Instituto Federal do Norte de Minas Gerais
Campus Montes Claros
Bacharelado em Ciência da Computação
Computação Gráfica

PRÁTICA 4

MODELAGEM 3D DE UMA RODA GIGANTE FUNCIONAL COM OPENGL

Discente:
João Vitor Ribeiro Botelho

Docente:
Prof. Dr. Wagner Ferreira de Barros

Março
2025

Sumário

1 Objetivo

Este trabalho tem como objetivo desenvolver um modelo hierárquico utilizando C/C++ com OpenGL e GLUT para representar uma roda-gigante simplificada, explorando conceitos de computação gráfica e transformações geométricas.

2 Introdução

O OpenGL (Open Graphics Library) é uma API (Application Programming Interface) amplamente utilizada para a criação de gráficos 2D e 3D. Desenvolvido originalmente pela Silicon Graphics Inc. (SGI) nos anos 90, o OpenGL se tornou um dos principais padrões da indústria para renderização gráfica em tempo real, sendo utilizado em diversas áreas como jogos, simulações, visualizações científicas e aplicações de CAD.

Sua principal característica é ser uma API multiplataforma e independente de hardware, permitindo que desenvolvedores criem gráficos avançados sem depender de um único fabricante de GPU. O OpenGL fornece um conjunto de funções para manipulação de vértices, texturas, iluminação e shaders, tornando-se uma ferramenta essencial para quem trabalha com computação gráfica.

Ao longo dos anos, o OpenGL evoluiu significativamente, introduzindo novas versões e funcionalidades, como o uso de shaders programáveis através da GLSL (OpenGL Shading Language), que permite maior flexibilidade e desempenho no processamento gráfico (OPENGL, 1997-2025).

3 Estrutura hierárquica

A roda-gigante é composta por vários elementos interconectados, formando uma estrutura hierárquica. A estrutura está organizada da seguinte forma:

- **Base:** Estrutura fixa onde a roda está montada.
- **Roda Principal:** Girada em torno de um eixo central.
- **Cabines:** Anexadas à roda principal e giram de forma independente para manter sua orientação.

A hierarquia é modelada utilizando transformações geométricas em OpenGL, garantindo que os elementos filhos (cabines) se movam em relação ao pai (roda principal).

4 Implementação das Movimentações

A movimentação do modelo foi implementada utilizando transformações geométricas, como rotação e translação. Abaixo estão os detalhes das principais movimentações:

4.1 Movimentação da Roda

A roda principal realiza uma rotação em torno de seu eixo central, simulando o movimento real de uma roda-gigante:

$$R = R_{inicial} + \omega * t \tag{1}$$

Onde:

- $R_{inicial}$ é a posição inicial da roda.
- ω é a velocidade angular da roda.
- t representa o tempo.

A rotação é aplicada com a função `glRotatef`.

```
glPushMatrix();
    glRotatef(anguloRoda, 0.0f, 0.0f, 1.0f);
    desenhaRoda();
glPopMatrix();
```

Figura 1 – Trecho mencionado, no código

4.2 Movimentação das Cabines

As cabines precisam manter sua orientação vertical independentemente da rotação da roda. Isso é alcançado aplicando uma transformação de rotação inversa:

$$\Theta_{cabine} = -\Theta_{roda} \tag{2}$$

Onde:

- Θ_{cabine} é o ângulo aplicado a cada cabine para corrigir sua orientação.
- Θ_{roda} é o ângulo da roda principal.

```
glPushMatrix();
    glRotatef(angulo, 0.0f, 0.0f, 1.0f);
    glTranslatef(2.0, 0.0, 0.0);

    // Aplica o balanço nas cabines antes de desenhá-las
    glRotatef(anguloCabineBalanço, 0.0f, 0.0f, 1.0f);

    glRotatef(-angulo, 0.0f, 0.0f, 1.0f);
    glRotatef(-anguloRoda, 0, 0, 1); // Mantém as cabines na vertical
```

Figura 2 – Trecho mencionado, no código

Isso é realizado dentro do loop de desenho das cabines, garantindo que cada uma gire corretamente.

4.3 Movimentação da Câmera

A câmera foi implementada utilizando transformações de visualização. A função *gluLookAt* é utilizada para definir a posição da câmera e seu ponto de observação:

$$gluLookAt(x_{eye}, y_{eye}, z_{eye}, x_{center}, y_{center}, z_{center}, x_{up}, y_{up}, z_{up},) \tag{3}$$

```
// Converte os ângulos esféricos para coordenadas cartesianas para controlar a posição da câmera
float cameraX = distanciaCamera * cos(anguloElevação) * sin(anguloAzimute);
float cameraY = distanciaCamera * sin(anguloElevação);  // Controla a altura (elevação)
float cameraZ = distanciaCamera * cos(anguloElevação) * cos(anguloAzimute);

// Configuração da câmera
gluLookAt(cameraX, cameraY, cameraZ,   // Posição da câmera com base nos ângulos
          0.0, 0.0, 0.0,   // Para onde a câmera aponta (centro da roda)
          0.0, 1.0, 0.0);  // Vetor "up" (mantido fixo no eixo Y)
```

Figura 3 – Trecho mencionado, no código

Essa abordagem permite que a câmera se mova ao redor da roda-gigante, proporcionando diferentes perspectivas.

