UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CIÊNCIA DA COMPUTAÇÃO

DANIEL MAHL GREGORINI

PROJETO DE COMPUTAÇÃO GRÁFICA: MODELO DE BICICLETA REALISTA COM OPENGL

MEDIANEIRA
2024

DANIEL MAHL GREGORINI

PROJETO DE COMPUTAÇÃO GRÁFICA: MODELO DE BICICLETA REALISTA COM OPENGL

Projeto de computação gráfica: modelo de bicicleta realista com OpenGL, apresentado à disciplina de Computação Grafica, do curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná – UTFPR.
Prof. Dr. Pedro Luiz De Paula Filho

MEDIANEIRA
2024

RESUMO

GREGORINI, Daniel. O desenvolvimento de modelos tridimensionais realistas usando OpenGL é uma prática avançada na computação gráfica que permite a criação de simulações visuais detalhadas e interativas. Este artigo apresenta um projeto realizado como parte da disciplina de Computação Gráfica no curso de Bacharelado em Ciência da Computação da UTFPR. O foco do projeto foi o desenvolvimento de um modelo de bicicleta em 3D que oferece não apenas uma representação visual realista, mas também interações dinâmicas com componentes móveis, como pedais e rodas. Foram utilizadas técnicas avançadas de modelagem, texturização e iluminação. O artigo explora as metodologias aplicadas na construção do modelo, as ferramentas de software utilizadas e discute os desafios encontrados no processo de simulação física e visual. O projeto demonstrou eficácia na aplicação de OpenGL para renderização 3D e destacou a importância do ensino prático de técnicas de computação gráfica avançadas em um contexto acadêmico.

Palavras-chave: Computação gráfica, OpenGL, Modelagem 3D, Simulação interativa, Realismo visual.

ABSTRACT

GREGORINI, Daniel. The development of realistic three-dimensional models using OpenGL is an advanced practice in computer graphics that allows for the creation of detailed and interactive visual simulations. This article presents a project carried out as part of the Computer Graphics course in the Bachelor of Computer Science program at UTFPR. The focus of the project was the development of a 3D bicycle model that offers not only a realistic visual representation but also dynamic interactions with moving components, such as pedals and wheels. Advanced modeling, texturing, and lighting techniques were used. The article explores the methodologies applied in building the model, the software tools used, and discusses the challenges encountered in the physical and visual simulation process. The project demonstrated effectiveness in the application of OpenGL for 3D rendering and highlighted the importance of practical teaching of advanced computer graphics techniques in an academic context.

Keywords: Computer Graphics, OpenGL, 3D Modeling, Interactive Simulation, Visual Realism.

SUMÁRIO

1 INTRODUÇÃO

No campo em rápida evolução da computação gráfica, a capacidade de criar modelos tridimensionais detalhados e realistas é crucial para uma ampla gama de aplicações, que vão desde jogos eletrônicos e realidade virtual até visualização arquitetônica e ferramentas educacionais. A utilização de bibliotecas gráficas avançadas, como o OpenGL, tem sido fundamental nesse processo, permitindo que desenvolvedores e designers produzam experiências visuais que são não apenas impressionantes em termos estéticos, mas também interativas e funcionalmente ricas

2 DESCRIÇÃO DO PROJETO

O projeto foi desenvolvido com o objetivo de criar um modelo tridimensional de uma bicicleta utilizando a biblioteca OpenGL, parte da disciplina de Computação Gráfica no curso de Bacharelado em Ciência da Computação da UTFPR. Este modelo inclui detalhes interativos e dinâmicos, como rodas, pedais, engrenagem e um guidão que se move para a direita e para a esquerda, replicando a funcionalidade de uma bicicleta real. Além disso, o usuário pode acelerar e frear a bicicleta, e a direção é controlada pelo movimento do guidão, aumentando a imersão e a interatividade do modelo.

Para experiência do usuário, o projeto também inclui controles para manipulação da câmera e da cena, permitindo aos usuários explorar diferentes ângulos e perspectivas e entender melhor a mecânica e a interação dos componentes da bicicleta.

