Universidade da Beira Interior

Licenciatura Engenharia Informática

Departamento de Informática

Elaborado por:

Rúben Ferraz nº45590
Pedro Hilário n45850
Ruben Serrano nº46325

Orientador:

Professor Doutor Abel J.P Gomes

Covilhã, 14 de janeiro de 2024

Agradecimentos

A realização do presente relatório não seria possível sem o apoio, contributo e esforço de várias pessoas que direta ou indiretamente influenciaram o rumo deste trabalho.

Assim sendo, pretendemos agradecer a todos que sempre nos encorajaram a seguir em frente e a fazer sempre melhor.

Ao Professor Doutor Abel J.P Gomes, pela sua orientação, total colaboração e disponibilidade, pelos seus ensinamentos que contribuem para o nosso enriquecimento educacional, por toda a ajuda prestada, não apenas neste trabalho, mas fundamentalmente ao longo de todo o semestre, sendo um privilégio e uma honra sermos seus educandos. A ele, o nosso muito Obrigado!

A todos os docentes que contribuem para a nossa formação e crescimento académico ao longo desta licenciatura, pelo saber que transmitem, os valiosos conhecimentos que nos incutem e pela excelência da qualidade técnica de cada um.

Aos nossos colegas de curso pela amizade e solidariedade, sempre disponíveis para nos ajudar, aconselhar e encorajar nos momentos cruciais desta caminhada estudantil.

Aos companheiros com os quais desenvolvemos este trabalho, pela dedicação, empenho inexcedível e saudavelmente exigível, pela partilha do esforço e dos inúmeros desafios que enfrentamos, sempre com o espírito colaborativo.

E por fim, cada elemento do grupo, agradece de forma especial à sua família, pela confiança no nosso progresso, pelo apoio incondicional e por nos incentivarem e apoiarem em todas as áreas da nossa vida.

A todos os nosso sincero e profundo Muito obrigado!

Conteúdo

Lista de Figuras

Capítulo

1

Introdução

1.1 Enquadramento, Âmbito e Motivação

Este projeto foi desenvolvido no âmbito da Unidade Curricular de Computação Gráfica [1] da Licenciatura em Engenharia Informática na Universidade da Beira Interior (UBI). Com o avanço contínuo e a necessidade de inovações na área da visualização computacional e interação humana com o computador, emerge a necessidade de criar soluções gráficas avançadas e intuitivas. À medida que a tecnologia evolui, torna-se essencial desenvolver ferramentas que não só resolvam problemas atuais mas também antecipem futuras necessidades em diversas áreas, como jogos, simulações, realidade virtual e aumentada.

Neste contexto, o presente projeto foca-se no desenvolvimento de sistemas gráficos inovadores, com ênfase na criação de ambientes virtuais realistas e interativos. Propõe-se a construção de uma aplicação de computação gráfica que permita aos utilizadores interagir com o ambiente virtual tridimensional de forma eficaz e realista. Este sistema deverá ser capaz de renderizar cenários complexos, gerir iluminação dinâmica e sombras, e também proporcionar uma experiência imersiva ao utilizador.

O objetivo principal é explorar e implementar técnicas avançadas de computação gráfica, como a renderização em tempo real, *shaders*, e algoritmos de iluminação, para criar uma plataforma robusta que possa ser utilizada em diferentes campos, incluindo educação, entretenimento e desenvolvimento profissional. Será dada especial atenção ao desenvolvimento de uma inter-

face intuitiva e a integração de elementos interativos que respondam a *inputs* do utilizador em tempo real.

A motivação para este projeto decorre da crescente demanda por soluções gráficas mais sofisticadas e imersivas, capazes de oferecer experiências mais ricas e realistas aos utilizadores. A implementação de técnicas avançadas de computação gráfica em aplicações diversas não só aumenta o potencial de inovação e criatividade mas também abre novas possibilidades para o design interativo e a visualização de dados.

Este projeto alinha-se com os objetivos desta Unidade Curricular, tirando partido dos conhecimentos adquiridos ao longo de toda a licenciatura (em programação, matemática) e ainda dos adquiridos nesta UC (OpenGL moderno). O trabalho será desenvolvido através da combinação da teoria e prática lecionadas para criar uma solução inovadora e eficaz que atenda aos requisitos impostos pelo docente. Com a conclusão deste projeto espera-se, não só aprimorar os conhecimentos teóricos e técnicos dos membros do grupo mas também contribuir significativamente para o avanço da computação gráfica aplicada a vários domínios.