5 Iluminação e efeitos metalizados

A iluminação da cena foi configurada para criar um ambiente realista, onde a luz ambiente suaviza e uniformiza a iluminação geral, enquanto uma luz difusa intensa destaca os contornos e texturas dos objetos, conferindo profundidade à cena. Além disso, foi aplicada uma forte componente especular, com um elevado coeficiente de brilho (shininess), para simular os reflexos característicos de superfícies metálicas, enfatizando o aspecto polido das hastes da roda gigante. Essa combinação de luz ambiente, difusa e especular, em conjunto com o uso de `GL_COLOR_MATERIAL`, permite que as cores definidas para as cabines interajam de forma natural com a iluminação, resultando em um equilíbrio visual harmonioso que realça tanto as cores vibrantes das cabines quanto o efeito metálico das estruturas.

6 Movimentação através do teclado

Para movimentar a câmera pode utilizar-se as seguinte teclas:

- ”w”(Rotaciona para cima)
- ”s”(Rotaciona para baixo)
- ”d”(Rotaciona para direita)
- ”a”(Rotaciona para esquerda)

Para aumentar ou diminuir a visualização:

- "e"ou "+"(Aumenta)
- "q"ou -"(Diminui)

Para parar a rotação da roda gigante:

- "l"(Pausa/continua)

Após pausar o movimento da roda gigante, as cabines se movimentarão em torno do seu próprio eixo por um determinado tempo. Simulando a inércia do movimento.

7 Compilação

Foi utilizado um makefile para diretivas de compilação.
Execute o código com:
`make run`