3 ESTUDO DO PROJETO

3.1 Modelagem hierárquica na bicicleta

Toda a modelagem objeto do projeto foi criado com vetores do OpenGL, cores solidas foram usadas nas partes da bicicleta, vamos usar de exemplo do código de modelagem do assento da bicicleta:

figura 1: código da modelagem do assento da bicicleta

```
void desenharAssento()
{
    glBegin(GL_POLYGON);
    glVertex3f(-0.1f, 1.0f, -0.5f);
    glVertex3f(1.0f, 1.0f, -0.3f);
    glVertex3f(1.0f, 1.0f, 0.3f);
    glVertex3f(-0.1f, 1.0f, 0.5f);
    glVertex3f(-0.5f, 1.0f, 1.0f);
    glVertex3f(-1.0f, 1.0f, 1.0f);
    glVertex3f(-1.0f, 1.0f, -1.0f);
    glVertex3f(-0.5f, 1.0f, -1.0f);
    glEnd();
    glBegin(GL_POLYGON);
    glVertex3f(-0.1f, -1.0f, -0.5f);
    glVertex3f(1.0f, -1.0f, -0.3f);
    glVertex3f(1.0f, -1.0f, 0.3f);
    glVertex3f(-0.1f, -1.0f, 0.5f);
    glVertex3f(-0.5f, -1.0f, 1.0f);
    glVertex3f(-1.0f, -1.0f, 1.0f);
    glVertex3f(-1.0f, -1.0f, -1.0f);
    glVertex3f(-0.5f, -1.0f, -1.0f);
    glEnd();
    glBegin(GL_QUADS);
    glVertex3f(1.0f, 1.0f, -0.3f);
    glVertex3f(1.0f, 1.0f, 0.3f);
    glVertex3f(1.0f, -1.0f, 0.3f);
    glVertex3f(1.0f, -1.0f, -0.3f);
    glVertex3f(1.0f, 1.0f, 0.3f);
    glVertex3f(-0.1f, 1.0f, 0.5f);
    glVertex3f(-0.1f, -1.0f, 0.5f);
    glVertex3f(1.0f, -1.0f, 0.3f);
    glVertex3f(1.0f, 1.0f, -0.3f);
    glVertex3f(-0.1f, 1.0f, -0.5f);
    glVertex3f(-0.1f, -1.0f, -0.5f);
    glVertex3f(1.0f, -1.0f, -0.3f);
    glVertex3f(-0.1f, 1.0f, 0.5f);
    glVertex3f(-0.5f, 1.0f, 1.0f);
    glVertex3f(-0.5f, -1.0f, 1.0f);
    glVertex3f(-0.1f, -1.0f, 0.5f);
    glVertex3f(-0.1f, 1.0f, -0.5f);
    glVertex3f(-0.5f, 1.0f, -1.0f);
    glVertex3f(-0.5f, -1.0f, -1.0f);
    glVertex3f(-0.1f, -1.0f, -0.5f);
    glVertex3f(-0.5f, 1.0f, 1.0f);
    glVertex3f(-1.0f, 1.0f, 1.0f);
    glVertex3f(-1.0f, -1.0f, 1.0f);
```

```
    glVertex3f(-0.5f, -1.0f, 1.0f);
    glVertex3f(-0.5f, 1.0f, -1.0f);
    glVertex3f(-1.0f, 1.0f, -1.0f);
    glVertex3f(-1.0f, -1.0f, -1.0f);
    glVertex3f(-0.5f, -1.0f, -1.0f);
    glVertex3f(-1.0f, 1.0f, 1.0f);
    glVertex3f(-1.0f, 1.0f, -1.0f);
    glVertex3f(-1.0f, -1.0f, -1.0f);
    glVertex3f(-1.0f, -1.0f, 1.0f);

    glEnd();
}
```

Neste código vemos a criação de vários polígonos para modelagem do assento. A bicicleta segue um modelo hierárquico, onde o com o desenho do quadro. Em seguida, a partir do quadro, a inserção dos o poligonos do banco e guidão, modelagem da corrente e por último dos pedais