1.2 Objetivos

Este projeto prático tem como objetivo principal desenvolver as capacidades gráficas e interativas de uma aplicação num ambiente virtual tridimensional. A movimentação pelo ambiente será realizada através do teclado de maneira a permitir que o utilizador possa explorar e interagir com o mundo criado. Durante a execução o utilizador deve ser capaz de visualizar detalhes gráficos realistas e interagir com os objetos e elementos do cenário. Para tal, é necessário implementar um conjunto de funcionalidades que habilitem a aplicação a renderizar cenários complexos e responder de maneira eficaz e precisa às ações do utilizador.

Os objetivos específicos do projeto incluem:

1. Implementar uma movimentação fluída e intuitiva pelo ambiente virtual.
2. Criar elementos gráficos, como iluminação, texturas e sombras.
3. Desenvolver uma interface de utilizador que permita interações simples e eficazes com o ambiente.

4. Modelação das peças originais do jogo Tetris.
5. Implementação dos movimentos descendentes das peças com as respetivas translações e rotações.
6. Otimizar o desempenho gráfico para garantir uma experiência de utilizador completa.
7. Texturização das peças.

Estes objetivos serão fundamentais para proporcionar uma experiência imersiva e interativa. O utilizador poderá então movimentar-se pelo cenário e interagir com os objetos enquanto desfruta de gráficos de alta qualidade e respostas realistas do sistema. Com este projeto pretende-se não só aprimorar as técnicas de computação gráfica lecionadas nesta Unidade Curricular mas também explorar novas possibilidades no design, interação e visualização digitais.

1.3 Organização do Documento

Para uma melhor e adequada interpretação do presente trabalho, o mesmo foi estruturado da seguinte maneira:

1. Primeiro Capítulo – **Introdução** - Este capítulo apresenta o contexto e o âmbito do projeto, delineando os objetivos gerais, o enquadramento temático e a estrutura organizacional do documento.
2. Segundo Capítulo – **Tecnologias e Ferramentas Utilizadas** - Descreve detalhadamente as ferramentas e bibliotecas selecionadas para o desenvolvimento da inteligência do Robô no ambiente virtual, justificando a escolha das mesmas.
3. Terceiro Capítulo – **Desenvolvimento e Implementação** - Apresenta em detalhe os algoritmos concebidos e os métodos utilizados na sua implementação, bem como as decisões tomadas pelo grupo durante o processo de desenvolvimento.
4. Quarto Capítulo – **Organização do Projeto** - Expõe a distribuição das tarefas pelos membros do grupo.
5. Quinto e Último Capítulo – **Conclusões e Trabalho Futuro** - Oferece uma reflexão sobre o trabalho desenvolvido e os conhecimentos adquiridos, bem como uma retrospectiva crítica do projecto com considerações sobre possíveis direcções para trabalho futuro.

Esta organização do trabalho visa facilitar um entendimento mais abrangente do projeto, permitindo uma compreensão clara dos processos, desafios e resultados associados ao seu desenvolvimento.

Capítulo

2

Ferramentas e Tecnologias Utilizadas

2.1 Introdução

Neste capítulo abordamos as ferramentas e tecnologias fundamentais selecionadas para o desenvolvimento do nosso projeto da Unidade Curricular de Computação Gráfica. A seleção apropriada de software e bibliotecas é crucial, pois cada componente escolhido tem um impacto direto na performance, na eficiência e na qualidade final do sistema. Através de uma análise criteriosa, optámos por um conjunto de tecnologias que são reconhecidas pela sua robustez, desempenho e compatibilidade no campo da computação gráfica.

