Computação Gráfica - TP2

Transformações Geométricas

29.03.2025

André Carvalho a100818,

Flávio Sousa a100715,

Vicente de Carvalho Castro a91677,

Índice

1. Introdução

Nesta Fase 2, o objetivo principal é criar estruturas de forma hierárquica, aplicando transformações geométricas como translação, rotação e escala. Isto significa que cada objeto na cena pode ter transformações próprias, mas também pode herdar as transformações dos seus "pais" na hierarquia, respeitando a ordem em que são aplicadas.

Para isso, foi necessário ajustar a abordagem do nosso motor gráfico (engine), garantindo que consiga suportar hierarquias de transformações de forma eficiente.

2. Engine

Explicamos então... Acrescentamos as seguintes estruturas de dados:

- **Transformation:** Armazena transformações 3D.

A estrutura guarda o tipo de transformação (0 = translate, 1 = rotate, 2 = scale) e os parâmetros da mesma

- **Transform:** Representa um modelo 3D e as suas transformações associadas.

A estrutura armazena transformações ordenadas, os parâmetros de translação, rotação e escala do modelo, bem como os pontos que definem a sua geometria. Permitindo assim organizar e aplicar facilmente as transformações necessárias para renderizar corretamente os modelos na cena 3D

3. Leitura e Renderização

Divisão da leitura do XML

Na primeira fase, a função *lerXM* lia e processava tudo diretamente. Agora, foi criada a função *processGroup* trata cada group recursivamente. Esta função percorre os elementos *group* do XML e aplica transformações de translação, rotação e escala. As transformações são armazenadas ordenadamente no vetor "transformations" dentro da estrutura Transform, garantindo que a ordem de aplicação das transformações seja preservada. Se um grupo contém modelos (), os ficheiros são carregados com readFile. A função chama a si mesma recursivamente para processar subgrupos.

"Flow Leitura XML": a função *lerXML* (Inicia a leitura do XML)
1. Carrega e verifica o ficheiro XML.
2. Processa a câmara (), extraindo os parâmetros necessários.
3. Chama *processGroup* para iniciar a leitura recursiva dos grupos.

Modularização da renderização Antes, a função *renderScene* renderizava tudo. Nesta Fase 2, criou-se a função *renderGroup* que renderiza cada grupo separadamente. Para cada transformação lida no XML aplica translação(glTranslatef),

rotação (glRotatef) e escala (glScalef). Renderiza os vértices associados a esse grupo usando GL_LINES. Define cores diferentes para os modelos, usando um esquema de cores baseado no índice.

"Flow" Renderização: a função *renderScene* (Chama renderGroup para desenhar todos os objetos)

1. Limpa o buffer e aplica *gluLookAt* para definir a câmara.
2. *Chama renderGroup* para cada conjunto de transformações armazenado.

