Computação Gráfica

Relatório do Projeto Final: Labirinto 3D

Autores:

Alexandre Santos n52011
Vasco Colaco n52290

Unidade Curricular: Computação Gráfica
Curso: Licenciatura em Engenharia Informática

29 de dezembro de 2025

Conteúdo

1 Introdução

Este relatório descreve o desenvolvimento do projeto final da unidade curricular de Computação Gráfica. O objetivo principal do trabalho foi a criação de uma aplicação gráfica interativa em 3D, utilizando a API OpenGL, que aplicasse os conhecimentos adquiridos ao longo do semestre.

1.1 O Jogo: Labirinto 3D

A aplicação desenvolvida consiste num jogo de exploração em primeira pessoa. O jogador é colocado num labirinto complexo e escuro, equipado apenas com uma lanterna. O objetivo é navegar pelos corredores, evitando obstáculos e encontrando o caminho correto até ao portão de saída.

1.2 Motivação

A escolha deste tema deveu-se ao interesse em compreender os fundamentos dos motores de jogo modernos (Game Engines). Implementar um sistema de câmaras, deteção de colisões e iluminação dinâmica "do zero"(utilizando apenas OpenGL e bibliotecas auxiliares básicas) permite uma compreensão profunda da pipeline gráfica e da matemática vetorial envolvida na renderização 3D.

2 Tecnologias Utilizadas

Para o desenvolvimento deste projeto, foram selecionadas tecnologias padrão na indústria e na comunidade de aprendizagem de Computação Gráfica:

- **OpenGL (Core Profile 3.3)**: API gráfica principal utilizada para a renderização de primitivas, gestão de buffers e shaders.
- **C++**: Linguagem de programação utilizada, escolhida pela sua performance e controlo sobre a gestão de memória.
- **GLFW**: Biblioteca utilizada para a criação da janela, contexto de renderização e gestão de input (teclado e rato).
- **GLAD**: Biblioteca para carregar os ponteiros das funções OpenGL em tempo de execução.
- **GLM (OpenGL Mathematics)**: Biblioteca de matemática (header-only) compatível com GLSL, utilizada para operações com vetores e matrizes (transformações, projeções, câmara).
- **stb_image**: Biblioteca para carregamento de texturas (PNG, JPG).
- **TinyOBJLoader**: Biblioteca para importação de modelos 3D no formato Wavefront (.obj).

3 Desenvolvimento e Implementação

3.1 Gestão do Projeto e Estrutura

O projeto foi estruturado de forma orientada a objetos para garantir a modularidade e a manutenibilidade do código. As principais funcionalidades foram encapsuladas em classes específicas.

3.2 Descrição das Classes e Módulos

3.2.1 Main (main.cpp)

O ficheiro principal contém o ciclo de jogo (*game loop*). É responsável por:

- Inicializar o GLFW e GLAD.
- Configurar os callbacks de input.
- Gerir o cálculo do `deltaTime` para garantir movimentos independentes da taxa de frames.
- Executar o loop de renderização e processamento logico.

3.2.2 Câmara (Camera.h)

Implementa uma câmara FPS (*First-Person Shooter*). Utiliza ângulos de Euler (Yaw e Pitch) para calcular o vetor diretor da câmara com base no movimento do rato. Processa também o input do teclado para mover a posição da câmara no mundo (W, A, S, D).

3.2.3 Maze (Maze.h)

Esta classe é responsável pela gestão do mundo do jogo.

- **Carregamento do Nível**: Lê o ficheiro .obj do labirinto e separa os dados em vértices, normais e coordenadas de textura.
- **Classificação de Superfícies**: Analisa a normal de cada triângulo para distinguir entre "chão"(normais a apontar para cima) e "paredes".
- **Sistema de Colisões (Grid Espacial)**: Para otimizar a deteção de colisões, o mundo 3D é dividido numa grelha 2D (XZ). Cada celula da grelha armazena referências para os triângulos que a intersetam. Quando o jogador se move, o sistema verifica colisões apenas contra os triângulos nas células vizinhas, drasticamente reduzindo o custo computacional.

3.2.4 Gestor de Shaders (Shader_m.h e Ficheiros GLSL)

Foi criada uma classe para abstrair a compilação e linkagem dos shaders. O projeto utiliza vários shaders:

- **Basic Lighting**: Implementa o modelo de iluminação (Phong/Blinn-Phong), texturização e a lógica da lanterna (Spotlight).

- **Skybox**: Renderiza o cubo envolvente (céu).
- **Overlay**: Renderiza imagens 2D sobre a cena 3D (GUI).

3.3 Documentação (Doxygen)

O código fonte foi documentado utilizando comentários compatíveis com Doxygen. Foi gerada uma documentação HTML que descreve a hierarquia de classes, os métodos e as interdependências, facilitando a consulta técnica do projeto.

4 Conclusões

O desenvolvimento deste projeto permitiu consolidar os conhecimentos teóricos de Computação Gráfica.

Resultados Alcançados:

- Foi criado um ambiente 3D navegável com performance estável.
- O sistema de carregamento de mapas é flexível, permitindo trocar o labirinto apenas alterando o ficheiro .obj.
- A iluminação e as texturas conferem ao jogo a atmosfera desejada.
- O sistema de colisões, embora simples, é robusto o suficiente para a jogabilidade proposta.

Trabalho Futuro: Como melhorias futuras, planeia-se:

- Refinar o sistema de sombras (Shadow Mapping) para suportar sombras dinâmicas projetadas pela lanterna.
- Implementar um menu inicial interativo.
- Adicionar efeitos sonoros para passos e ambiente.

5 Bibliografia

[1] Joey de Vries, *LearnOpenGL*, `https://learnopengl.com/`.

[2] Khronos Group, *OpenGL 3.3 Reference Pages*, `https://www.khronos.org/registry/OpenGL-Refpages/gl4/`.

[3] G-Truc Creation, *OpenGL Mathematics (GLM)*, `https://glm.g-truc.net/0.9.9/index.html`.

[4] Syoyo Fujita, *TinyObjLoader*, `https://github.com/tinyobjloader/tinyobjloader`.