2.2 Ferramentas e Tecnologias Utilizadas

Para a execução deste projeto, uma série de ferramentas e tecnologias específicas foram meticulosamente selecionadas, com o objetivo de atingir os melhores resultados possíveis. A escolha criteriosa destas ferramentas reflete as melhores práticas e inovações tecnológicas da área, sendo fundamental para o sucesso e eficácia do presente projeto. A seguir, descreve-se brevemente cada ferramenta e tecnologia utilizada, fornecendo um contexto para a sua seleção e o papel que desempenham no desenvolvimento do sistema:

2.2.1 OpenGL

Uma especificação que define uma API multiplataforma para gráficos 2D e 3D. É amplamente utilizada em aplicações que requerem renderização grá-

fica como jogos, simulações e programas de design. A sua escolha deve-se à sua eficiência, flexibilidade e constante atualização, oferecendo uma grande variedade de funcionalidades gráficas.[2]

2.2.2 Visual Studio Code (VSCode)

Um editor de código-fonte leve, mas poderoso, que suporta várias linguagens de programação. Foi escolhido pela sua versatilidade, vasta gama de extensões disponíveis e funcionalidades integradas que facilitam o desenvolvimento e depuração de código.[3]

2.2.3 GLM (OpenGL Mathematics)

Uma biblioteca de matemática para gráficos que segue as especificações do OpenGL Shading Language (GLSL). Esta ferramenta é usada para operações matemáticas, principalmente relacionadas a transformações de gráficos 3D.[4]

2.2.4 GLAD (OpenGL Loader Generator)

Uma ferramenta que gere a carga e inicialização de extensões OpenGL em várias plataformas. Esta foi escolhida por ser leve e oferecer uma maneira simples e direta de carregar todas as funções e extensões necessárias do OpenGL, garantindo a compatibilidade entre diferentes sistemas e versões do OpenGL.[5]

2.2.5 GLFW (Graphics Library Framework)

Uma biblioteca que permite a criação e gestão de janelas, bem como o manuseamento de inputs em aplicações OpenGL. A sua seleção deve-se à sua simplicidade e eficácia, facilitando a interação com o sistema operacional e a gestão de janelas e contextos de gráficos.[6]

2.2.6 C++

Uma linguagem de programação de alto nível com capacidades de baixo nível, escolhida pela sua performance, controle sobre recursos do sistema e vasto suporte a bibliotecas. Esta é ideal para o desenvolvimento de sistemas gráficos complexos permitindo assim um equilíbrio entre eficiência e facilidade de programação.[7]

A combinação de todas as tecnologias referidas acima fornece uma base sólida e versátil para o desenvolvimento do presente projeto de maneira detalhada, interativa e eficiente. Juntas essas tecnologias têm a particularidade de oferecer um ambiente de desenvolvimento que é ao mesmo tempo poderoso e adaptável às necessidades do projeto.

2.3 Conclusões

Neste capítulo, procedeu-se à descrição pormenorizada das tecnologias e ferramentas empregues na concretização e desenvolvimento do projeto em análise. As escolhas efetuadas serão referenciadas nos capítulos subsequentes, onde se detalhará a aplicação prática de cada uma na resolução das problemáticas abordadas pelo projeto. Esta secção visou não só evidenciar a importância de uma seleção tecnológica adequada, mas também estabelecer uma base para a compreensão integral de como as escolhas se ajustaram e contribuíram para o sucesso do projeto

Capítulo

3

Desenvolvimento e Implementação

3.1 Introdução

Este capítulo dedica-se a detalhar os algoritmos concebidos e os métodos adotados na implementação deste projeto, bem como as decisões estratégicas tomadas durante o processo de desenvolvimento do mesmo. Estas escolhas são cruciais para entender não só o seu funcionamento, mas também a eficácia das soluções propostas, refletindo assim a lógica e os objetivos por detrás de cada passo dado neste projeto. Vamos então explorar como foram aplicadas e adaptadas as técnicas de implementação já referidas para criar visualizações ricas e interativas, e como cada decisão impactou na estética e performance do sistema final. Este entendimento é essencial para apreciar a complexidade e a beleza do trabalho realizado e ainda para fundamentar as bases de conhecimento que permitirão futuras inovações na área.

3.2 Escolhas de Implementação

3.2.1 Implementar uma movimentação fluída e intuitiva pelo ambiente virtual

A movimentação pelo ambiente virtual deste projeto é manipulada principalmente pela classe `camera`. Esta classe é responsável pela posição e rotação da câmara dentro do ambiente virtual e ainda pela movimentação do utilizador. Abaixo podemos ter uma perceção de como esta movimentação é implementada:

Inicialização da Câmara:

No construtor da classe `camera`, a posição `pos` e rotação `rot` são inicializadas. Além disso, definimos variáveis do tipo inteiro para capturar o estado das teclas `w, a, s, d` que mais tarde iremos usar para a movimentação.

