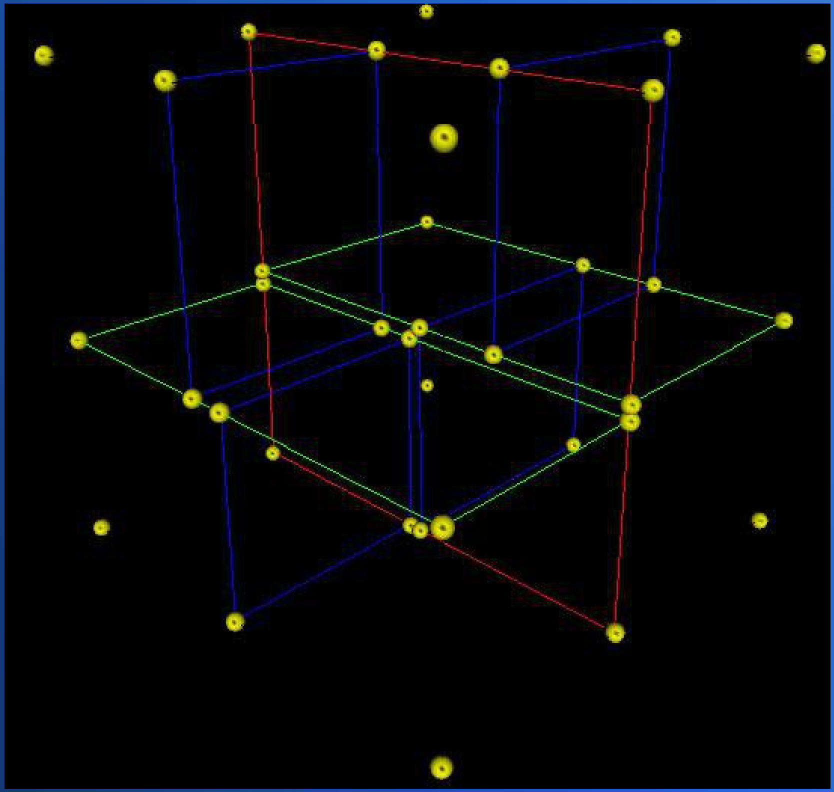
Rony Edde

- Using a K-D Tree to represent the triangles reduces the overall complexity of the lookup and the time needed to find the intersecting triangles. There is a cost of computation for generating the KD-Tree but in most cases the scene is mostly static geometry which should benefit from this data structure.

KD Tree Path Tracer Optimization

Rony Edde

- A KD-Tree is a data structure that can help solve this problem by dividing the space into sub-spaces and checking for intersections within the subdivided regions instead of the entire geometry. (image from of Wikipedia)



KD Tree Path Tracer Optimization

Rony Edde

Concept.

