Università della Svizzera italiana

Faculty of Informatics

Computer Graphics **2024**

Students: Davide Frova, Jamila Oubenali

Bonus Assignment — Spatial Data Structures

1. Overview

In this report, we detail our implementation of spatial data structures to accelerate rendering tasks for large meshes. The assignment required implementing two versions: one using bounding boxes (Version A) and another using a kd-tree (Version B). The results highlight the significant performance improvement achieved with the kd-tree approach.

2. Implementation Details

1. Version A: Bounding Boxes

Version A implemented a bounding box approach for spatial partitioning. This provided a moderate acceleration by reducing the number of unnecessary intersection tests during rendering. However, the linear processing of bounding box content remained a bottleneck for large meshes.

2. Version B: kd-Tree

Motivated by the inefficiencies in Version A, we implemented a kd-tree for Version B. The kd-tree was chosen for its efficient hierarchical partitioning, which allows for fast traversal. We configured the kd-tree with a leaf size of 5 elements, balancing tree depth and traversal cost.

The results were remarkable: the kd-tree drastically reduced rendering times, making it possible to handle meshes with millions of triangles. This significant improvement underscores the importance of effective spatial data structures for complex scenes.

3. Results and Discussion

To evaluate the effectiveness of kd-tree acceleration (Version B) compared to no acceleration, we analyzed render times for four different scenarios with increasing numbers of triangles: the Bunny, Armadillo, Lucy, and a combined scene (all three meshes). The render times and triangle counts are summarized below:

Mesh	# of Triangles	Render Time (No Acc)	Render Time (kd-tree)
Bunny	1,392	90.93 seconds	0.59 seconds
Armadillo	3,112	229.46 seconds	0.83 seconds
Lucy	2,804	199.99 seconds	0.87 seconds
All (Combined)	7,308	868.34 seconds	1.53 seconds

Table 1: Rendering time comparison for no acceleration vs kd-tree acceleration.

Plot Analysis

Figures 1 and 2 visualize the performance comparison.
Linear Scale (Figure 1): The kd-tree acceleration significantly reduces render times compared to the no-acceleration version, particularly as the number of triangles increases. For example, the render time for the combined scene drops from 868 seconds to just 1.53 seconds.
Logarithmic Scale (Figure 2): On a logarithmic scale, the trend highlights the exponential growth in render times for the unoptimized version compared to the near-linear growth for the kd-tree implementation. This confirms the scalability of the kd-tree for large datasets.