Método de Processamento da Câmara (process):

Este método é chamado a cada *frame* para atualizar a posição e rotação da câmara com base nos *inputs* do utilizador. O mesmo tem em consideração o tempo decorrido `ftime` do tipo `double` para garantir uma movimentação fluída.

Movimentação:

As teclas `W` e `S` são usadas para mover a câmara para frente e para trás, respetivamente. A velocidade é ajustada com base no tempo decorrido.

As teclas `A` e `D` rodam a câmara para a esquerda e para a direita. Similarmente à movimentação, a rotação é ajustada pelo tempo decorrido.

Atualização da câmara:

Este método atualiza as variáveis de estado da câmara (w, a, s, d) com base nas interações do utilizador com o teclado. As variáveis indicam se uma tecla específica está pressionada ou não. Quando uma tecla é pressionada a sua variável correspondente é marcada como 1, e 0 quando é solta.

Aplicação da Transformação da Câmara

A posição e rotação atualizadas da câmara são usadas para construir matrizes de transformação que são aplicadas ao ambiente virtual. Essas matrizes são:

Rotação: matriz de rotação em torno do eixo Y baseada na rotação atual da câmara. Translação: matriz de translação que posiciona a câmara no espaço 3D. O método `process` retorna uma matriz que é o resultado da multiplicação da matriz de rotação pela matriz de translação. Essa matriz representa a transformação total da câmara e é usada no método `render` para aplicar a perspetiva da câmara ao visualizar o ambiente.

3.2.2 Iluminação, texturas e sombras

Para a realização do Projeto utilizamos a integração de várias técnicas gráficas, incluindo iluminação avançada, mapeamento de texturas e cálculo de sombras. No código fornecido, esses aspectos são manipulados principalmente através de texturas e shaders.

Texturas:

Utilizamos a biblioteca stb_image.h para carregar imagens que são posteriormente ligadas a objetos no ambiente 3D. As IDs da textura (BlockTexId, BackgroundTexId, NormalTexId) representam diferentes superfícies e padrões que serão aplicados aos objetos renderizados. Essas texturas são essenciais para criar a percepção de detalhes sem a necessidade de geometrias complexas.

Shaders:

Os shaders são o coração da renderização, definindo como a luz interage com os materiais, como as texturas são aplicadas e como as sombras são formadas. Os nossos ficheiros responsávis por tais funcionalidades são os seguintes: `shader_vertex_block.glsl` e `shader_fragment_block.glsl`. Estes contêm a lógica essencial para a renderização. Eles são responsáveis por calcular a iluminação difusa e especular, mapeamento de normais, reflexões e técnicas de sombras como oclusão ambiental.

Os programas de shader (tetrominoProg, backgroundProg) e os seus atributos são usados para enviar informações relevantes dos objetos 3D, como posição, tipo e textura, para os shaders. Esses dados são cruciais para que os shaders possam executar os cálculos necessários para renderizar cada objeto com as características de iluminação e textura desejadas.

Renderização:

A função render() combina todas as técnicas mencionadas para criar a cena final. As matrizes de projeção e visão determinam a perspectiva e a posição da câmera, enquanto cada objeto é desenhado usando os shaders configurados com suas respectivas texturas e iluminação. A sequência de renderização cuida de aplicar todas as transformações, cálculos de iluminação e texturização, resultando em uma cena 3D.

3.2.3 Modelação das peças originais do jogo Tetris

Começámos por definir uma matriz de quatro dimensões de inteiros chamada `tetrominos` com a finalidade de ser possível a representação das diferentes formas que um tetromino pode assumir. A estrutura utilizada foi:

- O primeiro índice, '[7]', representa os 7 diferentes tipos de *tetrominos* (I, O, T, S, Z, J, L);
- O segundo índice,'[4]', representa as 4 rotações possíveis para cada *tetromino*, sendo cada uma delas de 90º;
- O terceiro e quarto índices, o par '[4][4]', representam uma matriz 4x4 que é usada para desenhar a forma de um *tetromino* numa determinada posição.

Cada bloco de código entre chavetas localizado dentro da matriz `tetrominos` define a forma de um tetromino numa rotação específica, por exemplo:

```
{
    {1, 1, 0, 0},
    {1, 1, 0, 0},
    {0, 0, 0, 0},
    {0, 0, 0, 0}
}
```

O bloco entre chavetas acima representado define a forma de um *tetromino* do tipo O (neste caso representa o quadrado) na sua posição inicial. Obviamente que, como se trata de um quadrado, ao serem aplicadas as rotações nele estas não serão visíveis. O número 1 representa uma parte ocupada do *tetromino* enquanto que o número 0 representa uma parte vazia. Esta lógica foi adotada para a definição das restantes peças, inclusive as suas respetivas rotações.