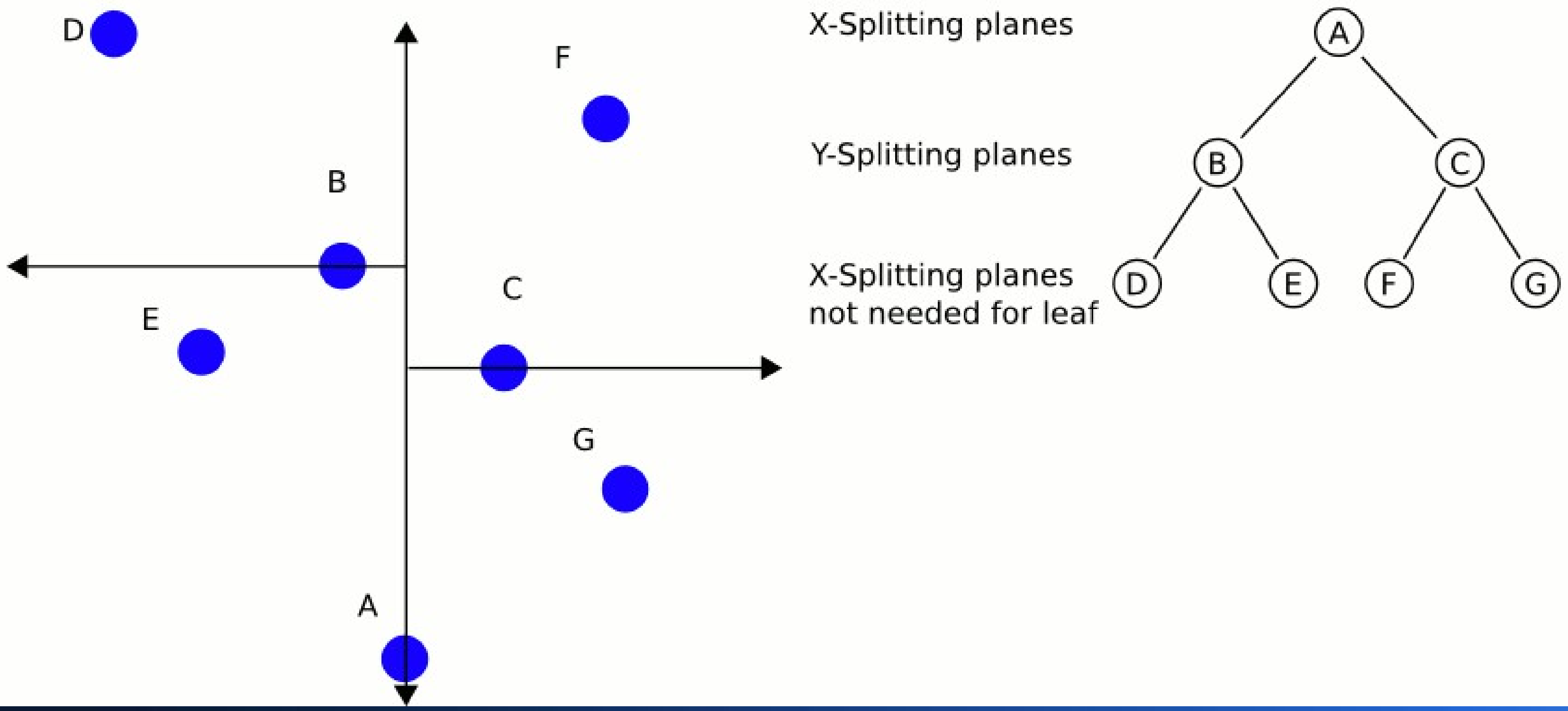
In order to build a KD-Tree, I start by generating a bounding box that encompasses all elements (triangles in our case). I then split on one axis. Every consecutive level will be split on the second axis. In 2D there are 2 axis splits, 3 in 3D. Every split divides the space in 2 sections with new bounds. Triangles that belong to the left side are moved to the left node, right otherwise. This partitions the space into a tree that can then be traversed by only checking if I have a hit with the boundary. This makes a KD-Tree efficient for complex geometry.

KD Tree Path Tracer Optimization

Rony Edde

Concept.

An illustration of a 2D KD-Tree in progress with points.



KD Tree Path Tracer Optimization

Rony Edde

A few problems to solve

- 1- GPU memory limit.
- 2- Streaming solution?
- 3- No support for recursion in CUDA.
- 4- Very tight schedule...

KD Tree Path Tracer Optimization

Rony Edde

A few problems to solve

- 1- GPU memory limit.

GPUs don't have enough memory to compete with CPUs and a typical path tracer loading millions of triangles can easily crash the graphics card. Just accessing the entire geometry without an optimized data structure can be close to impossible without resorting to optimizations. Just accessing the entire geometry without an optimized data structure can be close to impossible without resorting to optimizations. However, even after optimizations, this can remain a major issue with high risks of overwhelming the GPU memory and resulting in a crash.

KD Tree Path Tracer Optimization

Rony Edde

A few problems to solve

- 2- Streaming solution?

In order to solve the memory limitation, we can resort to streaming. If the object loaded has 1 million triangles, it would require 192MB of GPU memory on a 64 bit machine. In today's applications and games 1 million polygons is considered low resolution. We can see how this quickly becomes an issue.

KD Tree Path Tracer Optimization

Rony Edde

A few problems to solve

- 2- Streaming solution?

Streaming alone, however presents additional complexities for optimization. KD-Trees will need to be adaptive and fast in order to account for this. Another option is to split the complexity in sub-trees in order to fit in memory, thus reducing the impact caused by streaming. This could have an impact on the benefit of using KD-Trees and can also present issues when rendering double sided geometry.

