Universidade da Beira Interior
Departamento de Informática

Computação Gráfica - 2024: *Representação do Sistema Solar*

Elaborado por:

Artur Putyato, a47948
Gabriel Lázaro, a48396
Leandro Serra, a47941
Miguel Marques, a48011

Orientador:

Professor Doutor Abel Gomes

10 de janeiro de 2024

Agradecimentos

A realização deste trabalho, bem como a concretização académica dos envolvidos, não seria possível sem a ajuda do Professor Doutor Abel Gomes, que nos transmitiu o conhecimento necessário orientando-nos de forma clara e precisa para o desenvolvimento deste mesmo projeto. Deve-se também a todos os outros Professores que, de uma maneira ou de outra, nos transmitiram algum tipo de conhecimento que acaba sempre por se tornar necessário e finalmente, a todas as nossas famílas e amigos que nos dão o suporte para conseguirmos.

Conteúdo

Lista de Figuras

Acrónimos

UBI Universidade da Beira Interior

CG Computação Gráfica

Capítulo

1

Introdução

1.1 Enquadramento

Este projeto enquadra-se na unidade curricular de Computação Gráfica (CG), numa altura onde cada vez mais se denotam avanços nesta área, embora muito ligados à inteligência artificial, que já consegue gerar imagens sem qualquer apoio humano. Com este projeto, e de modo a sermos capazes de seguir e compreender tais avanços, conhecimentos de computação gráfica são fundamentais!

1.2 Motivação Universidade da Beira Interior (UBI)

Motivados por um semestre a ser lecionados pelo Professor Doutor Abel Gomes, que nos orientou no desenvolvimento de trabalhos e tarefas que nos permitiram aprender conceitos sobre desenvolvimento gráfico, necessitávamos de por em prática todo o conhecimento que obtivemos pelo que não existe melhor desafio para tal do que este projeto.

1.3 Objetivos

Este projeto tem como objetivo criar uma representação gráfica do sistema solar com recurso ao OpenGl e várias ferramentas e bibliotecas associadas. Utilizando este suporte, o desenvolvimento em C++ incorpora conceitos de rendering, manipulação de objetos e texturas, controlos de camara e utilização de shaders para proporcionar uma experiência visual o mais envolvente

possível. O objetivo em projetos deste tipo tem geralmente um ponto em comum, que é o seu objetivo. Este passa por criar uma imagem não distinguível de qualquer fotografia ou visão que possamos adquirir. Este ainda não é concretizável, mas a evolução desta área nesse sentido caminha.

O tema que escolhemos para este projeto foi o de representar, com as ferramentas mencionadas e tentando chegar o mais próximo possível do objetivo abordado, o sistema solar. A representação implementada exibe, de forma fidedigna, uma aplicação interativa do sistema solar em escala, incorporando caraterísticas realísticas, como rotação e órbita. Ao longo deste relatório são abordados vários conceitos dos principais componentes deste código e implementação, desde a configuração inicial do projeto até a renderização de cada planeta. Além disso, serão discutidas as estratégias adotadas para o controle de entrada do usuário, bem como a implementação de conceitos-chave, como transformações de matrizes e carregamento de texturas.

1.4 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a escolha do tema, o enquadramento, os seus objetivos e a respetiva organização do documento.

2. O segundo capítulo – **Estado da Arte** – enquadra o projeto e a área de inserção do mesmo.

3. O terceiro capítulo – **Tecnologias Utilizadas** – descreve os conceitos mais importantes no âmbito deste, bem como as tecnologias utilizadas no desenvolvimento da aplicação.

4. O quarto capítulo – **Implementação e Testes** – descreve a implementação da nossa solução para o trabalho escolhido.

Capítulo

2

Estado da Arte

2.1 Introdução

No contexto do desenvolvimento visual de sistemas, este capítulo explora as mais recentes inovações. A análise abrange desde técnicas avançadas de renderização até a interatividade melhorada, proporcionando uma visão abrangente do estado da arte nesse fascinante domínio.

2.2 Estado da Arte

O panorama atual no desenvolvimento de representações tridimensionais de sistemas astronómicos revela uma convergência fascinante entre avanços tecnológicos e a procura por experiências visuais imersivas. A utilização de frameworks gráficos robustos, como o OpenGL, tem sido um pilar essencial para a materialização dessas visualizações. Projetos contemporâneos abordam não apenas a precisão científica na modelagem de corpos celestes, mas também buscam cativar o usuário através de técnicas avançadas de renderização e interatividade.

A simulação precisa de órbitas planetárias, detalhes de superfície e fenómenos astronómicos complexos tem sido um foco proeminente. A representação realista de fenómenos atmosféricos, como auroras e eclipses, bem como a exploração detalhada de superfícies planetárias, são características distintivas de projetos de vanguarda. A interação do usuário é frequentemente melhorada por meio de controles intuitivos, possibilitando a exploração livre do espaço sideral e a compreensão mais profunda dos eventos cósmicos.

Além disso, a inclusão de técnicas de otimização gráfica e o emprego de bibliotecas especializadas para carregamento de texturas e manipulação matricial têm elevado a qualidade visual dessas simulações. A tendência de incorporar elementos educacionais, oferecendo informações contextualizadas sobre cada corpo celeste, destaca a busca por uma fusão entre entretenimento e aprendizado.