Relativamente ao desenho dos *tetrominos* utilizámos, como já foi referido em capítulos anteriores, o OpenGL para a renderização gráfica. Percorremos cada posição dentro de um bloco de tamanho definido (BLOCK_SIZE) que representa a área de jogo onde os *tetrominos* podem existir. Para cada posição dentro desse bloco é criado um objeto `Position` chamado `tetrominoPos`, que armazena as coordenadas atuais da célula a ser processada na matriz do *tetromino*.

De seguida fazemos uma verificação condicional para determinar se a célula atual contém parte de um *tetromino* (isto é, se o método `getValue` do objeto `currentTetromino` não retorna zero para a posição atual). Se a célula faz parte do *tetromino*, várias operações de transformação e renderização são realizadas:

`tetrominoTransMat = translate(mat4(1), vec3(...))`: Esta linha utiliza a função `translate` para aplicar uma transformação de translação à matriz de transformação do *tetromino*. A transformação é definida por um vetor tridimensional que centraliza e posiciona o *tetromino* na tela, com ajustes feitos com base nas posições das células.

`M = tetrominoTransMat * tetrominoScaleMat;`: Declaramos a matriz final de transformação M que será aplicada ao *tetromino*, matriz essa que é obtida multiplicando a matriz de transformação resultante `tetrominoTransMat` por uma matriz de escala `tetrominoScaleMat`.

Usámos chamadas `glUniformMatrix4fv` e `glUniform1i` para passar a matriz de transformação e informações adicionais ao *shader* que está a ser usado para desenhar o *tetromino*. Essas funções atualizam os valores das variáveis uniformes do *shader* `tetrominoProg`.

`cube->draw(tetrominoProg, false);`: Chamamos o método `draw` no objeto `cube`, que desenha o *tetromino* na tela através do *shader* especificado.

Por fim libertamos o *shader* da nossa função, `tetrominoProg->unbind()`, para evitar que o estado do *shader* afete outros objetos que possam ser desenhados posteriormente.

3.2.4 Mecanismos de Controlo e Dinâmicas de Jogo

Referente à mecânica do jogo Tetris, é imprescindível destacar a implementação do movimento descendente das peças, que simula o efeito da gravidade. Dentro do método de renderização, uma função específica verifica o passar do tempo para determinar se uma peça deve descer uma linha, incrementando a variável correspondente à fila a cada segundo. Este mecanismo assegura que a peça ativa desloque-se para baixo automaticamente, emulando a constância da gravidade.

É importante mencionar os movimentos laterais e a rotação das peças, funcionalidades igualmente essenciais. Estas ações são programadas na sequência do código que lida com o movimento descendente. O jogador tem a capacidade de mover a peça para a esquerda ou para a direita, mediante o clique das teclas correspondentes, o que ajusta a coluna da posição atual da peça. A verificação de possíveis colisões é imediata; se um movimento lateral resultar em uma colisão, ele é imediatamente anulado para prevenir sobreposições ou saídas do espaço de jogo. Paralelamente, a rotação da peça é facilitada pelo pressionar de outra tecla designada, que altera a orientação da peça. Esta ação também está sujeita a uma verificação de colisão, sendo que qualquer rotação que resulte em sobreposição é revertida para manter a integridade do jogo.

Um elemento interativo adicional é o movimento acelerado para baixo, ativado pela tecla de espaço, permitindo que a peça desça imediatamente até o ponto de colisão mais próximo. A funcionalidade de 'Hold' é igualmente notável, proporcionando ao jogador a opção de guardar uma peça para uso subsequente. Esta ação é realizada através da tecla "X"que, quando pressionada, substitui a peça em jogo por outra em espera.

O último tópico relevante para este segmento do relatório é a verificação de colisão e o armazenamento da peça no tabuleiro, que é implementado imediatamente após as verificações de movimento. O sistema assegura que, a cada movimento, seja verificado se a peça em jogo entra em contato com outras peças ou com os limites do tabuleiro. Se confirmada uma colisão, a peça é então fixada na sua posição atual dentro da estrutura do tabuleiro, tornando-se parte da configuração permanente do jogo.