3.2 Textura do chão

Para inserção da textura do chão foi usada a biblioteca SOIL (*Simple OpenGL Image Library*), uma imagem 10000 por 10000 pixels foi usada como grama para o chão.

figura 2: textura de grama

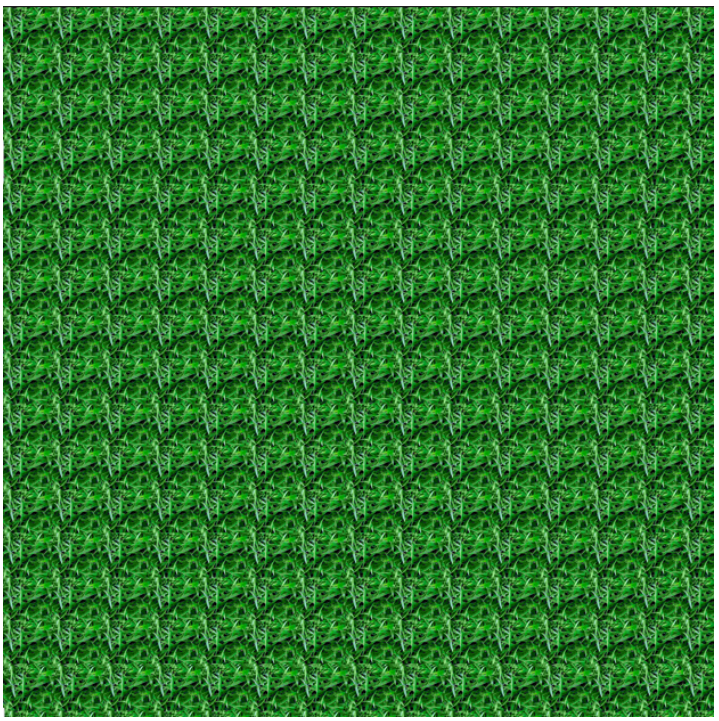

figura 3: código do carregamento da textura

```
int carregarTexturas() {
            texture = SOIL_load_OGL_texture ( "C:\\grama.jpg", SOIL_LOAD_RGBA,
SOIL_CREATE_NEW_ID,SOIL_FLAG_INVERT_Y);
             if (texture == 0) {
                fprintf(stderr, "Erro ao carregar textura: %s\n", SOIL_last_result());
                return false; // Pode retornar algum código de erro se necessário
             }
             glBindTexture(GL_TEXTURE_2D, texture);
             glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
             glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
             glEnable(GL_TEXTURE_2D);          // Habilitar mapeamento de textura
             glShadeModel(GL_SMOOTH);  // Habilitar sombreamento suave
             glClearColor(0.68f, 0.85f, 0.90f, 1.0f); // Cor de fundo azul claro
             glClearDepth(1.0f);               // Configurar buffer de profundidade
             glEnable(GL_DEPTH_TEST);          // Habilitar teste de profundidade
             glDepthFunc(GL_LEQUAL);           // Configurar o tipo de teste de profundidade
             glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST); // Melhorar cálculos de
perspectiva

            return true;
}
```

O código carrega a textura em formato de imagem do disco C, então guarda em uma variável para não ser colocada no polígono do chão na função de “desenhaSolo()”.

3.2 Iluminação do projeto

A iluminação é implementada através de uma combinação de luzes direcionais e difusas, configuradas para destacar adequadamente as formas e texturas do modelo. A luz direcional é empregada para simular a iluminação solar, caracterizada por raios paralelos que iluminam todos os objetos na cena de maneira uniforme. Isso é essencial para garantir que a bicicleta apareça tridimensional, mesmo em uma visualização 2D. Os parâmetros de luz direcional, incluindo posição e intensidade, são meticulosamente ajustados para realçar os detalhes do modelo e criar sombras que conferem profundidade e realismo.

Paralelamente, a luz difusa é utilizada para suavizar as sombras geradas pela luz direcional, prevenindo contrastes excessivamente marcantes que poderiam comprometer a percepção dos detalhes do modelo. A configuração dessa luz é crucial para assegurar que as superfícies da bicicleta sejam visíveis sob diferentes ângulos de visualização, melhorando a experiência interativa do usuário.