Contudo, desafios persistem, incluindo a demanda por maior realismo nas representações, a otimização do desempenho para lidar com vastos conjuntos de dados astronómicos e a integração de descobertas científicas mais recentes. Este projeto apresentará uma revisão aprofundada dessas tendências, inovações e desafios, situando o presente trabalho dentro do contexto dinâmico e em constante evolução da visualização de sistemas astronómicos em ambientes digitais.

2.3 Conclusão

Em suma, podemos entender este capítulo como uma fusão de avanços tecnológicos e a procura de experiências visuais cada vez melhores. Destaca-se a precisão científica, na ordenação correta dos planetas, por exemplo, também a aplicação de técnicas avançadas e a integração de elementos visuais que procuram e visam o realismo.

Capítulo

3

Tecnologias e Ferramentas Utilizadas

3.1 Introdução

Neste capítulo, exploramos as tecnologias e ferramentas fundamentais que impulsionam a implementação do projeto.

3.2 Conceito

Este projeto usa o OpenGL (Open Graphics Library), uma API gráfica de código aberto aplamente utilizada no desenvolvimento de aplicações gráficas. O OpenGL proporciona uma interface padrão para interação com placas gráficas, permitindo a criação de ambientes visuais complexos e interativos. Embora a explicar com mais detalhes nos capítulos de configuração do ambiente de desenvolvimento, existem 4 bibliotecas essenciais para configurar com sucesso o ambiente de desenvolvimento, sendo estas:

- **GLEW** (OpenGL Extension Wrangler Library) : Biblioteca que facilita a gestão de extensões OpenGL. Fornece maneiras de interagir com a GPU
- **GLFW** (Graphics Library Framework) : biblioteca que oferece suporte para a criação de janelas e manipulação de eventos em ambientes OpenGL
- **GLM** (OpenGL Mathematics) : ferramenta essencial para manipulações matemáticas em aplicações OpenGL. Neste projeto, a GLM é utilizada para operações matriciais, transformações de coordenadas e cálculos relacionados à geometria tridimensional.

- **STB Image** : biblioteca leve para carregamento de imagens. Neste projeto, ela é empregada para carregar texturas utilizadas na representação dos planetas do sistema solar.

3.3 Conclusão

A combinação dessas tecnologias e ferramentas proporciona um ambiente robusto para o desenvolvimento de aplicações gráficas tridimensionais, permitindo a criação de uma representação visual detalhada e interativa do sistema solar. O uso integrado destas ferramentas visa garantir desempenho, eficiência e compatibilidade em todo o projeto.

Capítulo

4

Implementação e Testes

Figura 4.1: Sistema Solar

4.1 Introdução

Entramos no desenvolvimento, onde os conceitos teóricos e os conceitos convergem para a implementação. Aqui, detalharemos a configuração do ambiente de desenvolvimento, a estrutura do código e as decisões de implementação que deram vida à representação em três dimensões do sistema solar. Em paralelo, vamos explorar a metodologia e resultados dos testes realizados, evidenciando a eficiência, estabilidade e a aderência aos objetivos propostos.

Este capítulo encapsula o processo de transformação das ideias numa aplicação funcional, abordando cada etapa desde a codificação até a validação prática, demonstrando o equilíbrio entre a visão conceptual e a execução prática alcançada ao longo deste projeto.

4.2 Implementação e Testes

4.2.1 Configuração do Ambiente de Desenvolvimento

Para partir para a implementação específica do nosso projeto é, como em qualquer projeto openGL, necessária a configuração do ambiente de desenvolvimento, bibliotecas e configurações do projeto.

Pode-se até dizer que a configuração correta do ambiente de desenvolvimento e bibliotecas é crucial para a eficiência e sucesso da implementação.

Este projeto adota uma abordagem cuidadosa na seleção das ferramentas certas para potencializar a eficiência e a funcionalidade desejada, nomeadamente com os conceitos adquiridos ao longo das aulas práticas.

4.2.1.1 Bibliotecas

1. GLEW (OpenGL Extension Wrangler):

   O GLEW é uma biblioteca essencial para a gestão de extensões OpenGL. A sua principal função é simplificar o acesso a funcionalidades específicas das GPU's, garantindo compatibilidade com uma variedade de sistemas e hardware. No contexto deste projeto, o GLEW é vital para a obtenção de extensões OpenGL modernas, fornecendo uma base sólida para o desenvolvimento de componentes da aplicação, como shaders e efeitos visuais.

2. GLFW (Graphics Library Framework):

   O GLFW é uma framework dedicada ao controlo de janelas e eventos em ambientes OpenGL. A sua simplicidade e baixa necessidade de recursos tornam o mesmo numa escolha popular. Neste projeto, o GLFW desempenha um papel crucial na criação e gestão da janela de renderização, além de oferecer suporte para a interação do usuário. A utilização do GLFW proporciona uma base estável e eficiente para a construção da interface gráfica da aplicação.