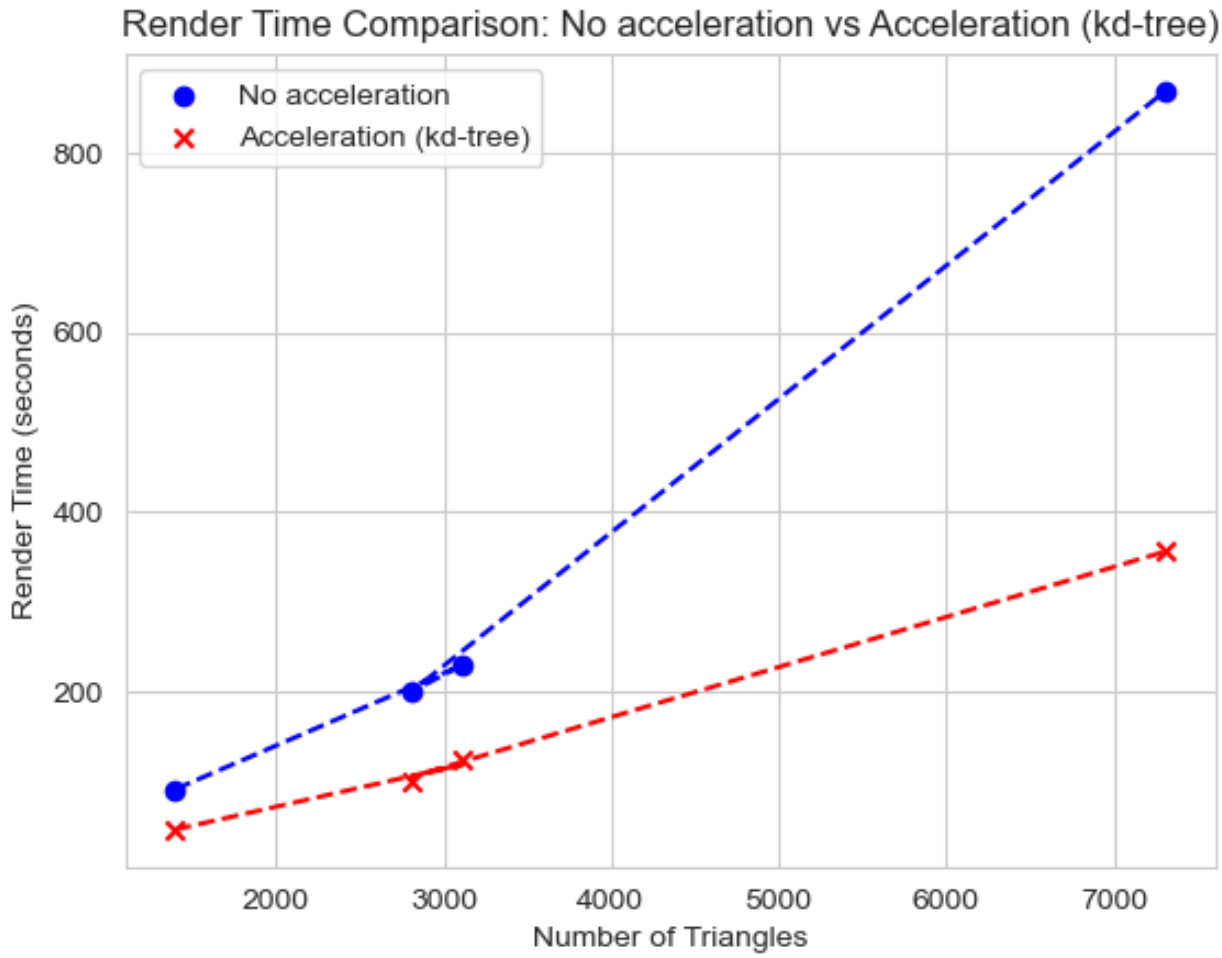

Figure 1: Rendering time comparison: No acceleration vs. kd-tree acceleration.