Estas dinâmicas e mecanismos são a base da experiência do Tetris, conferindo ao jogador o controlo completo sobre a manipulação das peças, o que é fundamental para a estratégia e o ritmo do jogo. A fluidez e a resposta do jogo aos comandos do utilizador são resultado direto destas implementações, que são vitais para assegurar a autenticidade e a diversão do jogo.

3.3 Detalhes de Implementação

Nesta secção, detalhamos a implementação, específica, para cada um dos tópicos, apresentados no enunciado do projeto prático. A explicação foca nos algoritmos e nas lógicas utilizadas para fornecer respostas precisas e eficientes.

3.4 Conclusões

Ao longo deste capítulo, detalhou-se os procedimentos e metodologias adotados para o desenvolvimento deste projeto. Foi realizada uma exposição cuidadosa dos algoritmos implementados, das estruturas de dados escolhidas e das lógicas de programação utilizadas, todas estas direcionadas para a criação de visualizações tridimensionais ricas e interativas. Além disso, proporcionámos uma visão geral sobre a integração das diversas ferramentas e tecnologias que suportaram o desenvolvimento do projeto, as quais foram extensivamente discutidas no capítulo anterior.

Este capítulo serve como um reflexo do presente projeto, evidenciando não só os aspetos técnicos e as soluções adotadas mas também a importância de uma seleção criteriosa de ferramentas e a aplicação de métodos robustos. Assim, consolidámos a compreensão sobre como cada componente contribui para a funcionalidade e eficiência do sistema, reafirmando o compromisso do projeto com a inovação e a qualidade técnica no campo da visualização e computação gráficas.

Capítulo

4

Organização do Projeto

4.1 Introdução

Neste capítulo, procedeu-se com uma organização estruturada, que visa facilitar a compreensão e o acompanhamento dos diversos aspetos relacionados ao desenvolvimento do Projeto Prático.

Inicialmente, concedeu-se uma visão geral da disposição dos conteúdos, seguida de uma descrição detalhada, de como as tarefas foram distribuídas entre os membros do grupo. Prossegui-se com uma análise cuidadosa dos desafios e problemas encontrados, ao longo do projeto, fornecendo insights sobre as adversidades enfrentadas e as estratégias adotadas para sua resolução.

Em seguida, apresentou-se uma reflexão crítica acerca do projeto, onde discutiu-se as lições aprendidas, as conquistas e as áreas que poderiam ser melhoradas em futuras implementações. Por fim, o capítulo é concluído com uma síntese dos pontos discutidos, englobando as experiências e conhecimentos adquiridos durante a realização do projeto.

4.2 Divisão de Tarefas pelo Grupo

No âmbito do presente projeto, a divisão de tarefas, entre os membros do grupo, foi realizada com o objetivo de assegurar uma distribuição equitativa e justa. Essa divisão levou em consideração a complexidade e o tema de cada tarefa, procurando alinhar as habilidades e preferências individuais dos membros com as necessidades específicas do projeto. Porém, muito embora

pese a individualidade de cada tarefa, relevou-se a colaboração e o trabalho em grupo, especialmente na definição de aspetos cruciais como as estruturas de dados e as metodologias para obtenção e armazenamento de informações do mundo virtual.

Mesmo com tarefas realizadas individualmente, houve uma interação constante entre os membros do grupo, promovendo um ambiente de troca de ideias e suporte mútuo. O relatório final e a apresentação do projeto refletem essa colaboração, onde cada membro contribuiu com a documentação e explicação das partes que implementou.

A tabela a seguir ilustra a distribuição das tarefas, evidenciando como cada componente do grupo contribuiu para a realização do Projeto Prático.

	Rúben Ferraz	Pedro Hilário	Ruben Serrano
Objetivo 1		X	
Objetivo 2	X		
Objetivo 3			X
Objetivo 4		X	
Objetivo 5			X
Objetivo 6	X	X	X
Objetivo 7	X		
Relatório	X	X	X
PowerPoint	X	X	X

Figura 4.1: Tabela Contribuição

4.3 Conclusão

Este capítulo proporcionou uma análise abrangente e detalhada do trabalho desenvolvido pelo grupo ao longo do projeto. Através da discussão sobre a divisão de tarefas e as reflexões críticas, consegiu-se avaliar não apenas o progresso e as realizações do grupo, mas também as dificuldades enfrentadas e as lições aprendidas durante o processo.

