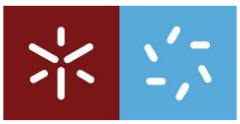

Universidade do Minho

Licenciatura em Ciências da Computação

Fase 2 - Transformações Geométricas

Computação Gráfica

Grupo 7

Cláudia Rego Faria
(A105531)

Diogo José Borges Dias
(A102943)

Maria Inês Barros de Matos
(A102937)

Patrícia Daniela Fernandes Bastos
(A102502)

30 de março de 2025

Conteúdo

Capítulo 1

Introdução

Este relatório é o segundo de um trabalho de quatro partes, no âmbito da UC Computação Gráfica da Licenciatura em Ciências da Computação e referente ao ano letivo de 2024/2025.

Para a Segunda Fase, foram feitas alterações ao *Engine* para suportar a criação de um Sistema Solar, para já estático. Esta fase teve como objetivo a aprendizagem e implementação de transformações geométricas aplicadas de forma hierárquica, usando os conhecimentos adquiridos na fase anterior.

Capítulo 2

Estrutura do projeto

Para uma melhor organização do projeto, este foi dividido nos seguintes diretórios:

- **3d**: armazena os ficheiros criados pelo generator que serão posteriormente utilizados pelo engine para a renderização dos modelos
- **engine**: contém a implementação do motor gráfico responsável pela leitura e visualização dos modelos 3d gerados
- **gen**: inclui o código responsável pelo cálculos dos pontos necessários para a construção das primitivas e pela geração dos ficheiros 3d
- **data_structs**: contém as structs desenvolvidas para um melhor e mais fácil funcionamento do projeto
- **test_files**: contém os ficheiros de teste que serão utilizados ao longo das diferentes fases do projeto
- **TinyXml**: contém a biblioteca *TinyXML* utilizada para auxiliar na leitura e manipulação de ficheiros .xml
- **demo_scenes**: contém o ficheiro XML para a verificação desta segunda fase, neste caso o Sistema Solar
- **outputs**: contém o ficheiro *CMakeLists.txt*

Capítulo 3

Engine

Nesta fase, foram feitas alterações ao *Engine* necessárias para a criação do modelo do Sistema Solar.

3.1 Novas estruturas de dados e outras alterações

Foram criadas duas novas estruturas:

- **Transform**: contém um caractere *type* (que corresponderá a T para translações, S para escalas e R para rotação), um *Point* para a transformação e um *float* para o ângulo. A esta estrutura foram criadas as funções associadas *newTransformation*, *getType*, *getTransVal* e *getAngle*.
- **Group**: contém uma lista de Modelos, uma lista de Transformações e uma lista de Grupos (os subgrupos). A esta estrutura foram criadas as funções associadas *creategroup*, *makeGroup*, *getModels*, *getTransformations* e *getSubgroup*.

No *Model*, à lista de *Points* que compõe a figura, foram adicionados 3 *floats*: *colorR*, *colorG* e *colorB*. Isto serve para, na representação do Sistema Solar, os planetas serem representados com diferentes cores para mais fácil distinção. Estes 3 valores vão compor o valor *RGB* da cor.

Foi ainda criada a função *parseGroup* e alterada a função *xmlToSettings*, de modo a adequar o projeto à nova estrutura hierárquica das transformações geométricas.

No engine foram também criadas função que colocam como título da janela a posição da câmara e os *frames per second* (*fps*).

3.2 Leitura do ficheiro XML

Para a leitura do ficheiro XML foi criada uma função principal denominada *xmlToSettings* que converte os dados do XML numa *struct Settings* previamente criada.

O processo de leitura começa por extrair os dados relativos às dimensões da janela (*window*) : largura (*width*) e altura (*height*) - que definem os parâmetros da área de renderização. Posteriormente vem a extração dos parâmetros da câmara, incluindo a sua posição inicial no espaço 3D (através das coordenadas x, y, z), o ponto para onde está a olhar (*lookAt*), o vetor *up* e os dados de projeção (*fov*, *near* e *far*).

Após a extração destes dados é invocada a função *parseGroup* responsável por processar a hierarquia dos grupos definidos no XML. Esta função começa por inicializar três vetores fundamentais:

- *transformations*: armazena as transformções geométricas
- *models*: armazena os modelos 3D das figuras e a sua respetiva cor
- *subgroups*: armazena os subgrupos

A função começa por recolher os dados relativos às transformações geométricas e aos modelos do grupo.

Transformações

No processamento das transformações geométricas, o sistema identifica cada tipo de transformações (translação, escala ou rotação) pelo nome da tag XML correspondente para poder atribuir o seu tipo ao campo *type*.

Para cada uma vai extrair os seguintes parâmetros:

- componentes x, y e z
- angulo (*angle*), caso se trate de uma rotação

Cada transformação vai ser então adicionada ao vetor de transformações do grupo atual.

