Universidade da Beira Interior
Departamento de Informática

Licenciatura em Engenharia Informática

Computação Gráfica

Projeto Prático

Elaborado por:

Samuel Dias nº48184 , João Domingues nº48897, Pedro Batista nº48389, Pedro Lourenço nº48213

Orientador:

Professor Doutor Abel Gomes

Covilhã, 11 de janeiro de 2024

Conteúdo

Lista de Figuras

Bibliografia

Capítulo

1

Introdução

1.1 Enquadramento

O presente relatório foi elaborado por alunos do 3.° ano da Licenciatura de Engenharia Informática da Universidade da Beira Interior. Este foi desenvolvido no contexto da Unidade Curricular de Computação Gráfica, mediante pesquisas pessoais e material pedagógico providenciado pelo Professor da própria UC.

1.2 Motivação

A Computação Gráfica é uma área de estudo que permite a criação de imagens e animações de alta qualidade, onde conseguimos obter a representação de objetos e cenários de forma realista e detalhada. O Sistema Solar é um tema interessante e daí nasce a vontade em realizar este trabalho. Além disso, a criação de um sistema solar em 3D também pode ser utilizada como ferramenta educacional, que via permitir que as pessoas tenham uma representação mais clara e concreta de como ele funciona. Portanto, a realização deste projeto de Computação Gráfica sobre o Sistema Solar em 3D é uma oportunidade de explorar a Astronomia.

1.3 Objetivos

Neste projeto pretende-se implementar um sistema solar interativo tanto em 2D como em 3D. Para isso, iremos utilizar a linguagem C++, bem como as bibliotecas OpenGL, GLFW, GLAD e GLM. Características da aplicação gráfica:

- Modelagem do sol, dos planetas e dos seus respetivos satélites (por exemplo, a lua em relação à terra).
- Menus para obter informações sobre os elementos do sistema solar (por exemplo, planetas).
- Aplicação das respetivas texturas dos planetas para aumentar o realismo.
- Capacidade da câmera movimentar-se em qualquer posição do sistema solar, afetando a luz que incide sobre os elementos.
- Utilização do teclado para fazer operações de *zoom in* e *zoom out* no sistema solar e alterar a vista.
- Iluminação através de um foco de luz, ou seja, o sol.
- Cálculo das sombras que os planetas e os satélites podem originar entre si.

1.4 Abordagem

Para abordar o tema do Sistema Solar neste projeto de Computação Gráfica, propomos seguir os seguintes passos:

- Realizar uma pesquisa aprofundada sobre o Sistema Solar, incluindo informações sobre as características de cada planeta, respetivas orbitas e movimentos, entre outros pormenores;
- Escolher um software de Computação Gráfica que seja adequado para a criação de modelos 3D e animações;
- Começar a criação dos modelos dos planetas, utilizando as informações obtidas na pesquisa para garantira precisão dos mesmos;
- Aplicar texturas e materiais realistas nos modelos, de maneira a torná-los o mais parecidos possível com os corpos celestes reais;
- Implementar animações que representem o movimento dos planetas ao redor do Sol e as respetivas rotações;
- Testar o sistema solar em 3D criado para garantir a qualidade e funcionamento;
- Redigir o relatório, apresentando os resultados da pesquisa e a metodologia utilizada para criar o sistema solar.

1.5 Organização do Documento

Para explicar de forma concisa o trabalho feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – Introdução – refere o projeto que nos foi proposto, o seu âmbito e Enquadramento, a Motivação para a sua realização, os objetivos e a respetiva organização do documento.

2. O segundo capítulo – Estado da Arte, Desenvolvimento e Aplicações – descreve as ferramentas utilizadas na realização deste projeto;

3. O terceiro capítulo – Ferramentas e Funcionalidades – abordada o Histórico de funcionalidades do projeto;

4. O quarto capítulo – Desenvolvimento – apresenta as dependências que existem no programa e os detalhes de implementação;

5. O quinto capítulo – Testes e Validação – explicita todos os Testes e Aprimoramentos realizados ao programa;

6. O sexto capítulo – Engenharia de Software – explica como são estruturados os dados e o ciclo de vida da aplicação;

7. O sétimo capítulo – Conclusões e Trabalho Futuro – como o nome indica, apresenta uma breve conclusão

Capítulo

2

Estado da Arte e Trabalhos Relacionados

Este capítulo apresenta de forma muito resumida as aplicações e ferramentas usadas no desenvolvimento deste projeto, os seus detalhes de implementação e uma breve descrição da plataforma Visual Studio. O capítulo divide-se do seguinte modo:

- Overleaf, uma plataforma de edição de texto, na secção 2.1 ;
- Trello, breve descrição do website, na 2.2;
- Visual Studio, o IDE utilizado, na secção 2.3
- Trabalhos Relacionados na secção 2.4

2.1 Overleaf

O Overleaf é uma plataforma online para escrita de documentos científicos, que inclui recursos para facilitar a criação de artigos, relatórios, apresentações e muito mais. É especialmente útil para escritores científicos, mas igualmente para nós, estudantes, pois permite-nos trabalhar no mesmo documento, juntos, em tempo real, compartilhar ideias e receber *feedback*. Oferece ainda a integração com várias bases de dados e repositórios de código, o que facilita a incorporação de Citações e Referências. O Overleaf foi essencial na escrita do Relatório Final, pois possui características que se destacam de programas de edição de texto, como o Microsoft Word, tais como:

- Capacidade de auto-completar funções;

- Permitir a compilação do trabalho várias vezes, para estar sempre atualizado e não se perder dados;
- Na eventualidade de existirem erros, indica os mesmos de forma precisa.

Figura 2.1: Logótipo do OverLeaf

2.2 Trello

O Trello é uma ferramenta de gestão de projetos que permite que as equipas criem quadros virtuais para organizar e priorizar as tarefas. Para o nosso projeto, o Trello foi essencial para dividir tarefas para a realização do projeto. Além disso, o Trello permite que as equipas personalizem os quadros e os cartões com etiquetas, lembretes e outros recursos para gerir o projeto mediante as necessidades. Caso pretendam, podem aceder à nossa área de trabalho através deste deste link.

Figura 2.2: Logótipo do Trello

2.3 Visual Studio

O Visual Studio é um Integrated Development Environment (IDE) que oferece uma ampla variedade de recursos e ferramentas para ajudar os desenvolvedores a criar, depurar e publicar aplicativos de maneira rápida e eficiente. Em suma, algumas das suas principais características incluem:

- Suporte a várias linguagens de programação;
- Depuração e teste;

- Possui ferramentas de design para ajudar os utilizadores a criar interfaces atraentes e intuitivas.

Figura 2.3: Logótipo do Visual Studio

2.4 Trabalhos Relacionados

Este trabalho, está diretamente relacionado com o projeto fornecido pelo professor Abel Gomes. Nome do Projeto: solarSystem Autor: Yonghui Rao Implementação:

- Cada um dos objetos é desenhado como uma esfera usando as texturas
- Os planetas giram em torno da estrela(sol), cada um tem o seu próprio caminho e a sua própria velocidade;
- A barra de espaços pode ser usada para pausar e retomar a animação;
- A cena é iluminada por uma fonte de luz localizada no centro da estrela.
- A própria estrela é iluminada por uma luz;
- Quando o rato está sobre um planeta, ele fica com um destaque brilhante.
- ID planetas:
    1. Branco - 0(sol)
    2. Amarelo - 1(Mercúrio)
    3. Rosa - 2(Vénus)
    4. Azul - 3(Terra)
    5. verde - 4(Marte)
    6. vermelho - 5(Júpiter)
    7. Teclas:
       Espaço - Stop

Capítulo

3

Tecnologias Utilizadas

No presente capítulo serão descritas as tecnologias utilizadas na concessão do projeto.

3.1 OpenGL

Open Graphics Library (OpenGL) é um software que funciona como uma interface de programação de aplicações (API) usada principalmente para renderização $2D$ e $3D$. Esta API é normalmente utilizada para renderizar aplicações recorrendo à aceleração de *hardware*, utilizando uma unidade de processamento gráfico (GPU).

Esta tecnologia foi originalmente desenvolvida pela "Silicon Graphics Inc." (SGI), em 1992, e é uma tecnologia recorrentemente usada para criar gráficos com elevado desempenho e qualidade multiplataforma, incluindo computadores, consolas de jogos e dispositivos móveis.