Além das configurações de iluminação, o projeto também envolve a definição de materiais que interagem com as luzes. Propriedades como a refletância especular são ajustadas para que a bicicleta responda de forma realista às

condições de iluminação, refletindo luz em ângulos específicos e contribuindo para a autenticidade visual do modelo.

figura 4: código da iluminação

```
void init()
{
    // Definições dos materiais para refletância e brilho
    GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0}; // Refletância especular do material
    GLfloat mat_shininess[] = {101.0}; // Intensidade do brilho do material
    // Definições da luz
    GLfloat light_directional[] = {1.0, 1.0, 1.0, 1.0}; // Direção e intensidade da luz direcional
     GLfloat light_positional[] = {1.0, 1.0, 1.0, 0.0}; // Posição da luz no espaço (luz pontual se o último
valor é 0)
    GLfloat light_diffuse[] = {1.0, 1.0, 1.0}; // Intensidade da luz difusa
    // Reset inicial dos parâmetros modificados pelo usuário ou por interações anteriores
    reset();
    // Define o modelo de sombreamento para suavização das cores
    glShadeModel(GL_SMOOTH);
    // Configura a luz direcional
    glLightfv(GL_LIGHT0, GL_POSITION, light_directional); // Define a posição/direção da luz
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_diffuse); // Define a intensidade da luz ambiente
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse); // Define a intensidade da luz difusa
    // Configura os materiais
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess); // Aplica a intensidade do brilho
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular); // Aplica a refletância especular
     glColorMaterial(GL_FRONT, GL_DIFFUSE); // Permite que a cor do material seja definida pela cor
corrente
    // Ativa os recursos de iluminação e profundidade
    glEnable(GL_LIGHTING); // Ativa a iluminação geral
    glEnable(GL_LIGHT0); // Ativa a primeira luz configurada
        glEnable(GL_COLOR_MATERIAL); // Permite a modificação das propriedades de cor dos
materiais
    glEnable(GL_DEPTH_TEST); // Ativa o teste de profundidade para renderização 3D correta
}
```

3.3 Animação dos pedais e engrenagens

No código da função “engrenagem()”, diversos conceitos matemáticos são habilmente aplicados para gerar a geometria de uma engrenagem tridimensional. A função começa definindo as variáveis e constantes necessárias para os cálculos, como o número de dentes da engrenagem (teeth), o raio interno (inner_radius), o raio externo (outer_radius), a profundidade dos dentes (tooth_depth), e a constante π (pi), fundamental para cálculos envolvendo círculos.

O ângulo de cada dente é calculado dividindo um círculo completo (2π radianos) pelo número total de dentes. Esse ângulo determina a posição de cada dente ao redor do círculo da engrenagem. Para dar forma aos dentes, o ângulo é subdividido em quatro partes menores (da), permitindo um detalhamento preciso nas faces e bordas de cada dente.

A conversão de coordenadas polares para cartesianas é usada para posicionar cada vértice dos dentes. Usando o seno e o cosseno do ângulo calculado, determina-se as coordenadas x e y para cada ponto, baseando-se nos

raios interno e externo ajustados pela profundidade do dente. Este método permite que a engrenagem seja construída com uma forma circular perfeita, essencial para o funcionamento mecânico e visual da engrenagem.

Para o giro dos penais se usa uma variável global “pedalAngle”, no desenho dos pedais é usado para calcular o ângulo do pedal, tudo frame que atualiza o programa se calcula essa variável e carrega de novo o modelo dos pedais

figura 5: calcula a parte traseira dos dentes da engrenagem

```
for (i = 0; i <= teeth; i++)
   {
      angle = i * 2.0 * pi / teeth;
      glVertex3f(r1 * cos(angle), r1 * sin(angle), -width * 0.5);
      glVertex3f(r0 * cos(angle), r0 * sin(angle), -width * 0.5);
      glVertex3f(r1 * cos(angle + 3 * da), r1 * sin(angle + 3 * da), -width * 0.5);
      glVertex3f(r0 * cos(angle), r0 * sin(angle), -width * 0.5);
   }
   glEnd();
   glBegin(GL_QUADS)
```