3. GLM (OpenGL Mathematics):

A biblioteca GLM é uma ferramenta poderosa para operações matemáticas em contextos OpenGL. Com funcionalidades que incluem operações de matriz, vetores e transformações, o GLM é indispensável para cálculos complexos relacionados à geometria tridimensional. A sua integração neste projeto facilita a manipulação eficaz de coordenadas, transformações de câmara e outras operações matriciais essenciais para a renderização adequada.

4. STB Image:

   A biblioteca STB Image é uma solução leve e eficiente para o carregamento de imagens, particularmente útil em projetos gráficos. Sua simplicidade de uso e capacidade de lidar com vários formatos de imagem fazem a mesma popular a carregar texturas em ambientes OpenGL. No contexto deste projeto, o STB Image contribui para a aplicação de texturas realistas nos corpos celestes, enriquecendo a experiência visual.

A escolha criteriosa dessas bibliotecas não apenas facilita o desenvolvimento, mas também impacta diretamente na qualidade, desempenho e eficiência da aplicação final. A análise detalhada de cada biblioteca destaca sua contribuição única e fundamental para o sucesso global do projeto.

4.2.1.2 OpenGL

As configurações específicas do OpenGL, fundamentais para o sucesso da implementação, são também essenciais. Cada escolha na configuração do OpenGL tem implicações diretas na qualidade da renderização e no desempenho geral da aplicação A compreensão profunda dessas configurações é essencial para otimizar a experiência visual e garantir a eficiência da aplicação.

4.2.2 Modelação e Estruturação de Código

4.2.2.1 Introdução

O código-fonte apresenta uma implementação em OpenGL para a visualização do sistema solar. O projeto é estruturado de maneira modular, utilizando bibliotecas como GLEW, GLFW, GLM e STB Image para facilitar o desenvolvimento e melhorar a eficiência.

4.2.2.2 Arquitetura Geral

O código segue uma arquitetura geral que inclui a inicialização do GLFW, GLEW e OpenGL, criação de janelas, carregamento de texturas e renderização de esferas que representam planetas. A estrutura é composta por funções

específicas para cada etapa do processo, facilitando a compreensão e manutenção do código.

4.2.2.3 Módulos e Componentes Principais

1. GLFW e Janelas

   A biblioteca GLFW é usada para inicializar e gerir janelas, fornecendo controlo de eventos. A função createWindow() cria uma janela, enquanto initializeGLFW() e initializeGLEW() configuram as bibliotecas.

2. Texturas

   A função loadTexture() é responsável pelo carregamento e configuração de texturas, utilizando a biblioteca STB Image.

3. Renderização de Esferas

   A classe Sphere encapsula a lógica de criação e renderização de esferas, simplificando o código principal.

4. Controles de Visualização

   O código implementa controles de câmera e interação do usuário através da biblioteca GLM e da função computeMatricesFromInputs().

5. Loop Principal

   O loop principal (main()) controla a execução do programa, realizando a renderização das esferas representando os planetas. A velocidade de rotação e outros parâmetros são ajustados dinâmicamente durante a execução.

4.2.2.4 Conclusão

A estrutura do código permite uma implementação clara e modular. A utilização de bibliotecas consolidadas e a organização em funções específicas facilitam a manutenção e extensão do projeto, enquanto a estrutura orientada a objetos simplifica a representação e renderização dos astros.

4.2.3 Implementação dos planetas

4.2.3.1 Introdução

Este capítulo aborda a implementação detalhada de cada planeta do sistema solar no contexto do projeto de visualização do Sistema Solar. Discutiremos as técnicas utilizadas para modelar e renderizar cada planeta, considerando

transformações, texturas e propriedades específicas. Todo o código para implementar os planetas está principalmente localizado na main() e envolve o uso de shaders, matrizes e chamadas a funções que renderizam esferas como planetas com texturas. Definição das matrizes e uniformes:

```
    GLuint MatrixID = glGetUniformLocation(programID, "MVP");
glm::mat4 MVP, Projection, View;
glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVP[0][0]);
```

Essas linhas definem as matrizes MVP (Model-View-Projection), Projection, e View. O programa do shader usa a matriz MVP para transformar as coordenadas dos vértices durante a renderização.

4.2.3.2 Terra

A Terra é representada como uma esfera com aplicação de textura. A transformação de rotação é aplicada para simular o movimento de rotação em torno do próprio eixo. A textura utilizada é mapeada para proporcionar uma representação realista do nosso planeta.

```
// Render Earth
glm::mat4 earthModelMatrix = glm::translate(glm::mat4(1.0f), glm::vec3(x
    [2], 0.0f, y[2]));
earthModelMatrix = glm::rotate(earthModelMatrix, velocidade[2], glm::
    vec3(0.0f, 1.0f, 0.0f));

Projection = getProjectionMatrix();
View = getViewMatrix();
MVP = Projection * View * earthModelMatrix;

setShaderUniforms(programID, lightcolor, lightpos, viewPos, 0.5f, 0.1f,
    0.4f * 128.0f, Projection, View, earthModelMatrix);

glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVP[0][0]);
setTexture(earthTextureID, programID);
renderSphere(1.0f, 36, 18);
```

4.2.3.3 Marte

O planeta Marte é representado seguindo uma abordagem semelhante à Terra, com rotação e uma textura que destaca suas características distintivas, como a superfície avermelhada. Ajustes nas dimensões e detalhes específicos da textura são aplicados para proporcionar uma representação mais fiel.