O OpenGL, como qualquer outra API, facilita o desenvolvimento do software ao programador, facilitando diversos comandos para renderizar pontos, linhas, triângulos, quadrados, objetos, texturas, iluminação, sombras, *shaders*, e entre outros.

Figura 3.1: Logótipo da API OpenGL

3.2 GLM

OpenGL Mathematics (GLM) é uma biblioteca de matemática desenvolvida na linguagem de programação C++ com o fim de ser usada em conjunto com o OpenGL, baseada na especificação *OpenGL Shading Language* (GLSL). Esta tecnologia fornece classes e funções matemáticas especialmente adaptadas para software gráfico alinhado com a ferramenta OpenGL, descrita na secção anterior.

Além disso, o GLM não se limita às características da especificação GLSL. Esta tecnologia, seguindo as convenções do GLSL, expande as suas capacidades, incluindo transformações de matriz, tipos de precisão reduzida, números aleatórios, entre outros recursos.

Figura 3.2: Logótipo da biblioteca GML

3.3 GLFW

Graphics Library Framework (GLFW) é uma biblioteca de código aberto e multiplataforma destinada ao desenvolvimento de software gráfico em "OpenGL", "OpenGL ES" e "Vulkan" em ambientes desktop. A sua interface simples possibilita a criação de janelas, contextos e superfícies, além de permitir a receção de entrada do utilizador e eventos. Escrita em C, esta biblioteca oferece suporte aos principais sistemas operativos.

Figura 3.3: Logotipo da biblioteca GLFW

3.4 Glad

A biblioteca "Glad" é uma ferramenta utilizada para gerir as funções do OpenGL em aplicações modernas, permitindo que estas sejam executadas em diferentes plataformas sem problemas de compatibilidade, dado que diferentes sistemas operativos e controladores de GPU podem fornecer suporte a diferentes versões do OpenGL. Esta biblioteca simplifica o processo de carregamento dessas funções, garantindo que os desenvolvedores utilizem os recursos mais recentes do OpenGL nas suas aplicações eficientemente e sem problemas de compatibilidade.

3.5 Assimp

Uma biblioteca muito usada para carregar e processar modelos 3D de uma variedade de formatos de ficheiro é a "Assimp", que significa "Open Asset Import Library".

O Assimp consegue importar dezenas de diferentes formatos de arquivo de modelos (e exportar para alguns também) carregando todos os dados do modelo nas estruturas de dados generalizadas da própria biblioteca. Como a estrutura de dados do Assimp permanece a mesma, independentemente do tipo de formato de arquivo que importamos, ele nos abstrai de todos os diferentes formatos de arquivo disponíveis.

Esta biblioteca oferece suporte a uma ampla gama de formatos de ficheiros de objetos 3D, como ficheiros do tipo "OBJ", "FBX", "COLLADA", entre outros, convertendo-os para uma estrutura de dados comum que pode ser facilmente manipulada e renderizada no OpenGL. Com a Assimp, os desenvolvedores podem carregar modelos complexos, incluindo informações sobre

geometria, texturas, materiais e hierarquias de objetos, facilitando a incorporação desses modelos em cenários e ambientes criados com OpenGL.

Capítulo

4

Etapas de desenvolvimento

4.1 Introdução

Neste capítulo, iremos descrever as fases de desenvolvimento do projeto para visualizar o sistema solar em 3D. Utilizando OpenGL, iremos apresentar a distribuição de tarefas entre os membros do grupo. Posteriormente, abordaremos as principais etapas do desenvolvimento do programa.

4.2 Divisão de tarefas

Para este projeto, as tarefas foram distribuídas da seguinte maneira:

- Planetas: Pedro Lourenço
- Texturas: Samuel Dias
- Órbitas: Pedro Batista
- Skybox: João Domingues

4.3 Procedimento Efetuado

Durante o desenvolvimento do projeto, foi imprescindível considerar alguns procedimentos.

4.3.1 Conhecimento das Bibliotecas OpenGL, GLFW, GLAD e GLM

Primeiramente, reservamos um período para adquirir um conhecimento fundamental sobre as bibliotecas OpenGL, GLFW, GLAD e GLM. Isso incluiu a compreensão do funcionamento de cada biblioteca, desde a criação de formas geométricas até a aplicação de transformações geométricas e iluminação no ambiente. Muito deste trabalho foi efetuado ao longo do semestre.