Modelos

Quanto aos modelos 3D, o sistema começa por verificar se existe algum campo *color* que altere a cor atual. Caso este exista vão ser extraídos os atributos *r*, *g* e *b* que irão ser responsáveis por alterar a cor de todos os modelos desse mesmo grupo.

Para cada modelo, irá ser lido o ficheiro .3d correspondente. Os pontos irão ser extraídos e armazenados numa *struct Model* junto com a respetiva cor, que irá ser adicionada ao vetor *models*.

Subgrupos

Para processar os subgrupos será invocada a função *parseGroup*, recursivamente, para tratar de todos os *groups* filhos do *group* que acabou de processar.

Após a recolha de todos os grupos ficará uma *struct Settings* completa com todos os dados do XML lido.

3.3 Desenho dos Modelos

O processo do desenho dos modelos é realizado através da função *drawFigures* que é invocada em *renderScene*. A função *drawFigures* recebe como argumento o *group* que contém toda a informação relativa para o desenho completo da cena e vai fazer uma travessia recursiva da *struct group*.

Quando a função é chamada, começa por obter três componentes essenciais do grupo atual: as transformações a aplicar, os modelos a desenhar e os subgrupos a processar. Imediatamente, salvaguarda o estado atual da matriz de transformação através de *glPushMatrix*, criando assim um contexto isolado para as operações seguintes.

Após fazer o *glPushMatrix* vão ser feitas as transformações geométricas pela mesma ordem que aparecem no ficheiro XML. Para cada transformação, a função indentifica o tipo (translação, escala ou rotação) e aplica a operação correspondente usando as primitivas do *OpenGL* apropriadas: *glTranslatef* para translações, *glScalef* para escalas e *glRotatef* para rotações.

Após ser configurado o sistema de coordenadas com todas as transformações do grupo atual, a função vai iniciar o desenho dos modelos. Cada modelo é processado como um conjunto de triângulos (*GL_TRIANGLES*) e com as suas cores próprias definidas por *glColor3f*. Todos os vértices de cada modelo vão ser desenhados através de *glVertex3f*.

O aspeto recursivo entra em ação quando a função processa os subgrupos. Para cada subgrupo, a função *drawFigures* é chamada novamente, mantendo assim a hierarquia definida na estrutura original. Esta abordagem garante que as transformações sejam acumulativas ,ou seja, cada subgrupo herda

as transformações dos seus pais enquanto pode adicionar as suas próprias modificações.

Finalmente, após processar todos os modelos e subgrupos do grupo atual, a função restaura o estado anterior da matriz de transformação com *glPopMatrix*. Este mecanismo é crucial para garantir que as transformações de um grupo não afetem outros grupos no mesmo nível hierárquico.

3.4 Câmara

Na primeira fase do projeto apenas era possível alterar a posição em que a câmara se encontrava de forma orbital em torno de um ponto fixo. Para facilitar a visualização do sistema solar, e de outras possíveis futuras cenas, a câmara foi alterada para uma câmara *FPS*. É possível alterar a sua posição e a sua direção livremente. Para tal, *alphaCam* e *betaCam* foram substituídas pelas coordenadas esféricas *yaw* e *pitch* e os cálculos alterados.

Em adição às funcionalidades de controlo da câmara referidas anteriormente e às já implementadas na primeira fase, foi também implementada a opção de ativar e desativar a visualização dos eixos *xyz*.

Movimento

W: Mover para a frente

S: Mover para trás

A: Mover para a esquerda

D: Mover para a direita

+: Mover para cima

- : Mover para baixo

Cursor: Alterar a direção da câmara

Modo de Desenho

M: Alternar entre preenchimento, linhas e pontos

P: Alternar entre mostrar ou não os eixos *xyz*

Capítulo 4

O Sistema Solar

4.1 Planetas, luas e astros

Para esta demonstração do Sistema Solar estático, foram representados os planetas Mercúrio, Vénus, Terra, Marte, Júpiter, Saturno, Urano e Neptuno. Foram ainda incluídas as principais Luas destes planetas

Esta representação não corresponde a uma escala proporcional à realidade, mas a uma estimativa que permitiu desenhar os corpos celestiais de forma simples e visível. Nas seguintes tabelas, encontram-se os valores usados para a escala estimada:

Sol e Planetas				
Astro	Diâmetro à escala	Diâmetro real (km)	Distância do Sol à escala	Distância do Sol real (km)
Sol	1.39	1391400	-	-
Mercúrio	0.004	4879	2.193	57900000
Venus	0.012	12104	2.36066667	108200000
Terra	0.012	12756	2.49866667	149600000
Marte	0.006	6792	2.75966667	227900000
Jupiter	0.142	142984	4.59533333	778600000
Saturno	0.12	120536	6.77833333	1433500000
Urano	0.051	51118	11.575	2872500000
Neptuno	0.049	49528	16.98366667	4495100000

