



Computação Gráfica

Fase 2 - Geometric Transforms

LEI - 2023/2024

GRUPO 23

Ana Margarida Sousa Pimenta – A100830
Inês Gonzalez Perdigão Marques – A100606
Miguel Tomás Antunes Pinto – A100815
Pedro Miguel Costa Azevedo – A100557

Índice

1	Introdução	2
2	Estruturação do projeto	2
3	Engine	4
4	Demonstração	5
	4.1 Manual de utilização	6
5	Resultado dos Testes	6
6	Sistema Solar	7
7	Conclusão	9

1 Introdução

A segunda fase do projeto da disciplina de Computação Gráfica representou uma etapa crucial na evolução e aprimoramento da Engine desenvolvida anteriormente. Nesta etapa, foram introduzidas funcionalidades fundamentais para a expansão das capacidades da Engine, com foco principal na aplicação hierárquica de transformações geométricas aos modelos tridimensionais. Essas transformações foram especificadas de forma detalhada num ficheiro XML com um formato específico, permitindo uma descrição precisa da cena a ser renderizada.

Uma das principais metas desta fase foi habilitar a Engine para lidar com a complexidade das cenas tridimensionais, possibilitando a organização hierárquica dos elementos e a aplicação sistemática de transformações geométricas em diferentes níveis da hierarquia. Para validar o sucesso destas implementações, foram criados dois ficheiros XML que descrevem as instruções necessárias para a Engine renderizar um Sistema Solar, demonstrando assim a capacidade da Engine em lidar com estruturas complexas e dinâmicas.

Este relatório detalhará as decisões e abordagens adotadas durante o processo de desenvolvimento, destacando os desafios enfrentados e as soluções encontradas para garantir a eficácia da Engine. No final, será evidente o progresso alcançado e as bases sólidas estabelecidas para futuras iterações e aprimoramentos da Engine, consolidando assim a sua posição como uma ferramenta versátil e poderosa para a renderização de cenas tridimensionais.

2 Estruturação do Projeto

As estruturas definidas no código são utilizadas para configurar e descrever o ambiente tridimensional em que os modelos serão renderizados. Vou explicar cada uma delas:

Window (Janela):

Armazena as dimensões da janela de visualização, representada pela sua largura e altura.

Camera(Câmera):

Define os parâmetros da câmera, como posição, ponto para onde está a olhar, orientação para cima e informações de projeção (campo de visão, plano próximo e plano distante).

Point3D(Ponto3D):

Representa um ponto no espaço tridimensional, definido pelas suas coordenadas x, y e z.

Model (Modelo):

Descreve um modelo 3D, composto por um conjunto de vértices. Possui um nome para identificação e uma lista de pontos 3D que formam o modelo.

Transform (Transformação):

Define uma transformação geométrica que pode ser aplicada a um modelo ou a um grupo de modelos. Pode incluir operações de translação, rotação e escala, além de indicadores para verificar se cada tipo de transformação está presente.

Group (Grupo):

Representa um grupo de modelos e subgrupos, organizados hierarquicamente. Cada grupo possui uma transformação que afeta todos os modelos e subgrupos contidos nele.

World (Mundo):

Contém as configurações globais do ambiente, incluindo a janela de visualização, a câmera e uma lista de grupos que compõem a cena.

Estas estruturas são utilizadas para definir a configuração inicial do ambiente tridimensional e organizar os modelos e as suas transformações antes de serem renderizados na tela. O código também contém funções para processar eventos do teclado,

como mudança na posição da câmera, e funções para renderizar os modelos e grupos na tela.

3 Engine

Este código é parte de uma aplicação de engine gráfica que permite renderizar cenas tridimensionais definidas em ficheiros XML. A implementação atual visa criar uma hierarquia de cenas utilizando transformações geométricas, como translação, rotação e escala. Vamos analisar cada parte do código de acordo com o que foi solicitado no enunciado.

Estruturas de Dados:

São definidas várias estruturas de dados para representar elementos da cena, como janelas, câmeras, modelos, transformações e grupos.

Função `spherical2Cartesian()`:

Esta função calcula as coordenadas cartesianas da câmera a partir das coordenadas esféricas (raio, alfa e beta).

Função `renderModel()`:

Renderiza um modelo definido por uma lista de vértices. A renderização é feita duas vezes: uma vez em modo sólido e outra em modo wireframe.