4.3.2 Escala do Sistema Solar e Distâncias

Foi de extrema importância definir o tamanho e a escala do sistema solar. Isso envolveu uma pesquisa das dimensões dos planetas em relação ao Sol, bem como a distância que os separa.

4.3.3 Desenho dos Planetas e Adição de Texturas

Os planetas foram concebidos com base na forma geométrica da esfera. Posteriormente, acrescentamos texturas aos planetas, utilizando ficheiros .jpg para textura disponíveis no link fornecido no enunciado do projeto.

4.3.4 Adição da Lua ao Planeta Terra

Foi realizado um passo crucial ao incorporar a Lua ao nosso planeta, a Terra. Para tal, utilizou-se o mesmo método empregado na criação dos próprios planetas.

4.3.5 Adição de Movimento ao Cenário

Fizemos a implementação do movimento do cenário, utilizando rotações e translações para simular o deslocamento dos planetas em torno do Sol e o deslocamento da Lua ao redor da Terra.

4.3.6 Adição de Iluminação e Sombras

Introduzimos iluminação e sombras ao cenário.

4.3.7 Configuração da Câmara e Ampliação/Redução

Permitimos que a câmara possua múltiplas posições e incluímos a capacidade de ampliar e reduzir. Isso ofereceu ao utilizador diversas perspetivas do sistema solar.

4.3.8 Adição de Texto Informativo

Adicionamos conteúdo informativo no terminal para tornar a experiência mais interativa, disponibilizando dados relevantes sobre os astros.

4.4 Imagem Final do Projeto

Nesta parte, iremos mostrar uma imagem conclusiva do projeto que demonstra o resultado obtido.

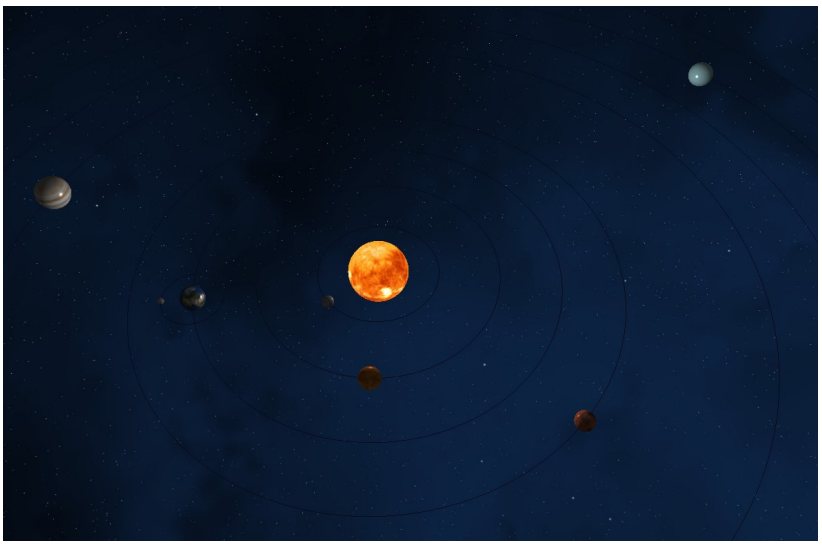

Figura 4.1: Representação visual final do sistema solar interativo.

Capítulo

5

Desenvolvimento e Implementação

5.1 Introdução

Para melhor perceção do código elaborado preferimos dividir este capítulo em duas partes, sendo elas:

- **Secção 5.2, Dependências e Includes:** Justificar as dependências do projeto;
- **Secção 5.3, Detalhes de Implementação:** Caracteriza e descreve cada ambiente gráfico desenvolvido.

5.2 Depêndencias e Includes

O programa possui algumas dependências, como os importes das seguintes bibliotecas:

- <glad/glad.h> - gerencia funções de pointers para o OpenGL,especialmente para a luz;
- <GLFW/glfw3.h> - é utilizada para criar janelas, contextos, receber instruções do utilizador e para eventos do programa;
- <glm/glm.hpp> -usada para funções matemáticas;
- <iostream> -habilita recursos que permitem interagir com o user por através de inputs do teclado
- <vector> - é usada para disponibilizar arrays de tamanho indefinido

- <math.h> - auxílio para funções e valores matemáticos.
- <stb_image.h> - possui funções que permitem usar imagens como texturas de objetos criados;
- <shader_m.h> - usado para ler os shader utilizados;
- <camera.h> - fornece um objeto Camera, permitindo à camera movimentar-se pelo Sistema Solar;
- <model.h> - carrega modelos 3D a partir de ficheiros .obj.