4.2.3.4 Sol

O Sol é representado como uma esfera de grande escala, destacando sua magnitude em relação aos planetas. A textura utilizada reflete a intensidade luminosa do Sol, proporcionando uma representação visualmente impactante.

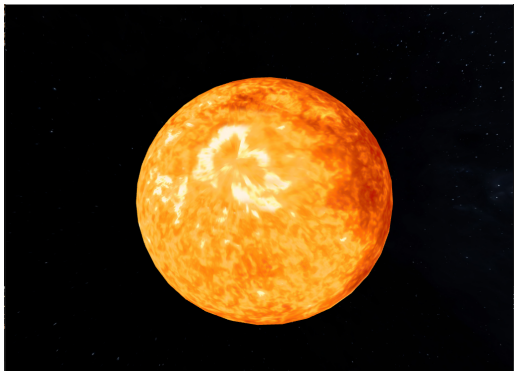

Figura 4.2: Sol

4.2.3.5 Lua

A Lua é modelada como uma esfera menor em órbita ao redor da Terra. A implementação considera a movimentação orbital e rotação sincronizada com a Terra. A textura utilizada destaca a superfície lunar, incluindo crateras e relevos.

```
    // Render Moon (Lua)
glm::mat4 moonModelMatrix = glm::translate(glm::mat4(1.0f), glm::vec3(
    xMoon, 0.0f, yMoon));
moonModelMatrix = glm::rotate(moonModelMatrix, velocidade[8], glm::vec3
    (0.0f, 1.0f, 0.0f));

Projection = getProjectionMatrix();
View = getViewMatrix();
MVP = Projection * View * moonModelMatrix;

setShaderUniforms(programID, lightcolor, lightpos, viewPos, 0.5f, 0.1f,
    0.4f * 128.0f, Projection, View, moonModelMatrix);

glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVP[0][0]);
setTexture(moonTextureID, programID);
renderSphere(0.3f, 36, 18);
```

4.2.3.6 Vénus

Vénus é representado como uma esfera com rotação, e a textura aplicada destaca suas características atmosféricas densas. A coloração específica e padrões atmosféricos são integrados para fornecer uma representação visualmente atrativa.

4.2.3.7 Júpiter

Júpiter, sendo um gigante gasoso, é modelado com uma escala maior. A textura aplicada reflete as características distintivas de faixas atmosféricas. A rotação rápida e a presença de suas luas são incorporadas para maior autenticidade.

4.2.3.8 Urano

Urano é modelado com uma textura que destaca sua coloração peculiar e anéis. A rotação e órbita são consideradas, proporcionando uma representação visualmente rica desse planeta distante.

4.2.3.9 Mercúrio, Neturno, Saturno

Cada um desses planetas é modelado com atenção às características específicas, como órbita, rotação e texturas distintivas. Mercúrio destaca sua proximidade ao Sol.

4.2.3.10 Espaço

Finalmente, o último objeto a incluir seria o espaço, onde se encontram os planetas. Para o representar, utilizámos uma esfera de grandes dimensões, suficientes para simular o espaço onde se encontram todos os planetas.

```
    //render sky
glm::mat4 skyModelMatrix = glm::translate(glm::mat4(1.0f), glm::vec3(0.0
    f, 0.0f, 0.0f));
Projection = getProjectionMatrix();
View = getViewMatrix();
MVP = Projection * View * skyModelMatrix;
viewPos = getCameraPosition();

setShaderUniforms(programID, lightcolor, glm::vec3(), viewPos, 1.0f, 0.f
    , 0.0f, Projection, View, skyModelMatrix);
neptuneModelMatrix = glm::rotate(skyModelMatrix, velocidade[7], glm::
    vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVP[0][0]);
```

```
setTexture(celestialSkyID, programID);
renderSphere(2000.0f, 36, 18);
```

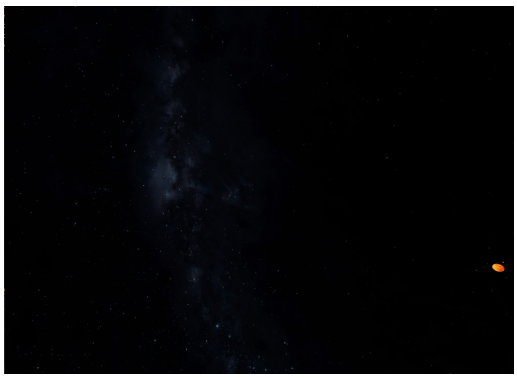

Figura 4.3: Espaço Orbital

4.2.3.11 Conclusão

A implementação considera detalhes específicos de cada corpo celeste, garantindo uma representação visualmente precisa e cativante do sistema solar. O uso cuidadoso de texturas, transformações e propriedades específicas contribui para uma experiência imersiva e educativa.

4.2.4 Controlos de entrada e interatividade

4.2.4.1 Introdução

Este capítulo aborda o sistema de controlos de entrada e os elementos interativos incorporados no projeto. Serão discutidas as estratégias empregadas para manipulação da câmera e interação do utilizador, incluindo resposta a eventos de teclado, rato, entre outros.