KD Tree Path Tracer Optimization

Rony Edde

A few problems to solve

- 3- No support for recursion in CUDA.

The lack of recursion support in CUDA presents a major challenge for any tree like data structure. The most efficient and clean implementation for a tree data structure is recursive.

KD Tree Path Tracer Optimization

Rony Edde

Solving the lack of recursion

- Implementing the KD-Tree without recursion involves a serious rewrite. The initial implementation was made recursively. Once the tree was built and the data is correctly generated, I run tests to make sure that everything works.

KD Tree Path Tracer Optimization

Rony Edde

Solving the lack of recursion

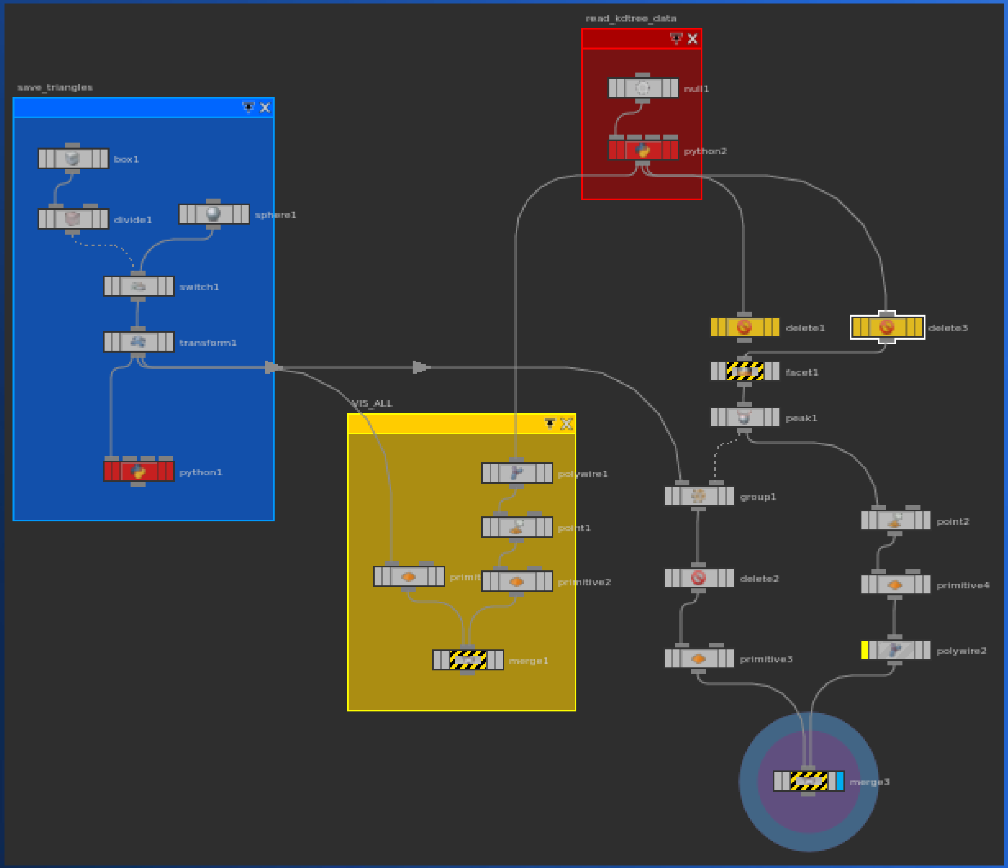
- Testing the tree using print statements is tricky so I've resorted to using Houdini to help prototype and display the KD-Tree before having enough GPU binding to test.
- The following slide shows a Houdini tool that was developed for this purpose

KD Tree Path Tracer Optimization

Rony Edde

Solving the lack of recursion

- Houdini to the rescue.