Função `renderGroup()`:

Renderiza um grupo, aplicando as transformações geométricas definidas para esse grupo e, em seguida, renderiza os modelos e subgrupos pertencentes a ele.

Função `display()`:

Define a função de exibição da cena. Configura a projeção e a câmera, em seguida, renderiza todos os grupos da cena.

Função `initialize()`:

Inicializa o ambiente gráfico OpenGL, configurando o teste de profundidade e a cor de fundo. Também calcula as coordenadas cartesianas iniciais da câmera.

Função readModel():

Lê um ficheiro de modelo (.3d) e retorna uma estrutura contendo os vértices desse modelo.

Função parseXML():

Analisa um ficheiro XML que descreve a cena, incluindo informações sobre a janela, câmera, grupos e modelos. Essas informações são armazenadas na estrutura de dados **World**.

Funções processKeys() e processSpecialKeys():

Lidam com a interação do usuário, como alterar a posição e a direção da câmera, bem como ajustar o zoom.

Função reshape():

Redefine a matriz de projeção quando a janela é redimensionada.

Função main():

Inicia a aplicação, inicializa o OpenGL, lê o ficheiro XML da cena, configura a janela e as funções de callback, e inicia o loop principal do OpenGL.





Conclusão:

Este código implementa a estrutura básica para renderização de uma cena 3D hierárquica, permitindo a aplicação de transformações geométricas em grupos de modelos. Essas transformações são especificadas num ficheiro XML, juntamente com outras informações da cena. A implementação atual é um passo importante para o desenvolvimento de uma engine gráfica mais completa e funcional.

4 Demonstração

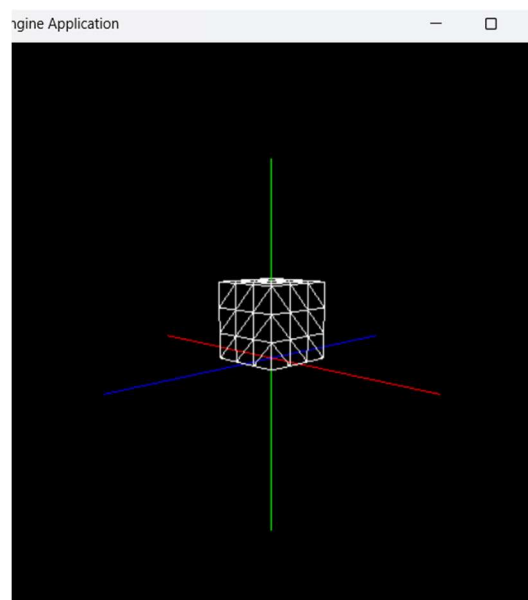
4.1 Manual de utilização

De forma a tornarmos o nosso sistema mais iterativo adicionamos os seguintes comandos:

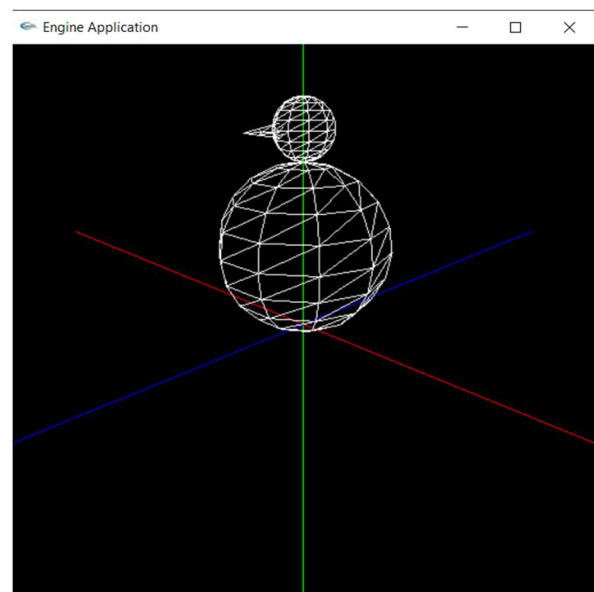
-  rotação da câmara para a esquerda
-  rotação da câmara para a direita
-  rotação da câmara para baixo
-  rotação da câmara para cima
- + zoom in
- zoom out
- q sair do programa

