Universidade do Minho
Escola de Engenharia

Universidade do Minho
Escola de Engenharia
Mestrado em Engenharia Informática

Unidade Curricular de Visualização e Iluminação

Ano Letivo de 2024/2025

Projeto final - Grupo 1

Martim Redondo
PG57889

Jéssica Cunha
A100901

Francisco Afonso
PG57873

Junho, 2025

Índice

1. Introdução **1**

2. Acceleration Structure - BVH **1**

2.1. Fundamentação Teórica 1

2.2. Implementação 1

2.3. Inovação: Dual BVH System 2

2.4. Resultados e Performance 2

3. Tone Mapping **2**

3.1. O Problema HDR 2

3.2. Evolução da Implementação 3

3.3. Algoritmos Implementados 3

3.4. Comparação Visual 4

4. Alternative Cameras **4**

4.1. Fisheye Camera 4

4.2. Características e Aplicações 5

4.3. Orthographic Camera 6

4.4. Performance 7

5. Conclusão **7**

6. Referências **8**

Lista de Figuras

Figura 1 Comparação visual entre os diferentes algoritmos de *Tone Mapping* **4**

Figura 2 Todos os tipos de configurações do Fisheye. **5**

Figura 3 Perspective VS Fisheye 180º **6**

Figura 4 Comparação visual entre as diferentes câmaras e angulos **7**

Lista de Tabelas

Tabela 1 Comparação do programa com HBV e sem HBV para cenas com diferentes primitivas **2**

Tabela 2 Speed up com e sem o uso do BVH em diferentes cenas com diferente número de primitivas **2**

1. Introdução

O seguinte documento aborda a implementação das três funcionalidades avançadas, escolhidas pelo grupo, para o *Ray Tracing* desenvolvido no âmbito da Unidade Curricular de Visualização e Iluminação. As *features* escolhidas foram: **Acceleration Structure: BVH**, **Alternative Cameras** e **Tone Mapping**.

Estas melhorias visam aumentar, significativamente, o desempenho e a versatilidade do sistema de renderização, seguindo, rigorosamente, as referências académicas fornecidas pelo professor, nomeadamente o livro "Physically Based Rendering" (PBRT).

2. Acceleration Structure - BVH

2.1. Fundamentação Teórica

O *Bounding Volume Hierarchy* (BVH) é uma ferramenta a ser referida quando o tema é *Ray Tracing* moderno. A nossa implementação segue, fielmente, a abordagem descrita nos capítulos **4.3 (2ª edição)** e **4 (3ª edição)**, garantindo uma base teórica sólida e uma performance otimizada.

O BVH organiza os objetos de uma cena 3D de forma hierárquica, onde cada nó da árvore contém uma *bounding box* que conterá **X** objetos nos ramos subsequentes. Desta forma, consegue-se descartar de forma eficiente grandes grupos de primitivas durante o teste de interseção com raios, reduzindo a complexidade de **O(N)** para **O(log N)**.

2.2. Implementação

Seguindo o livro **PBRT (capítulos 4.3 e 4)**, implementámos um BVH com as seguintes características:

- **Construção com SAH (Surface Area Heuristic)**: O algoritmo analisa diferentes formas de dividir as primitivas e escolhe a melhor com base no custo mínimo da travessia. Isto resulta em árvores mais equilibradas e eficientes.
- **Representação Linear Compacta**: Após a construção, a árvore é convertida para um *array linear*, melhorando, assim, a localidade da *cache* durante a travessia. Cada nó ocupa, exatamente, 32 bytes, para se conseguir obter o máximo de eficiência possível.
- **Travessia Otimizada**: Uso de uma *stack explícita* em vez de *recursão* e pré-computamos alguns valores, como o inverso da direção do raio, para acelerar os testes de interseção e, mais uma vez, maximizar a eficiência do algoritmo.

2.3. Inovação: Dual BVH System

Além da implementação padrão proposta no **PBRT (capítulos 4.3 e 4)**, decidiu-se que seria benéfico ter um sistema inovador com dois BVHs separados:

- **BVH Principal**: Para toda a geometria da *scene*;
- **Light BVH**: Exclusivo para fontes de luz de área.