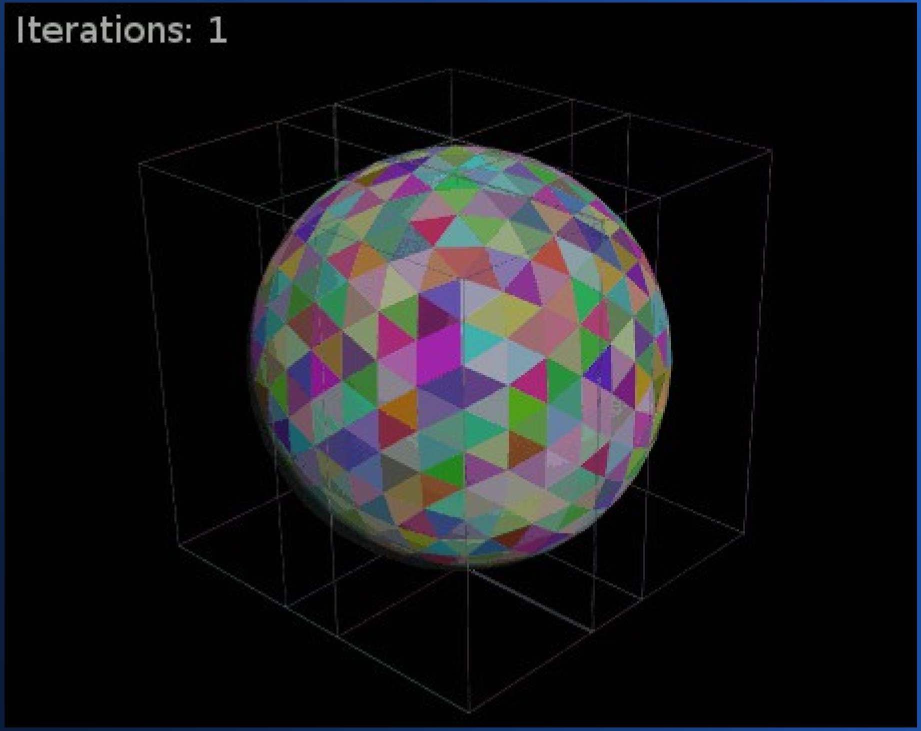


KD Tree Path Tracer Optimization

Rony Edde

Solving the lack of recursion

- Initial results

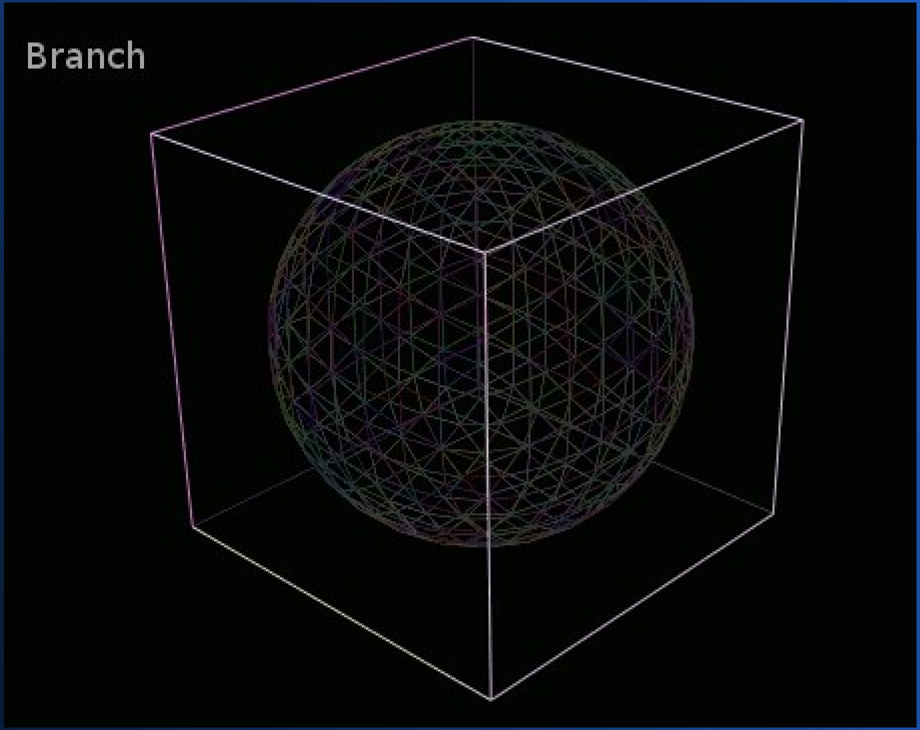


KD Tree Path Tracer Optimization

Rony Edde

Solving the lack of recursion

- Initial results

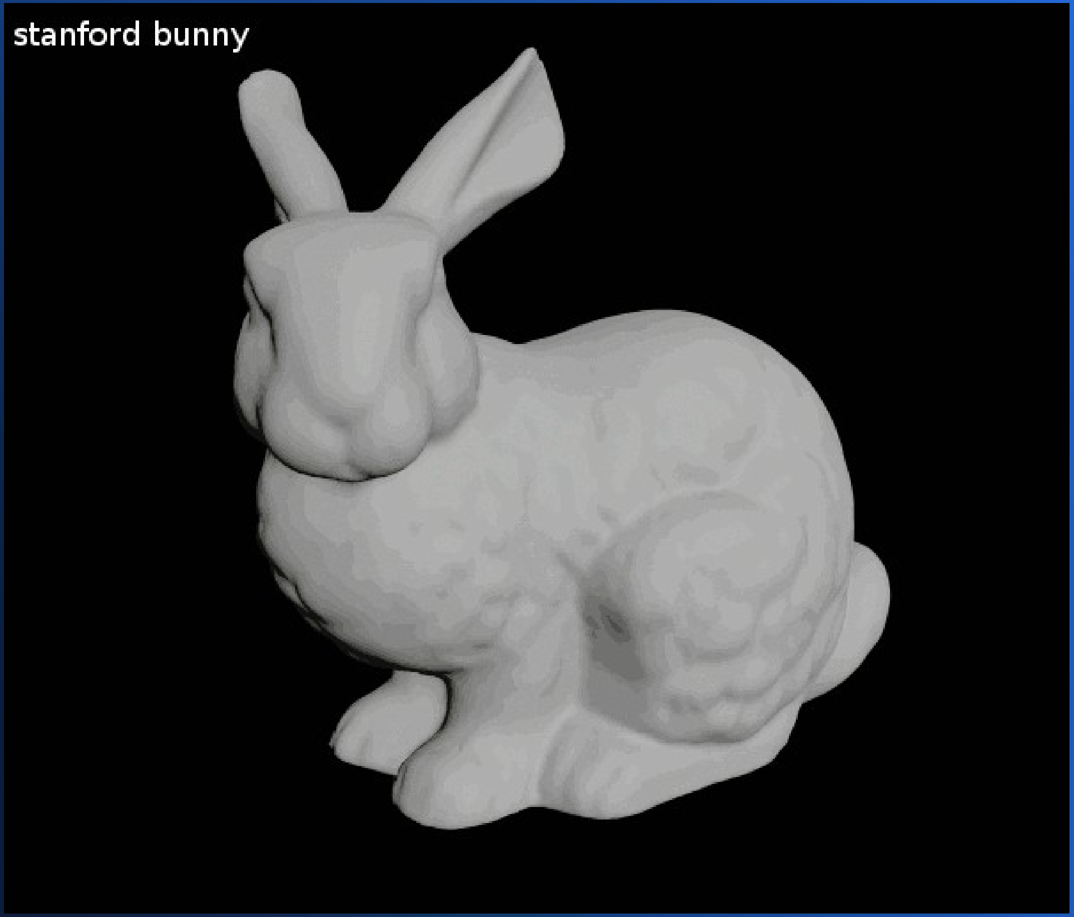


KD Tree Path Tracer Optimization

Rony Edde

Solving the lack of recursion

- Initial results



KD Tree Path Tracer Optimization

Rony Edde

Solving the lack of recursion

- Now that I have a semi working data structure, I start the flattening of the data so that I can traverse the tree without using recursion.

KD Tree Path Tracer Optimization

Rony Edde

Solving the lack of recursion

- Traversing the tree was initially done using attributes that were stored in the node. Visited was a variable added to the nodes that, once a node is visited, is set to true.
- This presented problems because it wasn't thread safe and the node data could get modified by other threads creating a race condition.

KD Tree Path Tracer Optimization

Rony Edde

Solving the lack of recursion

- The solution was to flatten the nodes and the triangles that belonged to the leafs and using indexing in order to traverse the tree.
- The following slide illustrates this.