5 Resultado dos Testes

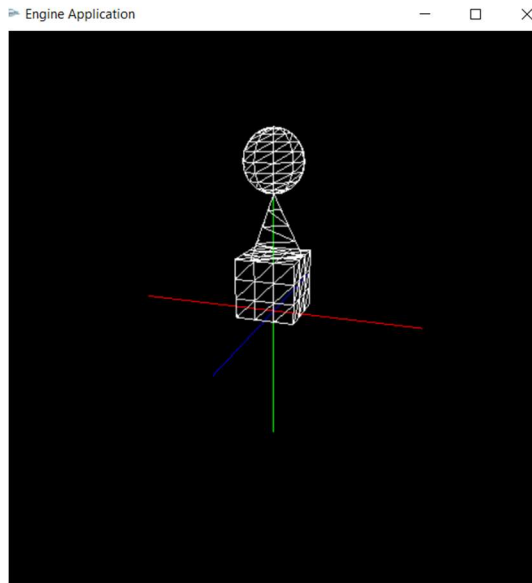
Neste ponto, demonstramos os resultados de todos os testes disponibilizados na BlackBoard.



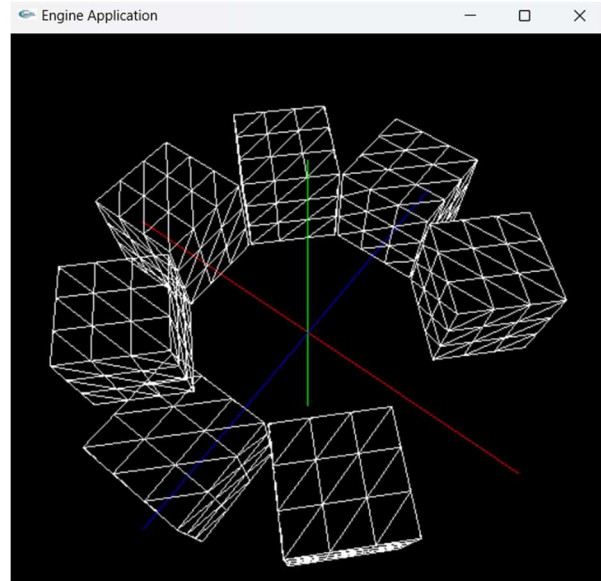
teste 2.1



Teste 2.2



Teste 2.3



Teste 2.4

6 Sistema Solar

Para a representação do sistema solar neste trabalho optamos por fazer dois tipos de xml diferentes para 2 tipos de representação diferentes.

Primeiramente corremos um ficheiro XML, onde usamos a função translate em relação ao sol para todos os planetas. Usamos ainda a mesma função para as respetivas luas de cada planeta, porém não em relação ao sol, mas sim o seu planeta correspondente.

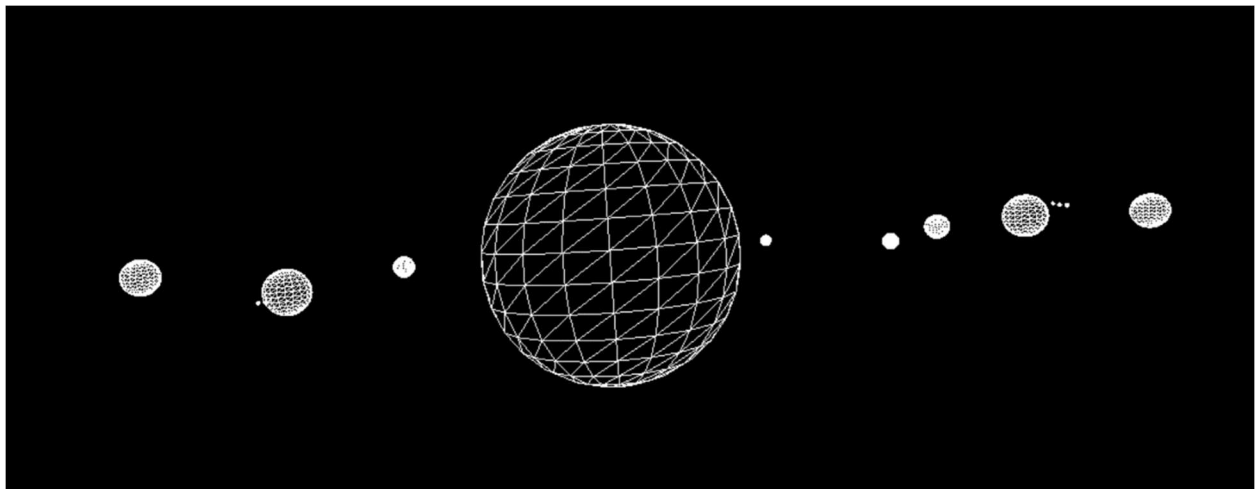


Imagem do sistema solar, através do SolarSystem.xml

Para o segundo XML, já usamos dimensões mais realistas para os planetas e para o sol. Mudamos também ainda a forma como os planetas são desenhados, sendo que, ao contrário do primeiro xml (SolarSystem.xml) a forma como estes são desenhados é aplicada a função translate relativamente ao planeta anterior. Neste xml metemos ainda os planetas alinhados no eixo dos X.