4. Resultados obtidos

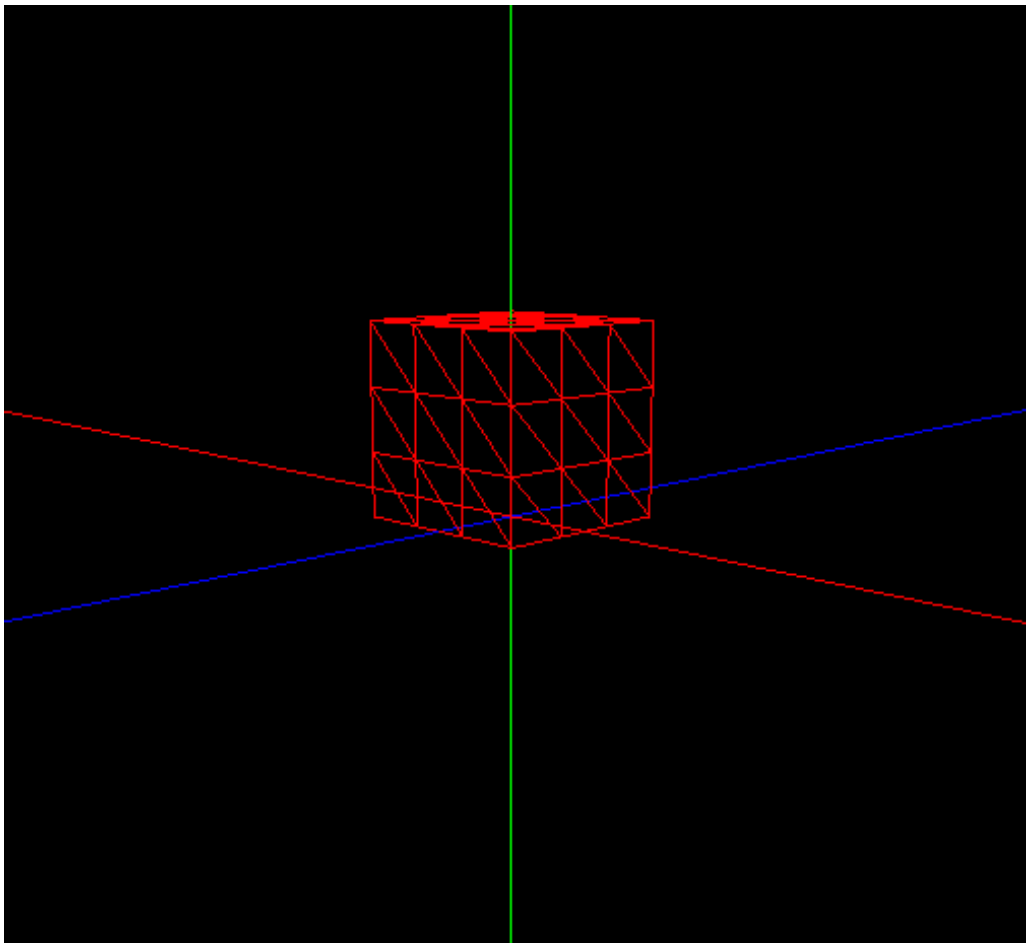

Figura 1: Cubo - teste 2.1

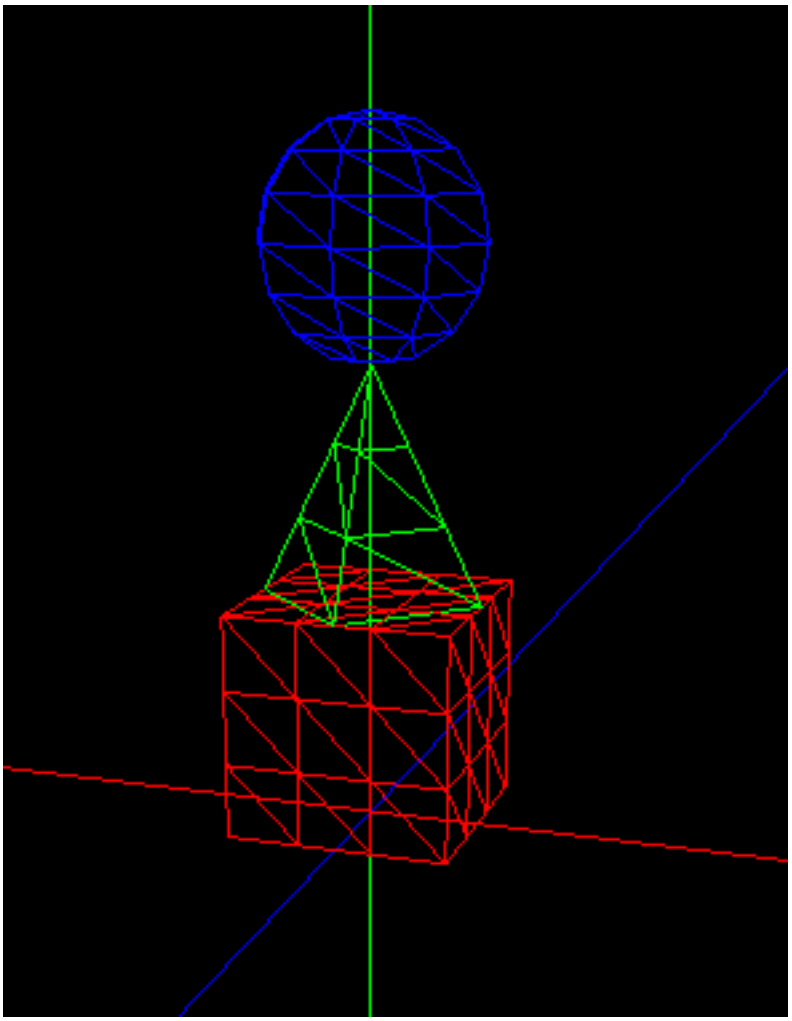

Figura 2: Cubo + Pirâmide + esfera - teste 2.2

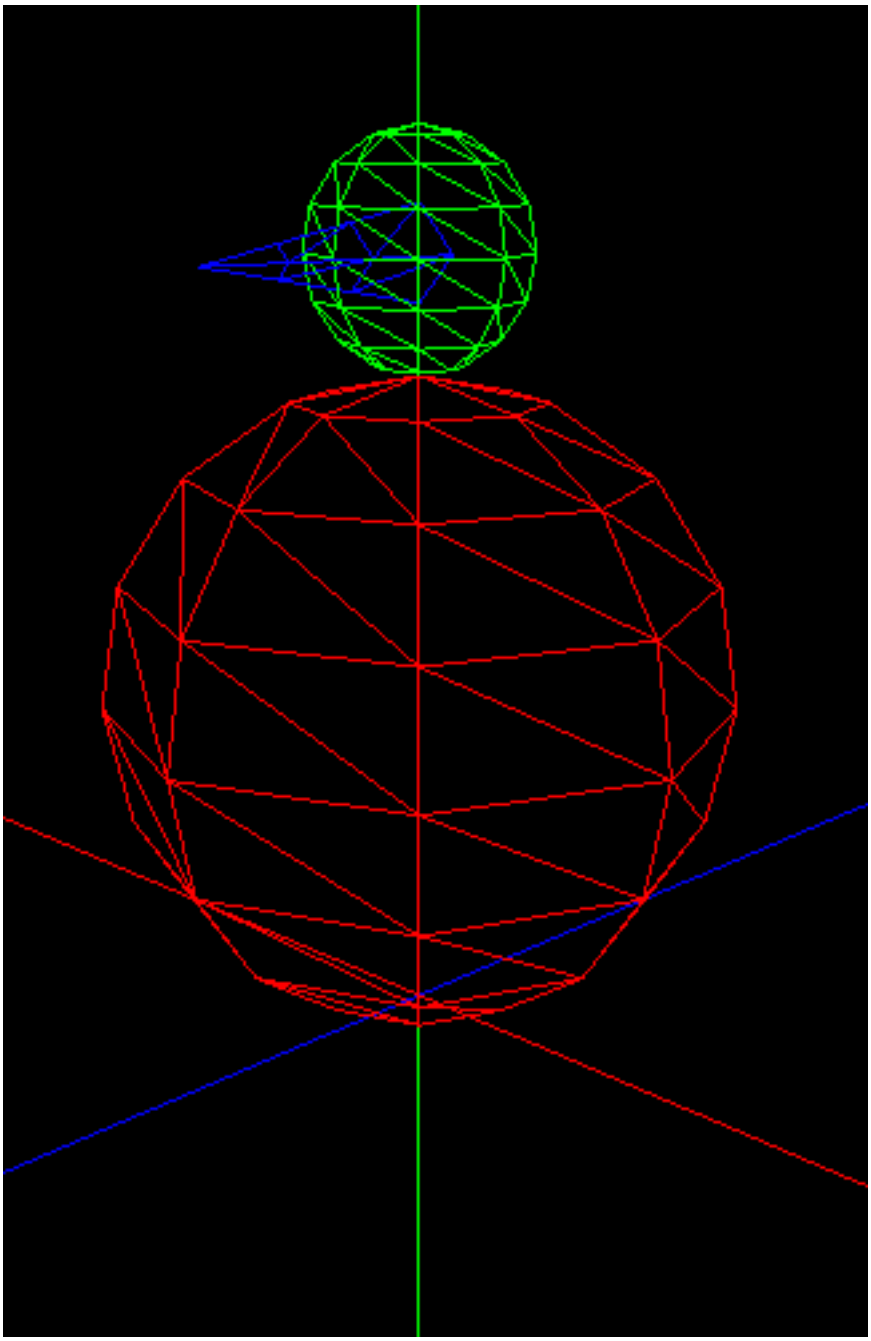