KD Tree Path Tracer Optimization

Rony Edde

Solving the lack of recursion

- 1- create the KD-Tree.
- 2- save parent / left and right child indices.
- 3- save all triangles into a new array and keep offset stored.
- 4- save all nodes into a new array.
- 5- use index mapping to track visited nodes.

KD Tree Path Tracer Optimization

Rony Edde

Solving the lack of recursion

- Ray hit tracking:

1- set a visited array for all nodes. Set all values to false.

2- Using a while loop, we check if the ray hit the root.

3- If we hit, we set the node index lookup to true.

else we set the current node to the parent

4- If we reached a leaf, check collision with its triangles.

and update hit distance.

- 5- repeat until node parent is -1 or all nodes are visited.

KD Tree Path Tracer Optimization

Rony Edde

Implementation issues.

- While this implementation works and has been tested, the GPU implementation failed initially due to one main problem.
- CUDA's memory management is rigid, meaning that one cannot simply allocate on the GPU without a major performance hit.
- KD-Tree storage must be hidden and only the data can be visible to CUDA. The algorithms must be able to handle dynamic sizes without crashing.

KD Tree Path Tracer Optimization

Rony Edde

Implementation issues.

- After several attempts that ended up crashing the GPU, the final solution was to lock the KD-Tree into the scene description and rewrite the geometry intersection algorithm while only exposing 2 types. KDnode and KDtriangle. This solved most of the problems that were crashing the GPU.
- Solving the non dynamic accessing of the data and the varying sizes was slightly less problematic. The final decision was to resort to a fixed size representation of the indexing that was both small enough not to hinder performance and large enough to contain all the indices.

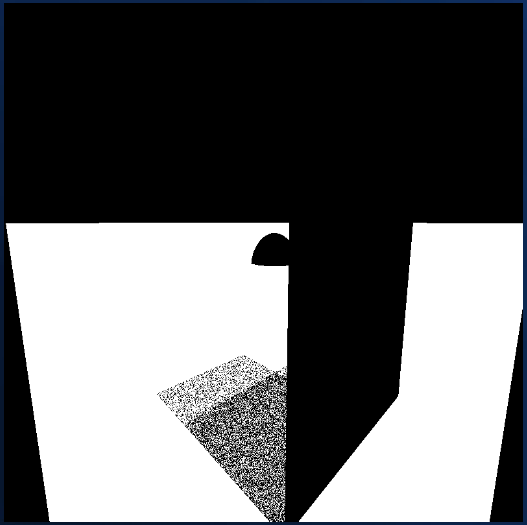
KD Tree Path Tracer Optimization

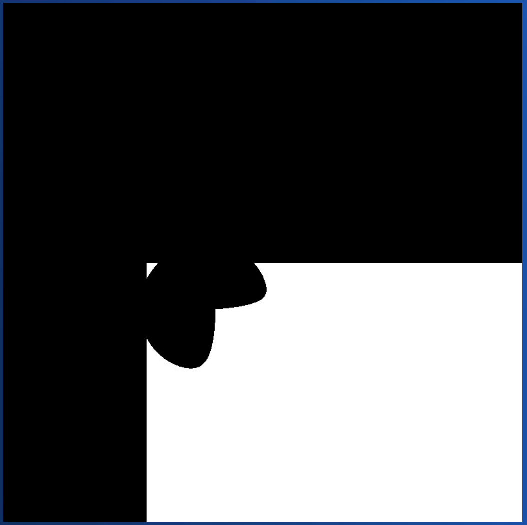
Rony Edde

Implementation issues.

- The last results I got might not look promising but we can clearly see the intersections of the bounding boxes of the tree and the sections of the polygons represented by the sphere.

Note that without the KD-Tree I get 1fps, this was around 30.





KD Tree Path Tracer Optimization

Rony Edde

The nasty bug

- CUDA makes debugging quite challenging sometimes. Nsight sometimes skips breakpoints and exits. This is when the GPU crashes. Printing things out was the only way to track some of the crashing bugs but CUDA can also crash when printing too many lines so tracking the indexing bug was the biggest hurdle.

KD Tree Path Tracer Optimization

Rony Edde

What next?

- The next milestone will be to try fixing the collisions and using the correct colors for correctly bouncing rays.