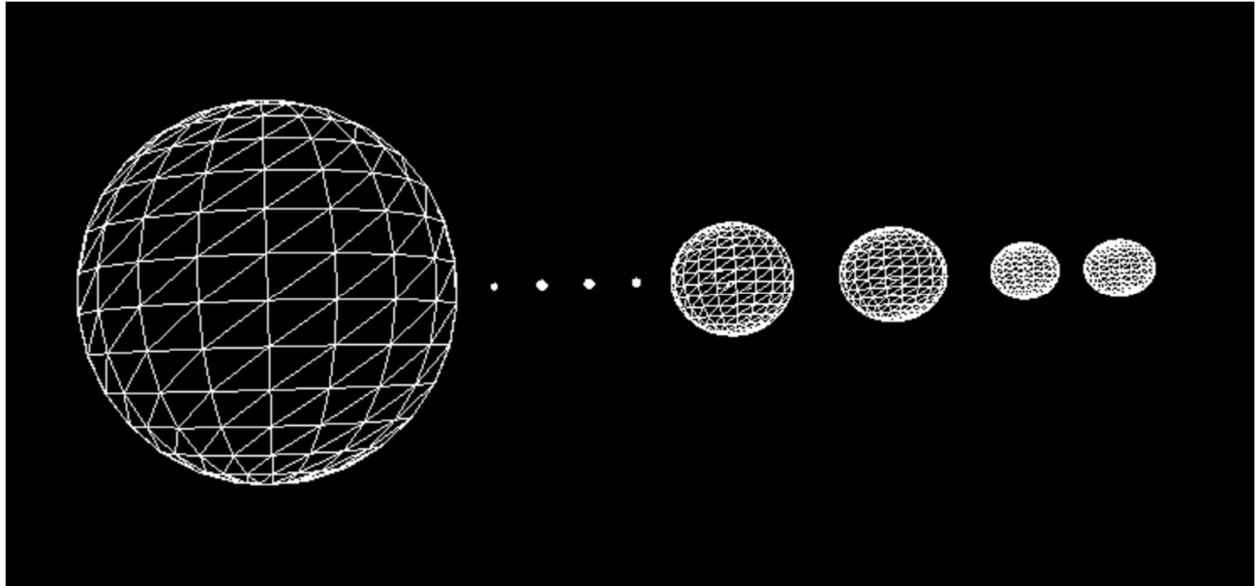


Imagem do sistema solar, através do SolarSystemAligned.xml

7 Conclusão

Nesta segunda fase do projeto, concentramo-nos na implementação de uma hierarquia de cenas utilizando transformações geométricas em modelos 3D. Ao visitar e expandir o código desenvolvido na fase anterior, conseguimos avançar significativamente no entendimento e na aplicação dos conceitos de renderização gráfica em um ambiente tridimensional.

Ao longo deste trabalho, enfrentamos diversos desafios, desde a análise e interpretação de especificações de cenários até a implementação eficaz de funcionalidades dentro do ambiente OpenGL. Foi gratificante observar a evolução em termos de organização do código, compreensão dos conceitos e habilidades de resolução de problemas.

É notável o progresso alcançado em relação à fase anterior, evidenciado pela implementação de transformações geométricas em grupos de modelos, a leitura e interpretação de arquivos XML para definir as cenas e a interação do usuário para manipulação da câmera.

Além disso, reconhecemos que há espaço para melhorias contínuas e otimizações adicionais no código e na funcionalidade da engine gráfica. Temos várias ideias para aprimorar ainda mais este trabalho, sendo a principal, a adição de um “anel” no generator, para acrescentar no sistema solar, à volta de saturno.

Em suma, esta segunda fase representou um avanço significativo no projeto, e estamos animados para continuar a explorar e aprimorar o desenvolvimento de engines gráficas, procurando sempre oferecer experiências visuais mais ricas e envolventes.