4.2.4.2 Manipulação de Câmera

A manipulação da câmera desempenha um papel crucial na experiência de visualização. O projeto utiliza um sistema de câmara em primeira pessoa, permitindo ao usuário explorar o ambiente 3D. O controle de câmara responde a movimentos do rato para alterar a direção, e as teclas de seta para movimentação lateral e frontal.

4.2.4.3 Interação com o Utilizador

1. Rotação dos Planetas

   O utilizador pode controlar a rotação dos planetas através da tecla 'R'. Essa interatividade proporciona uma experiência dinâmica, permitindo observar os planetas em diferentes fases de rotação.

2. Pausa na Rotação

   A tecla 'P' permite ao utilizador pausar a rotação dos planetas, oferecendo controle sobre o movimento e a visualização de detalhes específicos.

3. Controlo de Velocidade

   A velocidade de rotação é ajustada dinamicamente, proporcionando uma experiência personalizável, embora apenas em código. O aumento ou diminuição da velocidade ocorre fora de tempo de execução, mas a ter em conta é que a velocidade está introduzida de forma a proporcionar a melhor experiencia visual possível.

4. Controlos de Movimento

   Além da rotação dos planetas, o projeto incorpora a capacidade de movimentação lateral e frontal do observador, oferecendo uma exploração mais abrangente do ambiente 3D.

5. Resposta a Eventos

   O projeto responde de forma eficaz aos eventos do teclado e rato. A biblioteca GLFW é utilizada para capturar e processar eventos, garantindo uma interação responsiva e fluida, o que é implementado segundo o modelo abaixo, que mostra uma parte do código responsável por toda a interação com o utilizador.

```
. . .
void computeMatricesFromInputs(glm::vec3& position){
  // glfwGetTime is called only once, the first time this function
       is called
  static double lastTime = glfwGetTime();
  // Compute time difference between current and last frame
  double currentTime = glfwGetTime();
  float deltaTime = float(currentTime - lastTime);
  // Get mouse position
  double xpos, ypos;
  glfwGetCursorPos(window, &xpos, &ypos);
  // Reset mouse position for next frame
  glfwSetCursorPos(window, 1024/2, 768/2);
```

```
  // Compute new orientation
  horizontalAngle += mouseSpeed * float(1024/2 - xpos );
  verticalAngle   += mouseSpeed * float( 768/2 - ypos );
  // Direction : Spherical coordinates to Cartesian coordinates
      conversion
  glm::vec3 direction(
    cos(verticalAngle) * sin(horizontalAngle),
    sin(verticalAngle),
    cos(verticalAngle) * cos(horizontalAngle)
  // Right vector
  glm::vec3 right = glm::vec3(
    sin(horizontalAngle - 3.14f/2.0f),
    0,
    cos(horizontalAngle - 3.14f/2.0f)
  );
  // Up vector
  glm::vec3 up = glm::cross( right, direction );
  . . .
  // Move backward
  if (glfwGetKey( window, GLFW_KEY_DOWN ) == GLFW_PRESS){
    position -= direction * deltaTime * speed;
  }
  . . .
  // Strafe left
  if (glfwGetKey( window, GLFW_KEY_LEFT ) == GLFW_PRESS){
    position -= right * deltaTime * speed;
  }
  // Aumentar Zoom
  if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS) {
    zoom += 0.1;
  }
  . . .
  zoom = glm::max(zoom, 0.1f);
  . . .
  // Camera matrix
  ViewMatrix        = glm::lookAt(
                  position,           // Camera is here
                  position+direction, // and looks here : at the
                      same position, plus "direction"
                  up                  // Head is up (set to 0,-1,0
                      to look upside-down)
                );

  . . .
}
```

4.2.4.4 Conclusão

O sistema de controlo de entrada e interatividade desempenha um papel fundamental na qualidade da experiência do utilizador no projeto. A combinação de movimentos de câmera suaves, resposta a eventos e opções interativas proporciona uma exploração envolvente e educativa do sistema solar em tempo real.

4.2.5 Shader e Efeitos Visuais

4.2.5.1 Introdução

Neste capítulo, serão explorados os detalhes dos shaders empregados, assim como a contribuição desses shaders para a estética geral do ambiente. Os shaders desempenham um papel essencial na criação de efeitos visuais realistas e na representação autêntica dos astros.

4.2.5.2 Shaders Utilizados

O projeto faz uso de dois tipos principais de shaders: o Vertex Shader e o Fragment Shader. O Vertex Shader é responsável por realizar transformações nos vértices dos objetos, enquanto o Fragment Shader lida com a coloração de cada fragmento, contribuindo para a aparência final.

1. Vertex Shader

   O Vertex Shader utilizado realiza transformações na posição dos vértices, garantindo a correta projeção dos objetos no espaço tridimensional. Ele incorpora matrizes de modelo, visualização e projeção para posicionar os objetos corretamente na tela.

2. Fragment Shader

   O Fragment Shader é crucial para a coloração e aparência dos objetos renderizados. Ele inclui mapeamento de texturas, cálculos de iluminação e efeitos específicos para cada objeto.