5.3 Detalhes da Implementação

5.3.1 Corpos celestes

Para formar os planetas (à exceção do planeta Saturno) decidiu-se criar objetos esféricos, sendo as suas texturas armazenadas em variáveis dedicadas a cada planeta. De seguida, atribui-se a textura ao planeta correspondente, usam-se matrizes para calcular a posição da Terra no espaço 3D e aplicam-se transformações (translação, rotação e escala) à matriz de modelo, para cada planeta/satélite.
No caso de Saturno, usou-se um objeto mais específico, já com os anéis que o rodeiam. Depois, efetuaram-se todas as alterações que se aplicaram aos outros planetas.

5.3.2 SkyBox

5.3.2.1 O que é?

Uma SkyBox é um método de criar planos de fundo para fazer com que o um espaço 3D parece maior do que aquilo que ele realmente vem a ser. É uma técnica bastante explorada por empresas de videojogos e por nós.

5.3.2.2 Como foi usada?

Como é tradicional das SkyBox, foi criado um cubo, na qual foram inseridas as texturas do Universo em cada face. Usa-se um shader mais adequado para a SkyBox, e, à medida que a camera se vai mover, as texturas são atualizadas de modo a dar a ilusão de profundidade infinita.

```
float skyboxVertices[] = {
    // positions
    -1.0f,  1.0f, -1.0f,
    -1.0f, -1.0f, -1.0f,
     1.0f, -1.0f, -1.0f,
     1.0f, -1.0f, -1.0f,
     1.0f,  1.0f, -1.0f,
    -1.0f,  1.0f, -1.0f,

    -1.0f, -1.0f,  1.0f,
    -1.0f, -1.0f, -1.0f,
    -1.0f,  1.0f, -1.0f,
    -1.0f,  1.0f, -1.0f,
    -1.0f,  1.0f,  1.0f,
    -1.0f, -1.0f,  1.0f,

     1.0f, -1.0f, -1.0f,
     1.0f, -1.0f,  1.0f,
     1.0f,  1.0f,  1.0f,
     1.0f,  1.0f,  1.0f,
     1.0f,  1.0f, -1.0f,
     1.0f, -1.0f, -1.0f,

    -1.0f, -1.0f,  1.0f,
    -1.0f,  1.0f,  1.0f,
     1.0f,  1.0f,  1.0f,
     1.0f,  1.0f,  1.0f,
     1.0f, -1.0f,  1.0f,
    -1.0f, -1.0f,  1.0f,

    -1.0f,  1.0f, -1.0f,
     1.0f,  1.0f, -1.0f,
     1.0f,  1.0f,  1.0f,
     1.0f,  1.0f,  1.0f,
    -1.0f,  1.0f,  1.0f,
    -1.0f,  1.0f, -1.0f,

    -1.0f, -1.0f, -1.0f,
    -1.0f, -1.0f,  1.0f,
     1.0f, -1.0f, -1.0f,
     1.0f, -1.0f, -1.0f,
    -1.0f, -1.0f,  1.0f,
     1.0f, -1.0f,  1.0f
};
```