Esta separação permite otimizações específicas para cálculos de iluminação direta, particularmente útil em cenas com muitas luzes.

2.4. Resultados e Performance

Primitivas	**Com HBV**	**sem BVH**
1k	0.618 seg	19.786 seg
10k	6.328 seg	308.778 seg
100k	42.154 seg	4096.731 seg
1M	514.243 seg	51266.912 seg

Tabela 1: Comparação do programa com HBV e sem HBV para cenas com diferentes primitivas

---

Cena	**Primitivas**	**com BVH**	**sem BVH**	**Speedup**
Cornell Box	37	14.603 seg	19.132 seg	1.31x
Massive Sphere Scene	1k	0.618 seg	19.786 seg	32.02x
Massive Sphere Scene	10k	6.328 seg	308.778 seg	48.80x

Tabela 2: Speed up com e sem o uso do BVH em diferentes cenas com diferente número de primitivas

O sistema de monitorização integrado (*debugging*) permite acompanhar a eficácia do BVH em tempo real, mostrando estatísticas como:

- *hit rate*;
- número de testes de interseção evitados.

3. Tone Mapping

3.1. O Problema HDR

O *Ray Tracing* produz imagens com valores de iluminação que podem variar de 0 a milhares, contudo os monitores apenas conseguem mostrar valores entre 0 e 1.

O **Tone Mapping** resolve este problema comprimindo o range dinâmico, tentando ao máximo, preservar os detalhes visuais.

3.2. Evolução da Implementação

Situação Original: O sistema aplicava, automaticamente, o algoritmo *Reinhard Basic*, contudo durante a implementação reparou-se que esta automatização limitava a flexibilidade artística e não permitia uma comparação autentica entre diferentes algoritmos.

Nova Arquitetura: O *Tone Mapping* automático foi removido e substituído por um sistema modular e configurável.

O *Reinhard Basic*, anteriormente *hardcoded*, tornou-se uma das cinco opções disponíveis. Esta mudança permite:

- Escolha consciente do algoritmo;
- Possibilidade de se obterem imagens HDR sem *Tone Mapping*;
- Comparação direta entre diferentes algoritmos.

3.3. Algoritmos Implementados

Implementámos cinco algoritmos diferentes, cada um com características específicas:

- **Reinhard Basic**: O método clássico que usa a fórmula $\frac{L}{1+L}$. Simples mas eficaz para a maioria das cenas;
- **Reinhard Advanced**: Extensão que permite controlar o "*burn out*" de áreas muito brilhantes através de um parâmetro específico deste algoritmo, *white point*.
- **ACES (Academy Color Encoding System)**: O padrão da indústria cinematográfica pois, produz resultados com maior contraste e cores mais vibrantes, ideal para imagens finais de alta qualidade.
- **Exposure**: Simula o comportamento de filme fotográfico usando a fórmula $1 - e^{-\text{exposure} \times L}$. Permite controlo intuitivo através de um único parâmetro.
- **Linear**: Multiplicação simples com *clamp*. Útil como *baseline* e para cenas com range dinâmico limitado.