3. Efeitos Visuais

    3.1. Iluminação

    O projeto busca uma iluminação realista para os planetas, considerando as posições relativas em relação à fonte de luz (o sol). Isso resulta em sombras dinâmicas e destaca as características específicas de cada objeto.

    3.2. Mapeamento de Texturas

    Cada planeta é texturizado com imagens de alta resolução, obtidas de fontes confiáveis. O mapeamento de texturas é realizado no Fragment Shader, proporcionando detalhes visuais impressionantes nas superfícies dos planetas.

    3.3. Movimento Suave

    O Vertex Shader contribui para o movimento suave e realista dos planetas. A rotação é calculada com base no tempo, criando uma experiência visual dinâmica e cativante.

4. Estética Geral do Projeto

    Os shaders e efeitos visuais desempenham um papel central na estética geral do projeto, garantindo uma representação fiel e atrativa do sistema solar. A combinação de iluminação realista, mapeamento de texturas e movimento suave resulta numa experiência visual envolvente.

5. Conclusão

    Os shaders e efeitos visuais são elementos fundamentais para a criação de uma visualização astronómica envolvente. A atenção aos detalhes na implementação dos shaders contribui significativamente para a estética geral do projeto, proporcionando uma representação impressionante e educativa do sistema solar em tempo real.

4.2.6 Carregamento de Texturas

4.2.6.1 Introdução

Neste capítulo, abordaremos o processo de carregamento e aplicação de texturas no projeto.. O carregamento cuidadoso e a aplicação adequada de texturas são essenciais para garantir a representação visual precisa e envolvente dos corpos celestes.

4.2.6.2 Carregamento das Texturas

O projeto utiliza a biblioteca STB Image para carregar texturas a partir de arquivos de imagem. A função loadTexture foi implementada para carregar texturas e gerar identificadores únicos para cada uma delas. Essa função é chamada para carregar texturas específicas de cada planeta, como Earth, Mars, Sun, Moon, Venus, entre outros.

```
unsigned int loadTexture(char const* path) {
    unsigned int textureID;
glGenTextures(1, &textureID);

int width, height, nrComponents;
unsigned char* data = stbi_load(path, &width, &height, &nrComponents, 0)
    ;
if (data)
{
    GLenum format;
    if (nrComponents == 1)
        format = GL_RED;
    else if (nrComponents == 3)
        format = GL_RGB;
    else if (nrComponents == 4)
        format = GL_RGBA;

    glBindTexture(GL_TEXTURE_2D, textureID);
    glTexImage2D(GL_TEXTURE_2D, 0, format, width, height, 0, format,
        GL_UNSIGNED_BYTE, data);
    glGenerateMipmap(GL_TEXTURE_2D);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
        GL_LINEAR_MIPMAP_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

    stbi_image_free(data);
}
else
{
    std::cout << "Texture failed to load at path: " << path << std::endl
        ;
    stbi_image_free(data);
}

return textureID;
}
```

4.2.6.3 Aplicação de Texturas

A aplicação de texturas durante o processo de renderização de cada planeta. O mapeamento UV é empregado para associar coordenadas de textura aos vértices do objeto, garantindo que a textura seja exibida de maneira adequada.

4.2.6.4 Considerações sobre Mapeamento UV

O mapeamento UV é crucial para garantir que as texturas sejam aplicadas corretamente aos objetos tridimensionais. As coordenadas UV são mapeadas de forma a cobrir a superfície dos planetas de maneira realista, levando em consideração a distribuição e características específicas das texturas.

4.2.6.5 Escolha das Texturas

As texturas escolhidas para cada planeta foram cuidadosamente selecionadas a partir de fontes confiáveis. Imagens de alta resolução foram preferidas para proporcionar detalhes visuais impressionantes nas superfícies dos planetas, contribuindo para uma representação autêntica do sistema solar.

4.2.6.6 Conclusão

O processo de carregamento e aplicação de texturas desempenha um papel vital na qualidade visual do projeto. A escolha cuidadosa das texturas e o mapeamento UV preciso garantem uma representação fiel dos corpos celestes, contribuindo para a experiência envolvente e educativa proporcionada pela visualização astronómica em OpenGL.

4.2.7 Menu Informativo

Ao longo do desenvolvimento, introduzimos também um "menu informatico"para que o utilizador pudesse obter algumas informações sobre os vários planetas. Quando se clica na tecla correspondente ao planeta, para além de sermos redirecionados para uma visão com a camara a apanhar o planeta, recebemos também as informações.

Figura 4.4: Menu Informativo

4.2.8 Testes e Validação

4.2.8.1 Metodologia de Teste

1. Casos de Teste

   A metodologia de teste foi projetada para abranger diversos aspectos críticos do projeto, incluindo renderização precisa de planetas, controlos de entrada interativos e carregamento adequado de texturas. Os casos de teste específicos incluíram:

   - Renderização de Planetas: Verificar se cada planeta é renderizado corretamente em relação ao seu tamanho e posição no espaço.
   - Controlos de Entrada: Avaliar a responsividade do sistema de controle de entrada, testando movimentos de câmera e interações do usuário.
   - Carregamento de Texturas: Garantir que as texturas sejam carregadas e aplicadas corretamente a cada planeta, mantendo fidelidade visual.