Figura 3: Duas esferas + Cone - teste 2.3

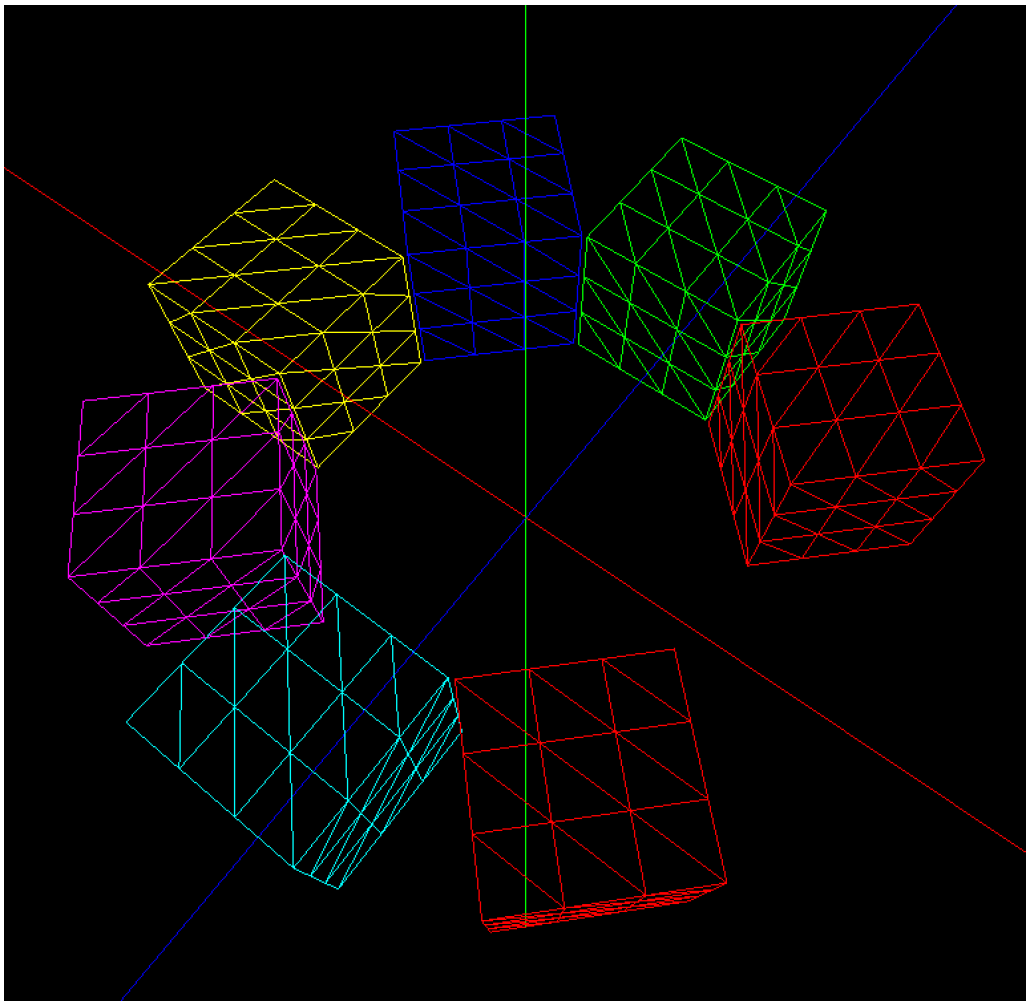

Figura 4: Sequência de Cubos - teste 2.4

5. Sistema Solar

O desafio principal para esta etapa foi encontrar um equilíbrio entre representar fielmente as dimensões dos planetas e as distâncias entre eles. Optou-se por uma abordagem que mantém proporções relativas, permitindo visualizar tanto os tamanhos comparativos quanto as distâncias aproximadas entre os corpos celestes.