3.4 Input e eventos do teclado e mouse

Para o teclado, o código utiliza a função glutKeyboardFunc, que registra uma função de callback chamada teclado. Esta função é chamada sempre que uma tecla é pressionada. Dentro da função teclado, diferentes teclas são verificadas usando uma estrutura de controle switch. Teclas '+' e '-' são usadas para aumentar ou diminuir a velocidade, respectivamente. Teclas 'b' e 'z' ajustam a direção do guidão para direita ou esquerda. A tecla 's' reinicia a cena para o estado inicial. A tecla 'esc' (27 em ASCII) fecha a aplicação.

Os inputs do Mouse utilizam a função do OpenGL glutMouseFunc, para tratar cliques de botões do mouse. Dentro desta função, dependendo do botão pressionado e seu estado (pressionado ou solto), variáveis são ajustadas para refletir essa ação. glutMotionFunc e glutPassiveMotionFunc são utilizadas para capturar o movimento do mouse quando um botão está pressionado ou quando nenhum botão está pressionado, respectivamente. Essas funções ajustam as variáveis de ângulo (anglex, angley, anglez) para rotacionar a cena com base no movimento do mouse.

Esses callbacks permitem uma interação interativa com a aplicação, permitindo ao usuário controlar a visualização e outros aspectos da cena em tempo real. A configuração dessas funções é feita no início do programa na função glSetupFuncs, garantindo que todas as funcionalidades de input estejam prontas assim que a aplicação iniciar

figura 6: função para lidar com o teclado

```
void teclado(unsigned char key, int x, int y)
{
   GLfloat r = 0.0f;
   switch (key)
   {
   case 's':
   case 'S':
      reset();
      break;
   case 'z':
      if (steering < HANDLE_LIMIT)
         steering += INC_STEERING;
      break;
   case 'b':
      if (steering > -HANDLE_LIMIT)
         steering -= INC_STEERING;
      break;
   case '+':
      speed += INC_SPEED;
      break;
   case '-':
      speed -= INC_SPEED;
      break;
   case 27:
      exit(1);
   }
   pedalAngle += speed;
   //não deixa a velocidade da bike menor que 0
   if (speed < 0.0f)
      speed = 0.0f;
   if (pedalAngle < 0.0f)
      pedalAngle = 0.0f;
   if (pedalAngle >= 360.0f)
      pedalAngle -= 360.0f;
   glutPostRedisplay();
}
```

4 CONSIDERAÇÕES FINAIS

Este projeto de simulação de uma bicicleta 3D, implementado utilizando OpenGL, destacou-se como uma valiosa oportunidade de aplicar e aprofundar conhecimentos em computação gráfica. Através deste trabalho, foi possível explorar técnicas avançadas de modelagem, animação e interação com o usuário, proporcionando uma compreensão mais profunda dos desafios e soluções na área de gráficos em tempo real.

A implementação de funções para a animação da bicicleta, como a rotação das engrenagens e o movimento dos pedais, permitiu não apenas a visualização de conceitos físicos e mecânicos, mas também a manipulação de elementos gráficos interativos, que são fundamentais em aplicações de realidade virtual e jogos. Além disso, o uso de iluminação e texturização contribuiu para um realismo maior no ambiente simulado, oferecendo uma experiência visual mais rica e engajante.

Os desafios encontrados durante o desenvolvimento do projeto, especialmente relacionados ao cálculo de movimentos e à resposta a interações do usuário, reforçaram a importância de uma boa base em matemática e física para a computação gráfica. A resolução de problemas específicos, como o cálculo de ângulos de direção e a implementação de interações intuitivas com o teclado e mouse, foi instrumental para o sucesso do projeto.

REFERÊNCIAS

Cohen, M., Manssour, I. H. **OpenGL - Uma Abordagem Prática e Objetiva**. 1ª ed. São Paulo: Novatec, 2006.

Gregorini, D. **Projeto Final de Computação Gráfica**. Disponível em: https://github.com/DanielGregorini/projeto-final-computacao-grafica. Acesso em:24/06/2024.