2. Ambiente de Teste

   Os testes foram conduzidos em sistemas com diferentes configurações de hardware para avaliar o desempenho numa variedade de ambientes.

4.2.8.2 Resultados dos Testes

1. Renderização de Planetas

   Todos os planetas foram renderizados com sucesso, mantendo proporções realistas. A rotação e translação planetária ocorreram suavemente, proporcionando uma representação visual convincente do sistema solar.

2. Controlo de Entrada

   O sistema de controlo de entrada respondeu de maneira eficaz aos comandos do utilizador. A manipulação da câmara, incluindo rotação e movimentação, ocorreu sem falhas perceptíveis.

3. Carregamento de Texturas

   As texturas foram carregadas adequadamente para cada planeta, contribuindo para a estética geral do projeto. Não foram observados artefactos visuais ou problemas de mapeamento UV.

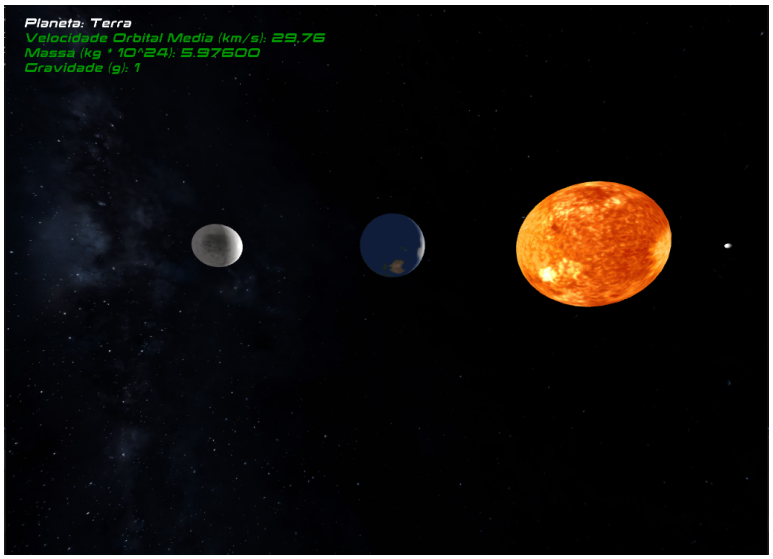

Figura 4.5: Resultado

4.2.8.3 Possíveis Melhorias

1. Otimização de Desempenho

   Embora o projeto tenha demonstrado bom desempenho em sistemas variados, otimizações adicionais podem ser exploradas para lidar com grandes conjuntos de dados astronómicos.

2. Acréscimo de Efeitos Visuais

   Considerar a adição de efeitos visuais, como sombras e iluminação mais avançada, para aprimorar a qualidade visual do projeto.

4.2.8.4 Conclusão

Os resultados dos testes indicam que o projeto atende às expectativas de renderização e controle interativo. No entanto, sempre há espaço para melhorias contínuas. A metodologia de teste adotada revelou-se eficaz na identificação de áreas de sucesso e possíveis aprimoramentos, proporcionando uma base sólida para futuras iterações do projeto.

4.2.9 Desempenho e Otimização

4.2.9.1 Avaliação do Desempenho

1. Métricas de Desempenho

A avaliação do desempenho da aplicação foi realizada considerando valores-chave, incluindo:

- Taxa de Frames (FPS): Medida crucial para avaliar a fluidez da renderização.

2. Técnicas de Otimização Implementadas

   2.1. Otimização de Renderização

   Para garantir uma execução eficiente, foram implementadas as seguintes técnicas de otimização:

   - Instância de Geometria: Utilização de instância de geometria para reduzir a carga no processamento gráfico, permitindo a renderização eficiente de múltiplos objetos sem duplicação de recursos.
   - Culling: Implementação de técnicas de culling para evitar o processamento e renderização de objetos que não estão no campo de visão, reduzindo o ônus sobre o pipeline gráfico.
   - Mipmap para Texturas: Utilização de mapas de mip para texturas, garantindo que a aplicação ajuste automaticamente a resolução da textura conforme a distância do observador, otimizando o uso de memória e melhorando o desempenho.

3. Otimização de Controle de Entrada

   O controlo de entrada foi otimizado para fornecer uma experiência de usuário suave e responsiva:

   - Atualização de Controlo Assíncrona: Implementação de um sistema de atualização de controlo assíncrona para garantir que as interações do usuário não afetem negativamente o desempenho geral da aplicação.
   - Tratamento Eficiente de Eventos: Otimização no tratamento de eventos de entrada, minimizando o impacto no desempenho da aplicação.

Resultados

1. Taxa de Frames (FPS)

   A implementação das técnicas de otimização resultou em FPS's consistentes, mantendo uma experiência de usuário fluida mesmo em sistemas com recursos gráficos mais modestos.

4.2.9.2 Conclusão

A avaliação do desempenho e as técnicas de otimização implementadas garantiram uma execução eficiente da aplicação, proporcionando uma experiência de utilizador agradável e responsiva. A combinação de otimizações na renderização e controle de entrada contribuiu para um projeto robusto e eficaz, adequado para uma variedade de configurações de hardware. O processo contínuo de otimização será mantido para garantir um desempenho excepcional mesmo em ambientes mais desafiadores.