8 Capturas de tela

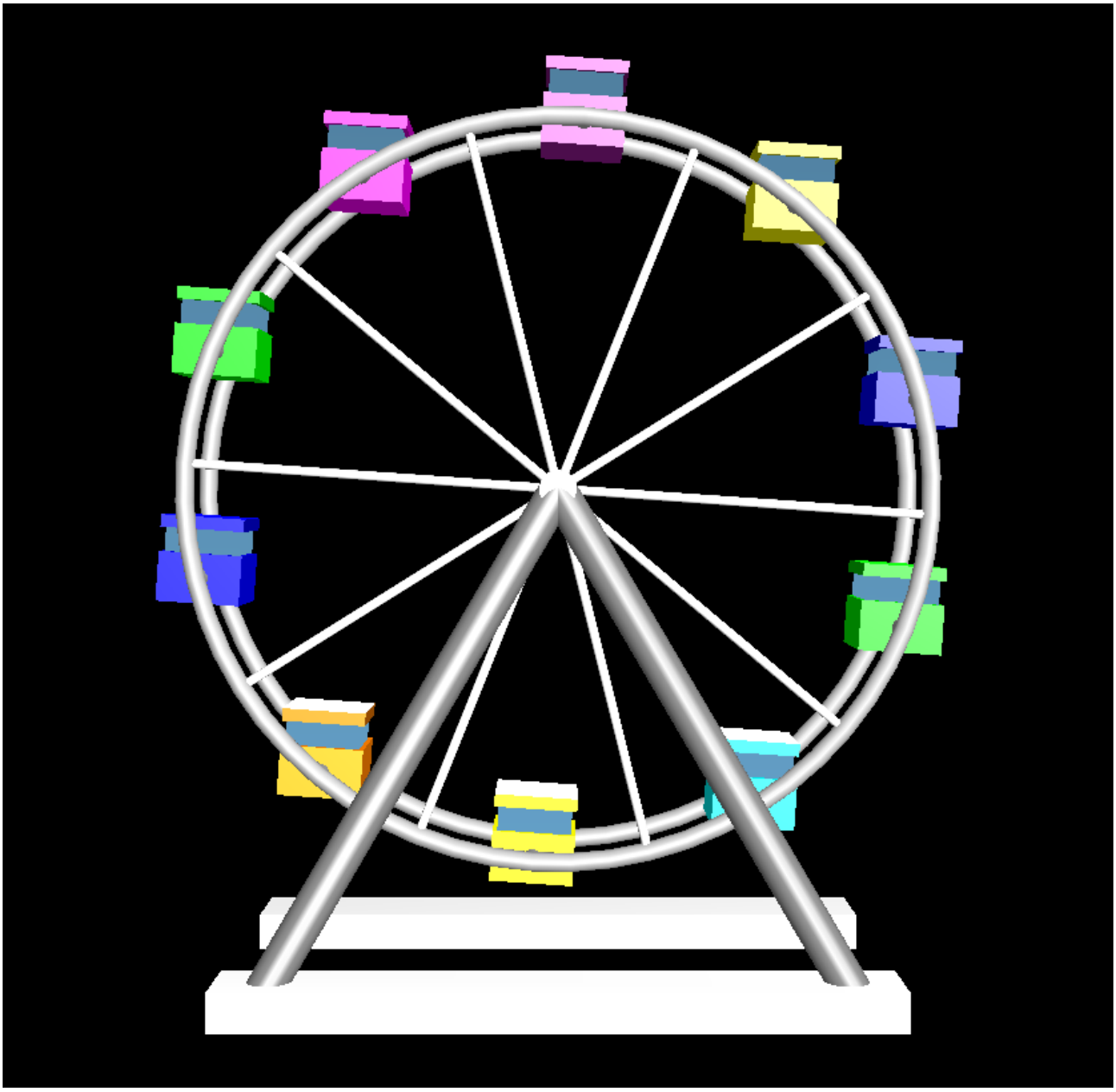

Figura 4 – Visão Frontal

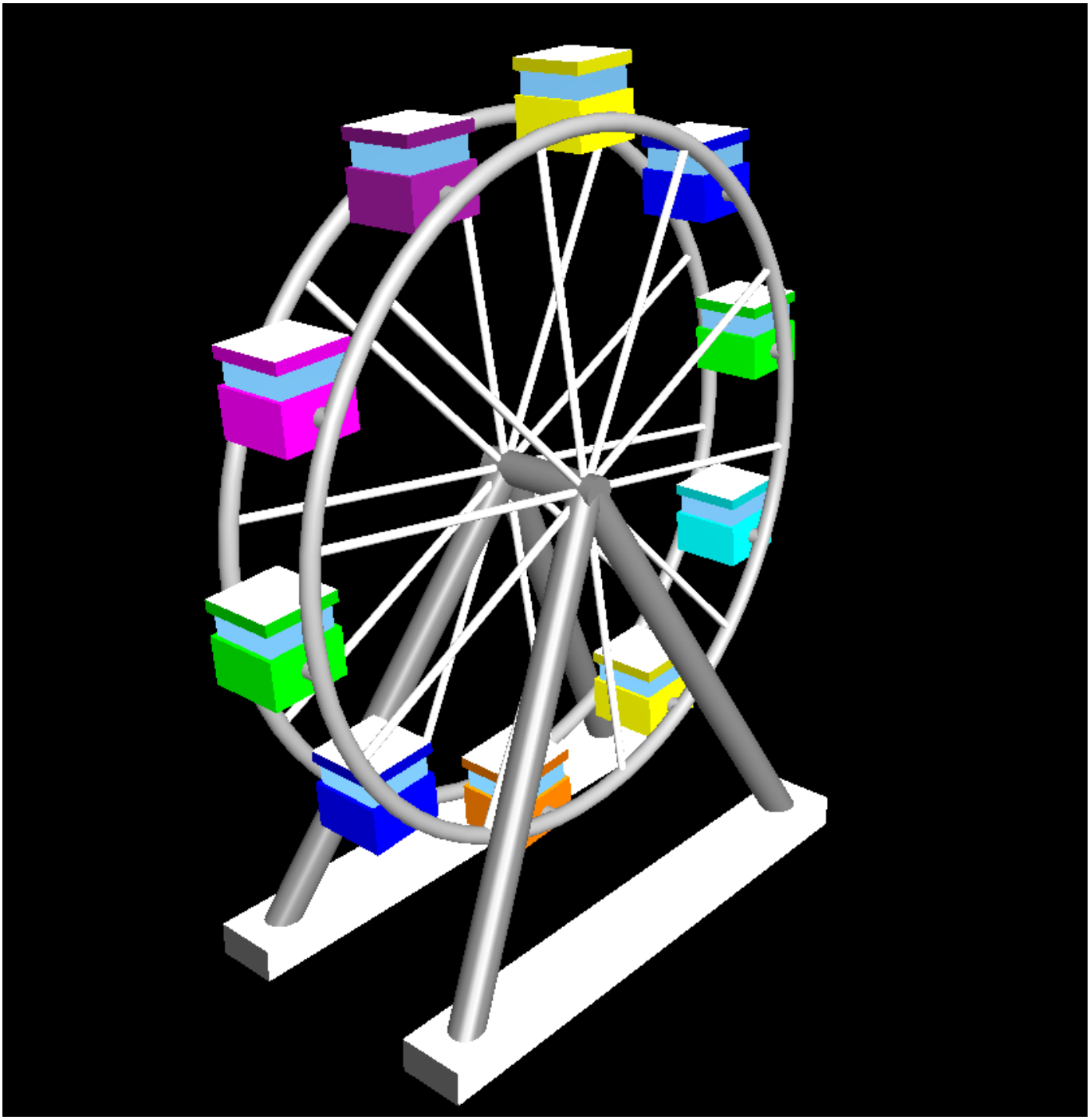

Figura 5 – Visão em perspectiva

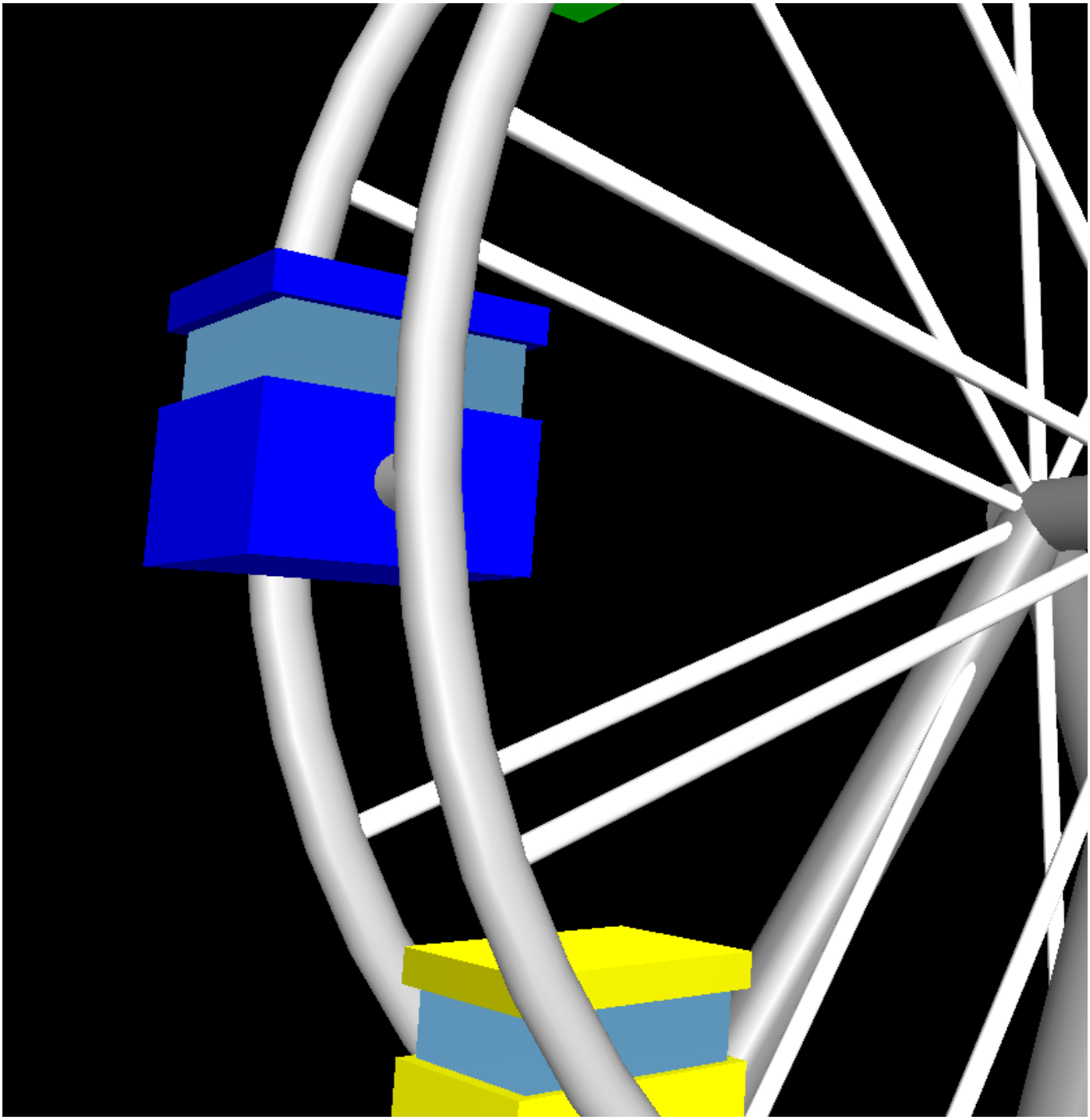

Figura 6 – Cabine

9 Considerações Finais

O programa utiliza conceitos fundamentais de OpenGL e GLUT para criar uma cena interativa. A estrutura modular do código facilita a manutenção e expansão do projeto.

Referências

OPENGL. *OpenGL API Documentation Overview*. 1997–2025. ⟨https://www.opengl.org/Documentation/Specs.html⟩. Accessed: March 11, 2025.