3.4. Comparação Visual

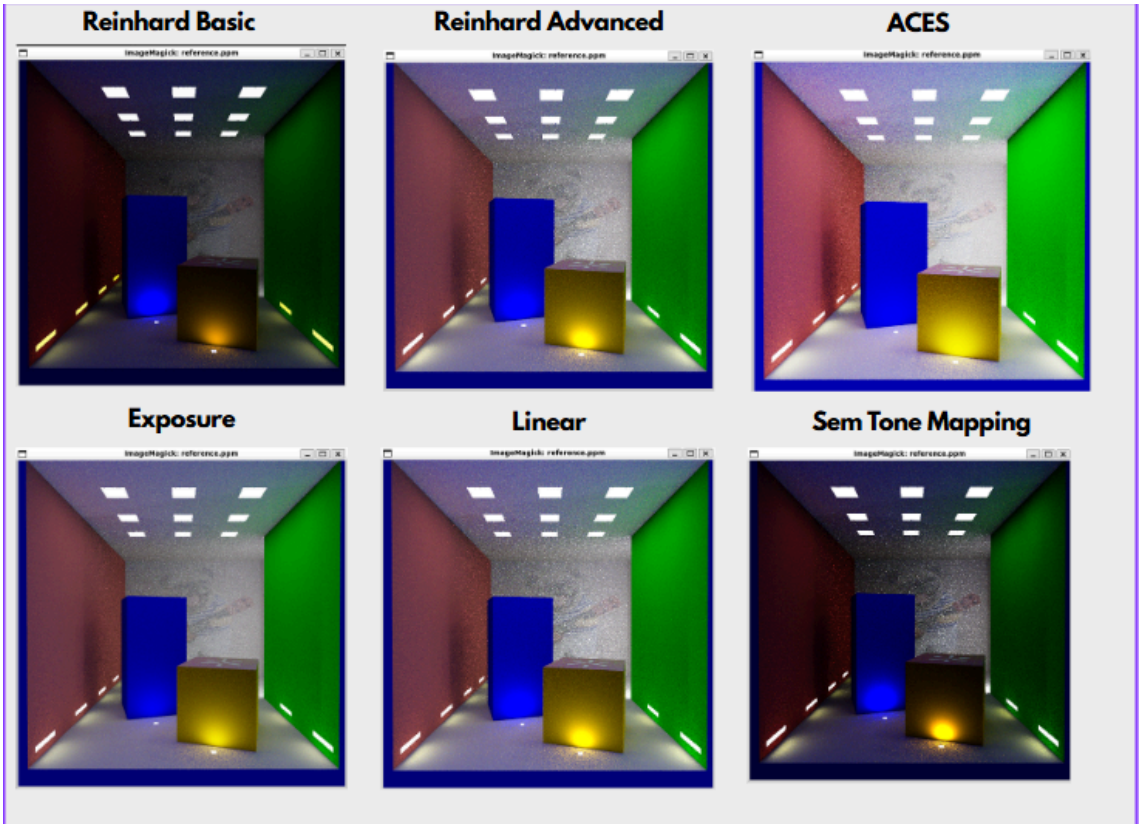

Figura 1: Comparação visual entre os diferentes algoritmos de *Tone Mapping*

Cada algoritmo tem aplicações específicas:

- **ACES**: Produção final, aspeto cinematográfico;
- **Reinhard**: Uso geral, boa preservação de detalhes;
- **Exposure**: Controlo artístico intuitivo.

4. Alternative Cameras

4.1. Fisheye Camera

Implementámos uma câmara **Fisheye** completa com quatro modelos de projeção diferentes, cada um adequado para aplicações específicas:

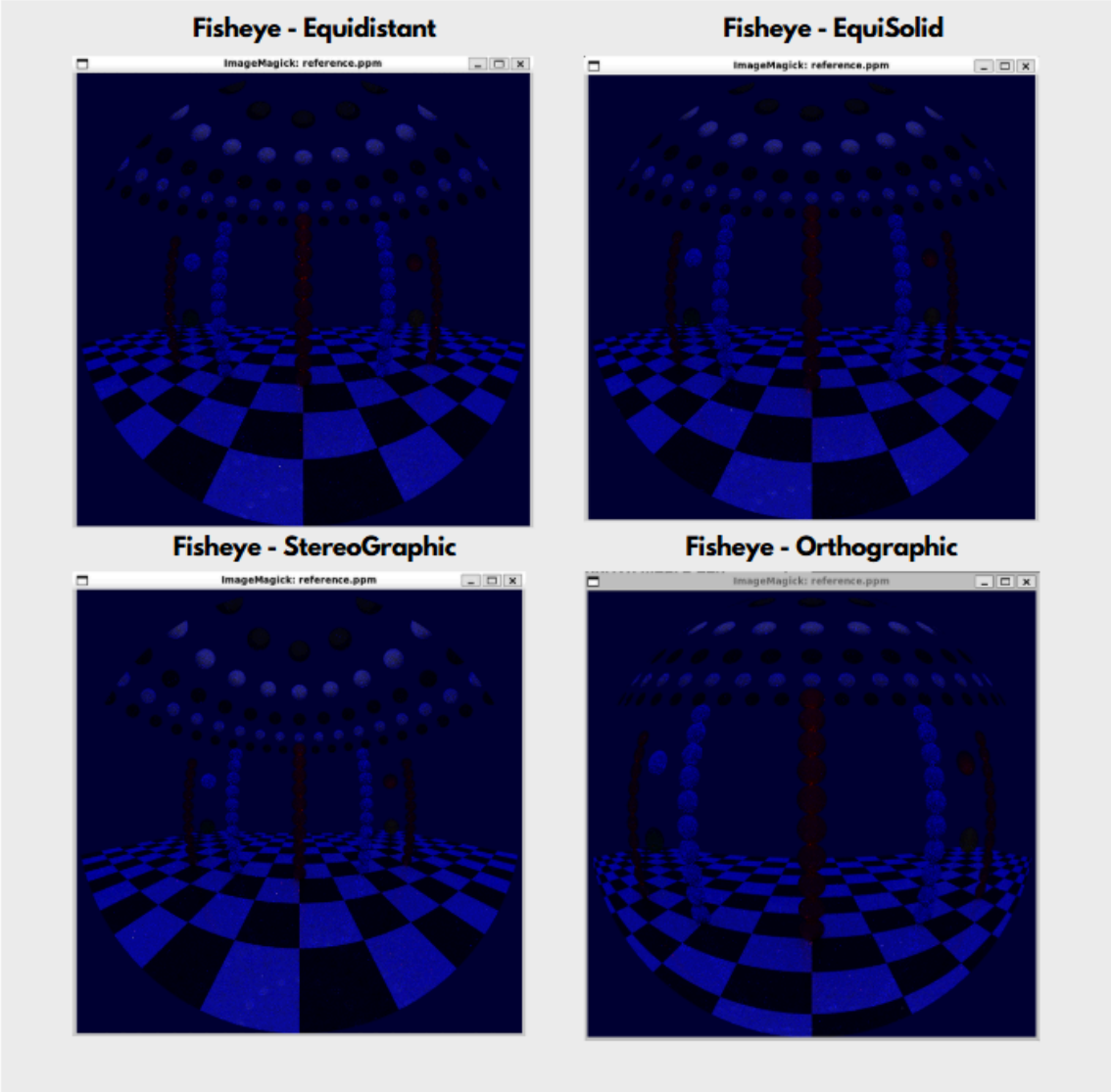

Figura 2: Todos os tipos de configurações do Fisheye.

As mudanças nas imagens podem não ser totalmente evidentes, contudo, na teoria e até mesmo, com atenção na prática, iremos ver os seguintes pormenores:

- **Stereographic**: Máxima distorção nas bordas, ideal para navegação (preserva direções);
- **Equidistant**: Distorção uniforme, ideal para medições angulares;
- **Equisolid**: Balanceada, ideal para análise de áreas;
- **Orthographic**: Mínima distorção, mais "natural" ao olho humano.

4.2. Características e Aplicações

Campo de Visão: Configurável até 180°, permitindo capturar toda uma semi-esfera numa única imagem.

Aplicações Práticas:

- Visualização arquitetónica de interiores;

- Sistemas de segurança e vigilância;
- Efeitos artísticos e criativos;
- Fotografia 360° (com múltiplas capturas).