As conclusões extraídas desta análise serão fundamentais para as próximas fases do trabalho, guiando as futuras decisões e estratégias do grupo. Os insights e experiências aqui documentados não apenas refletem o compromisso e o empenho do grupo, mas também estabelecem uma base sólida para futuras iniciativas e projetos. A próxima secção continuará a explorar as implicações dessas conclusões, delineando os passos e direções para avançar com base no conhecimento adquirido.

Capítulo

5

Conclusões Principais

A realização deste projeto representou uma oportunidade para o aprofundamento dos nossos conhecimentos em OpenGL e C++, e em conceitos fundamentais da Computação Gráfica. A experiência adquirida proporcionou uma perspetiva mais ampla sobre a criação e manipulação de ambientes virtuais, bem como as capacidades de renderização e interação em tempo real.

Além de desmistificar a complexidade inerente aos sistemas gráficos avançados, o projeto revelou as múltiplas possibilidades que a aplicação prática desses conhecimentos pode oferecer. Através do desenvolvimento e da implementação de ambientes virtuais detalhados e interativos, foi possível visualizar e compreender os desafios e os potenciais que acompanham a criação de visualizações tridimensionais ricas e a sua interação com o utilizador.

Esta jornada de aprendizagem não só reforçou a compreensão teórica e prática dos conteúdos lecionados nesta Unidade Curricular mas também estimulou o pensamento crítico e a resolução criativa de problemas nesta área. As experiências e conhecimentos adquiridos durante a execução deste projeto certamente vão servir de base para futuras investigações e desenvolvimentos, abrindo assim caminho para novas descobertas e inovações no universo da Computação Gráfica.

Capítulo

6

Trabalho Futuro

Para o seguinte Projeto, existem várias direções promissoras para o desenvolvimento futuro. Estas não só enriquecerão a experiência do utilizador mas também aprimorarão a eficiência da aplicação. Os seguintes pontos destacam as áreas chave para o trabalho futuro a realizar:

1. **Criação de um Menu Inicial**: Implementar um menu inicial é fundamental para uma boa experiência. Este menu deverá incluir opções como 'Iniciar Jogo', 'Configurações', 'Créditos' e 'Sair'. A adição de um sistema de menu contribuirá para uma navegação mais intuitiva e organizada, permitindo ao utilizador personalizar a sua experiência antes de iniciar o jogo.

2. **Adição de Trilha Sonora e Efeitos Sonoros**: A música e os efeitos sonoros são componentes chave para criar uma atmosfera imersiva. A implementação de uma trilha sonora apropriada e efeitos sonoros para ações específicas, como a movimentação e rotação das peças, melhorará significativamente a imersão do utilizador no ambiente virtual.

3. **Otimização do Código**: Para garantir uma experiência fluida e eficiente, é essencial otimizar o código da aplicação. Isto inclui a melhoria da gestão de memória, a otimização de algoritmos e a redução do tempo de carregamento. A otimização contribuirá para um desempenho mais estável e uma melhor experiência.

4. **Correção de Bugs**: Uma fase contínua de testes e correções de bugs é vital. Deve-se identificar e resolver quaisquer problemas técnicos, glitches ou falhas que possam afetar a jogabilidade ou a experiência do utilizador. A correção de bugs é um processo contínuo que garantirá a estabilidade e confiabilidade da aplicação.

Em conclusão, estas melhorias e expansões propostas são fundamentais para elevar a qualidade e aprofundar a experiência oferecida pelo nosso projeto. Através da implementação destes desenvolvimentos, podemos assegurar que a aplicação não só atende às expectativas atuais, mas também está preparada para as demandas futuras.

Bibliografia

[1] Abel Gomes. Projeto, 2024. [Online] https://www.di.ubi.pt/~agomes/cg/.

[2] OpenGL. OpenGL, 2024. [Online] https://www.opengl.org/.

[3] VSCode. VSCode, 2024.

[4] GLFM. GLM, 2024. [Online] https://glm.g-truc.net/0.9.9/index.html.

[5] GLAD. GLAD, 2024. [Online] https://glad.dav1d.de/.

[6] GLFW. GLFW, 2024. [Online] https://www.glfw.org/.

[7] Wikipedia. C++, 2024. [Online] https://pt.wikipedia.org/wiki/C%2B%2B.