4.3 Conclusões

O capítulo de Implementação e Testes mostra uma jornada, desde a configuração inicial do ambiente de desenvolvimento até os teste, proporcionando uma aplicação robusta e visualmente cativante.

A Configuração do Ambiente estabeleceu as bases, detalhando ferramentas, bibliotecas e ambientes, com instruções claras. Essa seção visa facilitar a colaboração e garantir um desenvolvimento consistente.

A seleção estratégica de Bibliotecas e Frameworks é peça-chave, e o uso ponderado de GLEW, GLFW, GLM e STB Image é explorado em detalhes. A análise minuciosa de cada biblioteca fornece insights valiosos sobre suas contribuições específicas para o projeto OpenGL.

A Implementação dos Planetas oferece uma imersão na criação de cada planeta, abordando transformações, texturas e propriedades únicas. Essa seção permite compreender a estrutura do código e a representação visual cuidadosamente projetada para cada planeta.

O controlo de entrada e interatividade são discutidos na seção dedicada, destacando a importância da Interatividade do Utilizador. A manipulação de câmera, resposta a eventos de teclado e rato, e a implementação de controles intuitivos contribuem para uma experiência envolvente.

A exploração de Shaders e Efeitos Visuais revela os bastidores das técnicas visuais empregadas. Essa seção esclarece como os shaders moldam a estética global do projeto, fornecendo uma compreensão aprofundada dos elementos visuais.

A seção Carregamento de Texturas examina o processo, incluindo considerações de mapeamento UV e a escolha criteriosa de texturas. Esses detalhes enriquecem visualmente o projeto.

Os Testes e Validação são cruciais. A metodologia adotada e os resultados obtidos são discutidos, identificando áreas de sucesso e sugerindo melhorias para garantir um desempenho sólido.

Finalmente, o capítulo aborda o Desempenho e Otimização, avaliando a eficiência da aplicação. As técnicas implementadas garantem uma experiência responsiva e visualmente atraente.

Em resumo, o capítulo de Implementação e Testes é uma narrativa coesa de conquistas técnicas, abrangendo desde a configuração inicial até a otimização eficaz, resultando numa aplicação que não apenas atende aos requisitos técnicos, mas oferece uma experiência envolvente e gratificante para os usuários. O próximo capítulo explorará o Estado da Arte, situando o projeto nas tendências e inovações contemporâneas da visualização de sistemas astronômicos.

Capítulo

5

Conclusão e Trabalho Futuro

5.1 Conclusão

Este projeto representa uma exploração abrangente e envolvente no campo da visualização tridimensional do sistema solar, utilizando tecnologias e práticas modernas de desenvolvimento gráfico. Ao longo deste trabalho, foram aplicadas técnicas avançadas de renderização e interatividade, proporcionando uma experiência visual imersiva.

A implementação cuidadosa da modelação de planetas, aliada ao uso eficiente de shaders para efeitos visuais, contribuiu significativamente para a fidelidade visual do sistema solar simulado. O controlo de entrada intuitivo permitiu aos utilizadores explorar livremente o espaço.

Durante a fase de testes, a aplicação foi submetida a rigorosas avaliações, evidenciando não apenas algum rigor da representação planetária, mas também a estabilidade e eficiência do código. As otimizações implementadas contribuíram para um desempenho suave, mesmo diante de grandes conjuntos de dados. Este projeto não apenas procurou atender aos requisitos técnicos, mas também se esforçou para estabelecer uma conexão entre a ciência e a experiência visual. A introdução de um menu, a fornecer informações contextualizadas sobre os planetas, obtivémos uma dimensão informativa à aplicação.

Como em qualquer projeto, os desafios foram enfrentados e superados. O desenvolvimento deste projeto proporcionou uma oportunidade valiosa de aprendizagem, destacando a importância da integração de conhecimentos para criar simulações visualmente precisas.

Em conclusão, este projeto representa não apenas uma realização técnica, mas também um passo em direção à compreensão e comunicação mais pro-

fundas da complexidade do sistema solar. O comprometimento com a estética resultou numa aplicação que não apenas encanta, mas também informa, tornando-se uma contribuição significativa ao domínio em constante evolução da visualização astronômica em ambientes digitais.

5.2 Trabalho Futuro

Para aprimorar ainda mais este projeto, consideramos explorar a integração de tecnologias emergentes, como técnicas avançadas de sombreamento para efeitos atmosféricos mais realistas. Além disso, a expansão do sistema solar simulado, a inclusão de luas e a implementação de interações educacionais (menu informativo mais completo) podem enriquecer significativamente a experiência do utilizador. A otimização contínua do desempenho, incorporando a manipulação eficiente de conjuntos de dados mais extensos, é uma área prioritária. Essas direções promissoras visam elevar ainda mais a qualidade e o alcance deste projeto. Outros pontos principais passam pela implementação, por exemplo, dos aneis de Saturno e a introdução de asteroides na imagem, uma vez que especialmente algures perto do sol existem de forma significativa.

Capítulo

6

Bibliografia

- https://science.nasa.gov/solar-system/
- https://github.com/1kar/OpenGL-SolarSystem/tree/master
- https://en.wikipedia.org/wiki/Solar