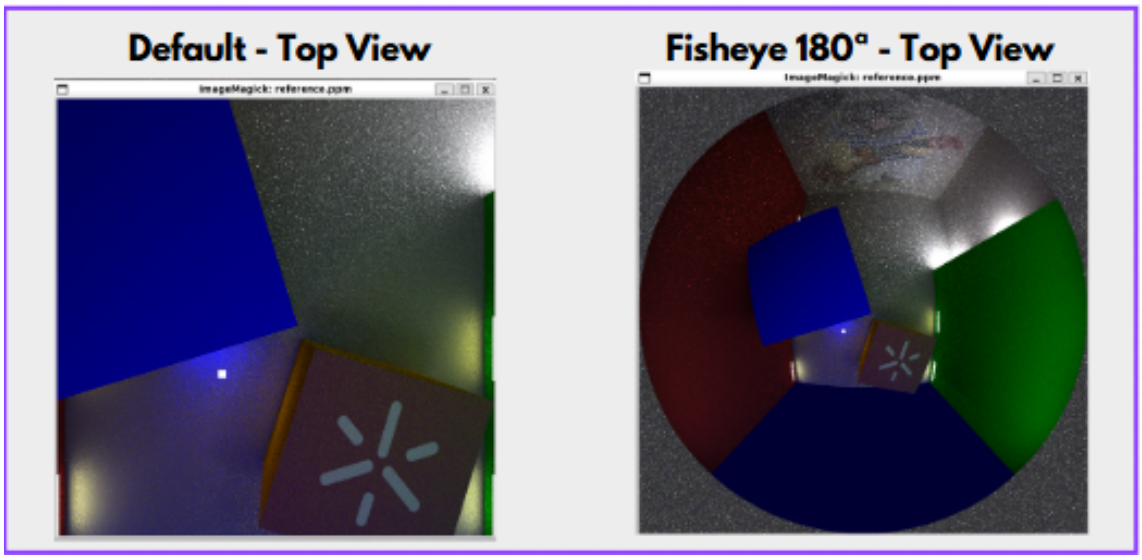

Figura 3: Perspective VS Fisheye 180º

4.3. Orthographic Camera

A câmara ortográfica elimina completamente a perspetiva, sendo essencial para aplicações técnicas:

Características:

- Todos os raios são paralelos;
- Objetos mantêm o mesmo tamanho independentemente da distância;
- Sem pontos de fuga.