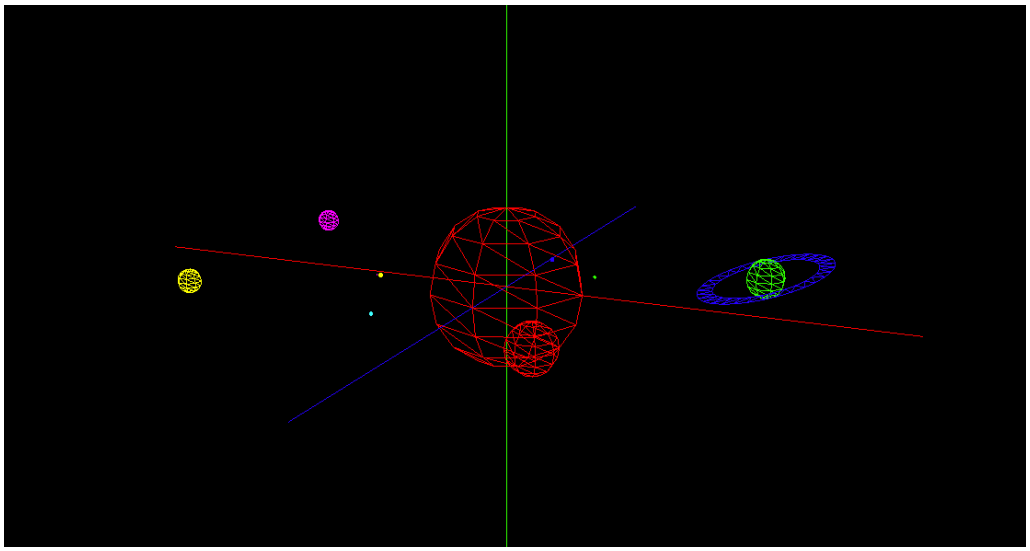

Figura 5: Sistema Solar c/ eixos

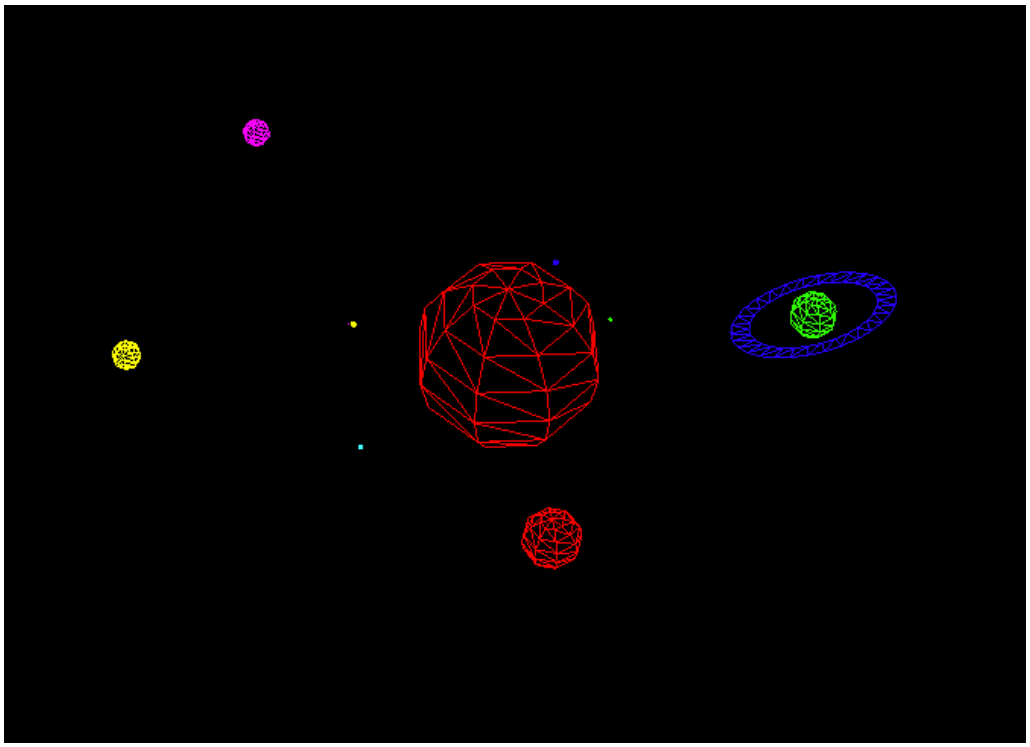

Figura 6: Sistema Solar c/ eixos

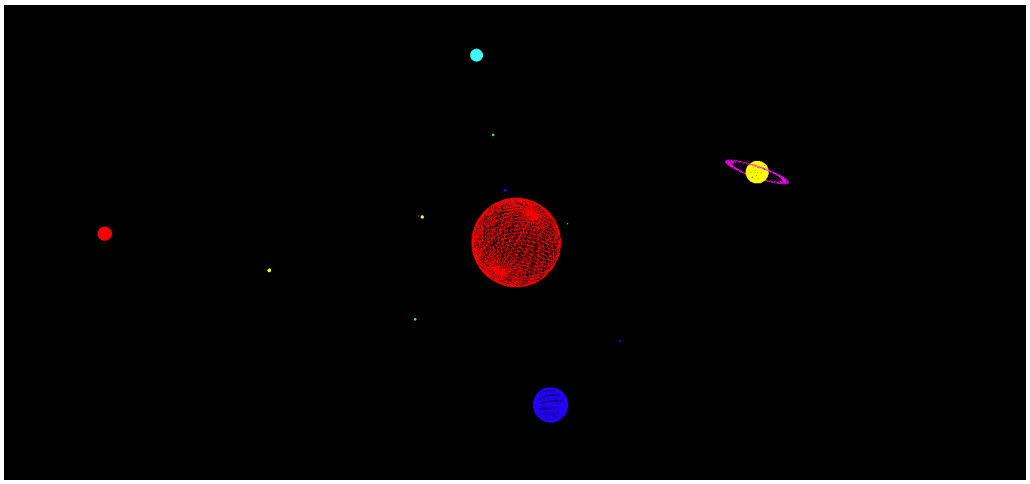

Figura 7: Sistema Solar c/ eixos

Implementação

O modelo foi desenvolvido em um único arquivo XML (solar-system.xml) contendo todas as transformações necessárias. A hierarquia de transformações foi organizada de forma clara, com cada planeta sendo tratado como uma transformação independente. Características especiais implementadas:

Anel de Saturno: Criado utilizando o arquivo ring.cpp no generator, que gera um modelo de anel circular para representar os característicos anéis de Saturno.

Estrelas: Adicionadas como pequenos pontos luminosos no fundo para criar um ambiente mais realista.

Lua: Implementada como satélite natural da Terra, com proporções e posição relativa adequadas.

O arquivo XML único contém toda a hierarquia de transformações necessárias para visualizar corretamente o sistema

6. Conclusões e trabalho futuro

Desta forma conseguimos obter uma Maior Organização visto que existe a divisão clara entre leitura e renderização. Modularização uma vez que usamos funções independentes para processar grupos e renderizá-los. Recursividade, pois grupos são processados hierarquicamente, permitindo estruturas 3D complexas