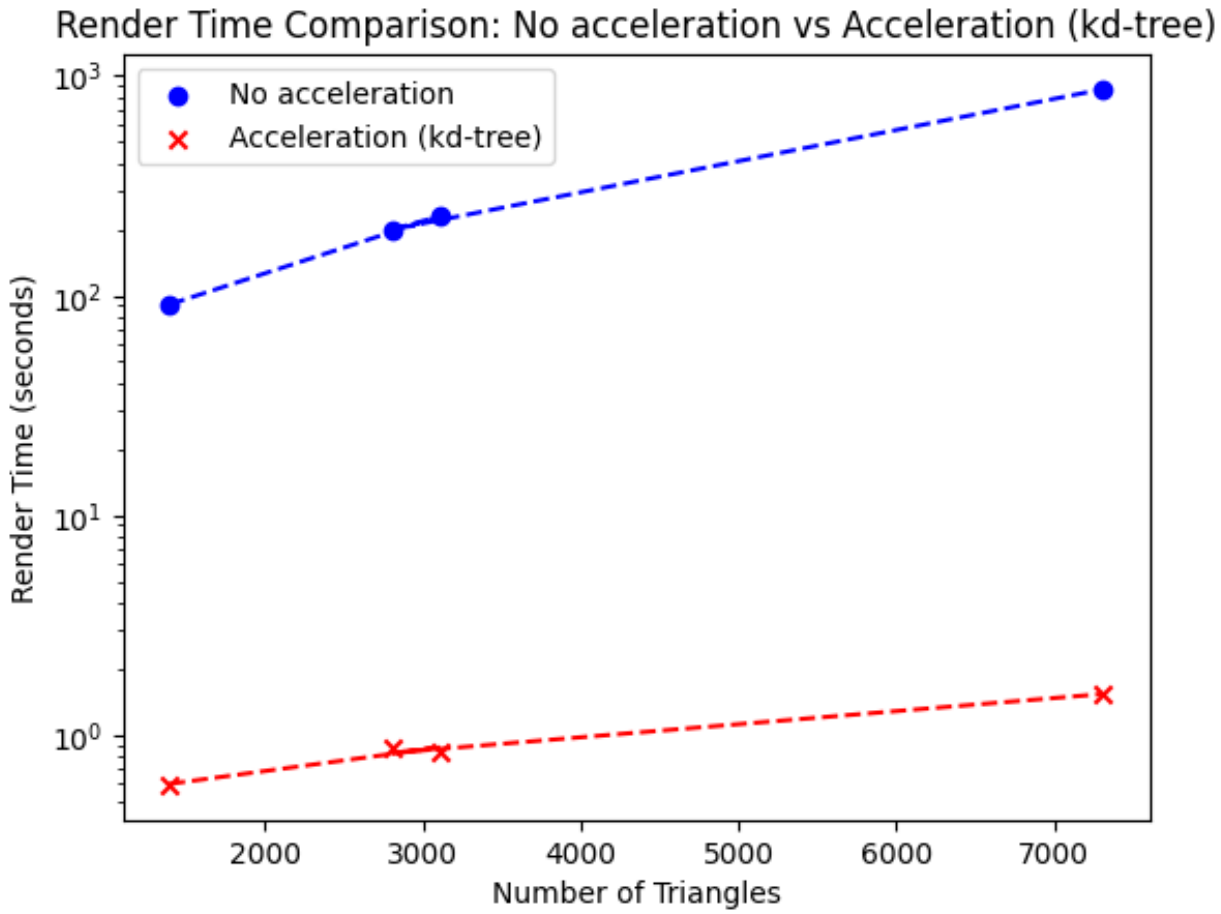

Figure 2: Rendering time comparison: No acceleration vs. kd-tree acceleration (logarithmic scale).

1. High-Resolution Meshes with Shadows Enabled

While the above comparisons were based on low-resolution meshes with shadows disabled, we also tested the kd-tree implementation on a full-resolution scene with shadows enabled. The scene included the following properties:

- **Meshes**:
  - **Armadillo**
    - **Number of Triangles**: 345,944
    - **Number of Vertices**: 172,974
  - **Bunny**
    - **Number of Triangles**: 69,451
    - **Number of Vertices**: 35,947
  - **Lucy**
    - **Number of Triangles**: 2,805,572
    - **Number of Vertices**: 1,402,794
- **Total Number of Triangles**: 3,220,967
- **Total Number of Vertices**: 1,611,715

The results were impressive:

- Time to define the scene (building kd-tree included): **13.41 seconds**.

- Time to render the image: **18.28 seconds**.

These results demonstrate the kd-tree's scalability, even when rendering scenes with millions of triangles and computationally expensive features such as shadows.

Key Observations

1. **Drastic Speedup**: The kd-tree reduced render times by 2-3 orders of magnitude for low-resolution meshes and maintained exceptional performance for high-resolution scenes.
2. **Scalability**: The kd-tree handled a full scene with over 3.2 million triangles and shadows enabled, demonstrating its scalability for highly complex scenes.
3. **Low Overhead**: For low-resolution meshes, the kd-tree maintained sub-second render times, even for moderately complex scenes.

4. Future Optimizations

One potential optimization is to explore a balance between the kd-tree's leaf size and introducing a Bounding Volume Hierarchy (BVH) for processing leaf nodes. Currently, leaf nodes are processed linearly, which may still become a bottleneck for highly complex meshes. Implementing a BVH within the leaf nodes could further improve traversal efficiency.

5. Conclusion

The implementation of the kd-tree (Version B) demonstrated a dramatic improvement in performance compared to the bounding box approach (Version A). The kd-tree scaled efficiently, handling both low-resolution meshes and high-resolution scenes with millions of triangles. While further optimizations, such as BVH integration, remain as potential future work, the kd-tree has proven to be a highly effective spatial data structure for complex rendering tasks.