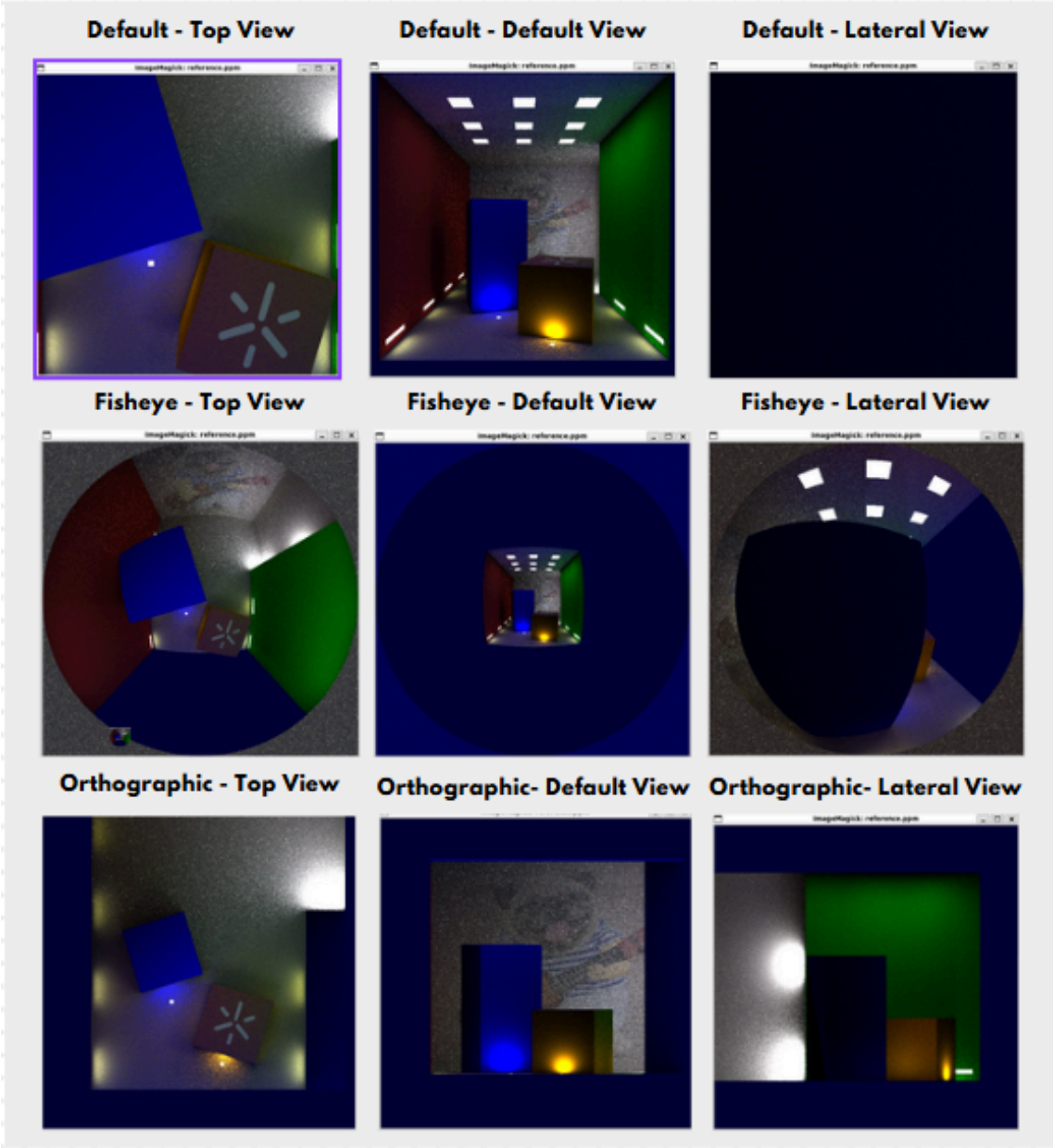

Figura 4: Comparação visual entre as diferentes câmaras e angulos

4.4. Performance

Todas as câmaras mantêm complexidade **O(1)** por raio gerado, sem *overhead* adicional comparado com a câmara perspetiva tradicional. A diferença de performance entre elas é passível de ser ignorada (<1%).

5. Conclusão

Em suma, na nossa opinião, conseguiu-se concluir todas as *features* escolhidas com um bom resultado e desempenho.

As três funcionalidades implementadas elevam, não só a complexidade, mas, principalmente, a capacidade do *Ray Tracing*:

- **BVH** transformou o sistema de um protótipo académico num *render* capaz de lidar com cenas de produção. Com este tipo de algoritmo torna-se plausível trabalhar com modelos mais complexos do que os dados no *BuildScenes*.
- **Tone Mapping** expandido oferece controlo profissional sobre o aspeto final das imagens.
- **Alternative Cameras** abrem novas possibilidades criativas e técnicas, desde visualizações arquitetónicas precisas até efeitos artísticos imersivos.

Quanto às possíveis melhorias no projeto, poderia-se ter implementado mais câmaras, pois daria maior variabilidade às imagens geradas. Poderia-se ter criado mais cenas para se conseguir explorar melhor algumas funcionalidades, principalmente, as diferentes configurações do *FishEye*, uma vez que fica pouco evidente as diferenças entre as quatro possíveis. Quanto ao *BVH*, poderia-se melhorar a forma de acesso às luzes, o que melhoraria, ainda mais, o desempenho da renderização da cena.

6. Referências

[1] Pharr, M., Jakob, W., & Humphreys, G. (2016). Physically Based Rendering: From Theory to Implementation (3rd ed.). Morgan Kaufmann. Capítulos 4 e 6.3.

[2] Pharr, M., & Humphreys, G. (2010). Physically Based Rendering: From Theory to Implementation (2nd ed.). Morgan Kaufmann. Capítulo 4.3.

[3] Wikipedia Contributors. (2024). Tone mapping. https://en.wikipedia.org/wiki/Tone_mapping

[4] Hable, J. (2024). Tone Mapping. https://64.github.io/tonemapping/