Figura 5.1: Estruturação dos vértices da SkyBox

5.4 Órbitas dos corpos celestes

Inicialmente, cria-se um vetor para armazenar os valores das órbitas, que serão posteriormente colocados no buffer. Durante o ciclo while, consultamos os valores presentes no buffer, itera-se sobre os oito planetas e realiza-se as translações das órbitas para a posição de cada planeta. Em seguida, enviamos os dados necessários para os shaders e procedemos com o desenho no ecrã. Posteriormente, para a órbita da Lua, realizamos a translação de forma a ter a Terra como centro da órbita e enviamos os dados correspondentes para os shaders para desenhar no ecrã.

```
// ################################### orbitas ###################################
glBindVertexArray(VAO_t);

// mudar a cor da orbita
glm::mat4 modelorb = glm::mat4(1.0f);



for (int i = 1; i < 9; i++)
{
    modelorb = glm::mat4(1);
    // iniciar no meio para fazer o circulo conforme o meio
    modelorb = glm::translate(modelorb, posicao_centro);
    if (i <= 4) {
        modelorb = glm::scale(modelorb, glm::vec3(i * 2.f, i * 2.f, i * 2.f));
    }
    else {
        modelorb = glm::scale(modelorb, glm::vec3(i * 2.5f, i * 2.5f, i * 2.5f));
    }
    sphereShader.setMat4("model", modelorb);
    glDrawArrays(GL_LINE_LOOP, 0, (GLsizei)orbVert.size() / 3);

}

modelorb = glm::mat4(1.0f);
modelorb = glm::translate(modelorb, posicaoPlanetas[2]);
modelorb = glm::scale(modelorb, glm::vec3(0.5f * 2.f, 0, 0.5f * 2.f));
sphereShader.setMat4("model", modelorb);
glDrawArrays(GL_LINE_LOOP, 0, (GLsizei)orbVert.size() / 3);
```

Figura 5.2: Excerto de código dedicado às órbitas

5.5 Movimentação da Câmara

Todas as operações da câmara usam como recurso o ficheiro camera.h. Este ficheiro permite que a câmara:

- se movimente no espaço 3D;
- aumente ou diminua o Zoom
- se anexe a um dado planeta.

```
// processar todos os inputs do usuario
// -----------------------------------------------------------------------------
void processInput(GLFWwindow* window)
{
    if (glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS)
        glfwSetWindowShouldClose(window, true);

    if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS)
        camera.ProcessKeyboard(FORWARD, deltaTime);
    if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS)
        camera.ProcessKeyboard(BACKWARD, deltaTime);
    if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS)
        camera.ProcessKeyboard(LEFT, deltaTime);
    if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS)
        camera.ProcessKeyboard(RIGHT, deltaTime);
    // zoom in com o z e zoom out com o x
    if (glfwGetKey(window, GLFW_KEY_X) == GLFW_PRESS)
    {
        float Zoom = camera.Zoom;
        if (Zoom > 45.0f)
            camera.Zoom = 45.0f;
        else {
            camera.Zoom += 0.2f;
        }
    }
    if (glfwGetKey(window, GLFW_KEY_Z) == GLFW_PRESS)
    {
        float Zoom = camera.Zoom;
        if (Zoom < 1.0f)
            camera.Zoom = 1.0f;
        else {
            camera.Zoom -= 0.2f;
        }
    }
}
```

Figura 5.3: Excerto de código da câmara

5.6 Conclusão

Encontrámos como principal dificuldade a criação do planeta Saturno, tanto na estipulação da sua forma como na atribuição das texturas adequadas. Porém, uma vez ultrapassados estes bloqueios, conseguimos, com grande satisfação, atingir um Sistema Solar que representasse aproximadamente o verdadeiro.

Capítulo

6

Conclusões e Trabalho Futuro

6.1 Principais Conclusões

A realização deste projeto permitiu criar uma representação em 3D do Sistema Solar de forma realista e detalhada. Através da utilização de técnicas de Computação Gráfica e de uma pesquisa aprofundada sobre o Sistema Solar, foi possível criar modelos 3D precisos dos planetas e dos respetivos satélites. Além disso, a implementação de texturas, materiais realistas e a implementação das animações que executam o movimento dos planetas contribuíram para tornar a representação ainda mais realista e detalhada. Os recursos interativos, como as informações sobre cada elemento do sistema solar, também foram importantes para tornar o sistema solar uma ferramenta útil e eficaz. Em resumo, o sistema solar criado neste projeto é uma excelente maneira de aumentar o conhecimento e a compreensão acerca do sistema solar.

6.2 Trabalho Futuro

O Sistema Sola como conhecemos pode não ser o Sistema Solar que podemos conhecer amanha. Portanto, é provável haver mais detalhes e informações para se representar neste projeto. Além disso, existem outras áreas relacionadas com a Computação Gráfica que podem ser exploradas no futuro, como a criação de uma realidade virtual, ou a utilização de técnicas de *Machine Learning* para gerar uma representação mais realista. Em suma, há muitas possibilidades de concretização de projetos futuros relacionadas com o tema. O mais importante continuar a pesquisar e a explorar novas técnicas e tecnologias para poder criar representações com mais detalhadas do nosso sistema solar.