POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Ray Distribution Aware Heuristics for Bounding Volume Hierarchies Construction

Tesi di Laurea Magistrale in
Computer Science and Engineering

Author: **Lapo Falcone**

Student ID: 996089
Advisor: Prof. Marco Gribaudo
Academic Year: 2023-24

Abstract

Abstract

Keywords: here, the keywords, of your thesis

Abstract in lingua italiana

Abstract Italiano

Parole chiave: qui, vanno, le parole chiave, della tesi

Contents

Introduction

Intro [1]

1 | Chapter one

Chapter 1

2 | Chapter two

Chapter 2

Bibliography

[1] D. Meister, S. Ogaki, C. Benthin, M. J. Doyle, M. Guthe, and J. Bittner. A survey on bounding volume hierarchies for ray tracing. In *Computer Graphics Forum*, volume 40, pages 683–712. Wiley Online Library, 2021.

[2] S. Owen. Ray - box intersection. `https://education.siggraph.org/static/HyperGraph/raytrace/rtinter3.htm`, 2001. Accessed: (10/01/2024).

A | Collision and Culling Algorithms

A.1. Ray-AABB Intersection

The algorithm we used to detect intersections between a ray and an AABB is the branchless slab algorithm [2]. Given a ray in the form: $r(t) = O + t \cdot d$, where O is the origin and d the direction, the main idea of the algorithm is to find the 2 values of t ($\overline{t_1}$ and $\overline{t_2}$) such that $r(\overline{t_{1,2}})$ are the points where the ray intersects the AABB.

Since the object to intersect the ray with is an axis-aligned bounding box in the min-max form, the algorithm can proceed one dimension at a time:

- First, it finds the intersection points of the ray with the planes parallel to the yz plane, and sorts them in an ascending order with reference to the corresponding $\overline{t_{1,2}}$ values.
- Then it does the same with the xz plane:
  - As closest intersection point, it keeps the furthest between the 2 closest intersection points found so far.
  - As furthest intersection point, it keeps the closest between the 2 furthest intersection points found so far.
- Then it does the same with the xy plane.
- Finally, an intersection is detected only in the case where the furthest intersection point is actually further than the closest one found by the algorithm.
- The returned $\bar{t}$ value is the smaller one, as long as it is greater or equal to 0, otherwise it means that the origin of the ray is inside the AABB, and one of the intersection points is *behind* the ray origin.

It is interesting to note how, under the floating-point IEEE 754 standard, the algorithm also works when it is not possible to find an intersection point along a certain axis (i.e. when the ray is parallel to certain planes). Indeed, in such cases, the values $\overline{t_{1,2}}$ will be $\pm$ inf, and the comparisons will still be well defined.

Algorithm A.1 Ray-AABB branchless slab intersection algorithm in 3 dimensions

```
function INTERSECT(aabb, ray)
    tx1 ← (aabb.min.x − ray.origin.x)/ray.direction.x
    tx2 ← (aabb.max.x − ray.origin.x)/ray.direction.x
    tMin ← min(tx1, tx2)
    tMax ← max(tx1, tx2)
    ty1 ← (aabb.min.y − ray.origin.y)/ray.direction.y
    ty2 ← (aabb.max.y − ray.origin.y)/ray.direction.y
    tMin ← max(tMin, min(ty1, ty2))
    tMax ← min(tMax, max(ty1, ty2))
    tz1 ← (aabb.min.z − ray.origin.z)/ray.direction.z
    tz2 ← (aabb.max.z − ray.origin.z)/ray.direction.z
    tMin ← max(tMin, min(tz1, tz2))
    tMax ← min(tMax, max(tz1, tz2))
    areColliding ← tMax > tMin and tMax ≥ 0
    collisionDist ← tMin < 0?tMax : tMin
    return ⟨areColliding, collisionDist⟩
```

List of Figures

List of Tables

List of Symbols

Symbol	Description	Unit
alpha	symbol 1	km

Acknowledgements

Ringrazio...