Partição Binária do Espaço (BSP) com Triângulos 3D

Richard Fernando Heise Ferreira

3 de junho de 2025

Resumo

Este relatório descreve a implementação de uma árvore BSP (Binary Space Partitioning) para modelagem e análise geométrica no espaço tridimensional. A estrutura é construída a partir de triângulos 3D e utilizada para verificar interseções com segmentos de reta. O projeto foi desenvolvido com fins educacionais e computacionais, utilizando C++ como linguagem base.

1 Introdução

A técnica de Partição Binária do Espaço (BSP) é utilizada para subdividir recursivamente um espaço em regiões convexas, sendo útil em várias áreas como gráficos computacionais, simulação física e modelagem de cenas. Neste projeto, implementamos uma árvore BSP para o espaço tridimensional $\mathbb{R}^3$, com planos definidos por triângulos e operações de classificação e interseção com segmentos de reta.

2 Arquivos do Projeto

O projeto está organizado da seguinte forma:

- `main.cpp` – Função principal que realiza a leitura de entrada, construção da BSP e testes de interseção.
- `bsp.cpp / bsp.hpp` – Implementação da estrutura de dados BSP e algoritmos associados.
- `tests/` – Contém os arquivos de teste organizados em subpastas:
  - `inputs/` – Arquivos de entrada.
  - `answers/` – Saídas esperadas.
  - `outputs/` – Saídas geradas.
  - `imgs/` – Imagens que representam graficamente os casos de teste.
- `run_tests.sh` – Script para execução automatizada dos testes.
- `Makefile` – Compilação do projeto.
- `README.md` – Instruções gerais sobre a execução.

3 Estrutura BSP

3.1 Definições

Cada nó da árvore BSP representa uma divisão do espaço tridimensional por um plano, definido por um triângulo da cena. A árvore é binária: triângulos que ficam "à frente"do plano vão para o filho da frente; os que ficam "atrás", para o filho de trás. Triângulos coplanares são armazenados no próprio nó.

O objetivo da BSP é organizar o espaço de modo eficiente para facilitar operações de consulta, como verificação de visibilidade, detecção de colisões ou interseções com segmentos.

3.2 Construção da Árvore

A construção da árvore BSP segue uma abordagem recursiva. A cada chamada, um triângulo é escolhido como plano de partição. Os demais triângulos são classificados em relação a esse plano:

- Se um triângulo está inteiramente de um lado do plano, é encaminhado para o lado correspondente.
- Se está parcialmente de cada lado, ele é subdividido em dois ou mais triângulos menores.
- Se está coplanar, é mantido no nó atual.

Esse processo é repetido recursivamente nas sublistas até que todos os triângulos estejam alocados.

A escolha do triângulo de partição afeta significativamente o balanceamento da árvore. Uma má escolha pode gerar árvores degeneradas com profundidade linear, enquanto escolhas cuidadosas (ex.: heurísticas de balanceamento) podem produzir árvores mais equilibradas.

Complexidade: No pior caso, a complexidade da construção é exponencial em relação ao número de triângulos, devido à possível subdivisão em múltiplos triângulos. Na prática, com partições razoáveis, a construção tende a ter custo próximo de $O(n \log n)$.

3.3 Classificação de Triângulos

Cada triângulo é classificado com base nos sinais do produto escalar entre o vetor normal do plano de partição e o vetor que liga um ponto do plano ao vértice do triângulo. Dependendo dos sinais dos três vértices:

- Todos positivos: triângulo está na frente.
- Todos negativos: triângulo está atrás.
- Mistos: triângulo cruza o plano e deve ser subdividido.

A subdivisão é feita encontrando as interseções entre as arestas do triângulo e o plano, criando novos triângulos com vértices nos pontos de interseção.

Complexidade: Cada operação de classificação é $O(1)$. No entanto, a necessidade de subdivisão pode aumentar o número total de triângulos, afetando a construção da árvore.

4 Classificação de Triângulos

A classificação é feita comparando os vértices de cada triângulo com o plano definido pelo triângulo de partição. Usamos produtos escalares para determinar se os pontos estão na frente, atrás ou no plano.

Em casos de interseção, o triângulo é subdividido em subtriângulos em lados distintos do plano.

5 Interseção com Segmentos

Após construída a BSP, é possível testar interseções entre segmentos e a malha de triângulos de forma eficiente.

O algoritmo percorre a árvore recursivamente. Em cada nó, o segmento é classificado em relação ao plano:

- Se ambos os pontos do segmento estão de um mesmo lado, o algoritmo continua apenas naquela subárvore.
- Se o segmento cruza o plano, o ponto de interseção é calculado e a busca continua nos dois lados.
- Se o segmento está coplanar ao triângulo, a interseção é tratada com projeção 2D e coordenadas baricêntricas para verificar se ele de fato intersecta a área do triângulo.

Essa estratégia permite evitar testes desnecessários com triângulos irrelevantes, já que muitos são automaticamente descartados ao seguir apenas os ramos pertinentes da árvore.

Complexidade: Para uma árvore balanceada com n triângulos, a verificação de interseção para um segmento ocorre em tempo $O(\log n + k)$, onde k é o número de triângulos efetivamente intersectados.