Tabela 4.1: Escala estimada dos planetas e do sol

Luas				
Astro	Diâmetro à escala	Diâmetro real (km)	Distância do Planeta à escala	Distância do Planeta real (km)
Lua	0.20	3474	2	384400
Io	0.026	3643	2.4211	421700
Europa	0.022	3138	2.6699	670900
Ganymede	0.037	5268	1.068	1070000
Callisto	0.034	4821	1.880	1883000
Titan	0.043	5152	1.220	1222000
Triton	0.055	2707	2	354800

Tabela 4.2: Escala estimada das luas

4.2 Primitivas usadas

Os modelos para os corpos celestiais são gerados pelo Generator através de instruções no ficheiro XML, que indicam qual a figura e os seus parâmetros.

Para representar os corpos do Sistema Solar foi usada a primitiva da esfera criada na fase anterior.

4.2.1 Anel

Para a representação do anel de Saturno, foi utilizada a função *genRing()*, que gera um anel tridimensional a partir de dois raios (*innerRadius* e *outerRadius*) e de um número de divisões (*slices*). Essa abordagem permite criar um conjunto de triângulos organizados de forma circular, formando um anel ao redor do planeta.

A estrutura do anel é gerada da seguinte forma:

1. Cálculo dos pontos internos e externos:
    - Para cada fatia (*slice*), são calculados os pontos internos e externos do anel, baseados nos raios fornecidos.
    - A posição de cada ponto é definida em coordenadas polares, convertendo para coordenadas cartesianas usando funções trigonométricas (*sin* e *cos*).
2. Construção da malha do anel:
    - O anel é composto por um conjunto de triângulos que formam pequenos quadriláteros entre as bordas interna e externa.

- Para cada fatia, são criados quatro triângulos, dois na parte superior e dois na parte inferior, garantindo uma estrutura completa e simétrica.

3. Escrita dos pontos no ficheiro:

   - Os pontos gerados são armazenados em um vetor e, em seguida, gravados num ficheiro para serem utilizados na renderização da cena.

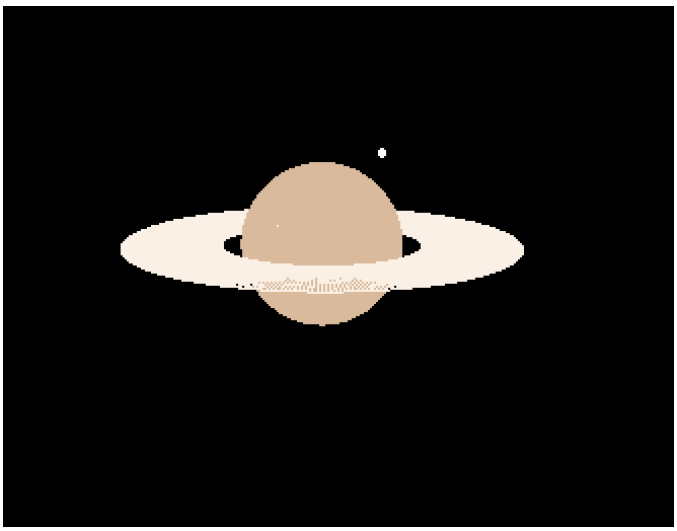

Figura 4.1: Saturno em *solar_system.xml*

4.3 Ficheiro XML do Sistema Solar

O ficheiro *solar_system.xml* representa um modelo 3D do nosso sistema solar. Segue-se um resumo da sua estrutura:

O Sistema Solar é organizado em elementos *group*, criando uma hierarquia. O Sol é o objeto principal, com os planetas como elementos secundários. Cada planeta é definido em um *group* junto com os elementos *transform* para posicionamento, rotação e escala *models* que definem a aparência. Por sua vez, as luas são definidas como subgrupos dos seus planetas.

Transformações aplicadas:

- Ao Sol é apenas aplicada uma escala, ficando no centro da cena.

- Aos planetas são aplicadas escalas para definir o seu tamanho, translações para afastar o astro do Sol e ainda uma rotação (de modo a espalhar os diferentes planetas pela órbita solar).
- No caso das luas, são aplicadas escalas e translações. Quando um planeta possui mais do que uma lua, são aplicadas ainda rotações para as separar.

Caraterísticas especiais:

- Saturno tem um modelo de anel para além da sua esfera.
- A Terra tem uma lua.
- Júpiter tem quatro luas (Io, Europa, Ganimedes, Calisto).
- Saturno tem a lua Titã.
- Neptuno tem a lua Tritão.

Todos os planetas utilizam o mesmo ficheiro de modelo 3D (*sphere.3d*) mas com cores, escalas e posições diferentes para criar a representação do sistema solar.

O ficheiro de exemplo fornecido gera a seguinte representação do Sistema Solar:

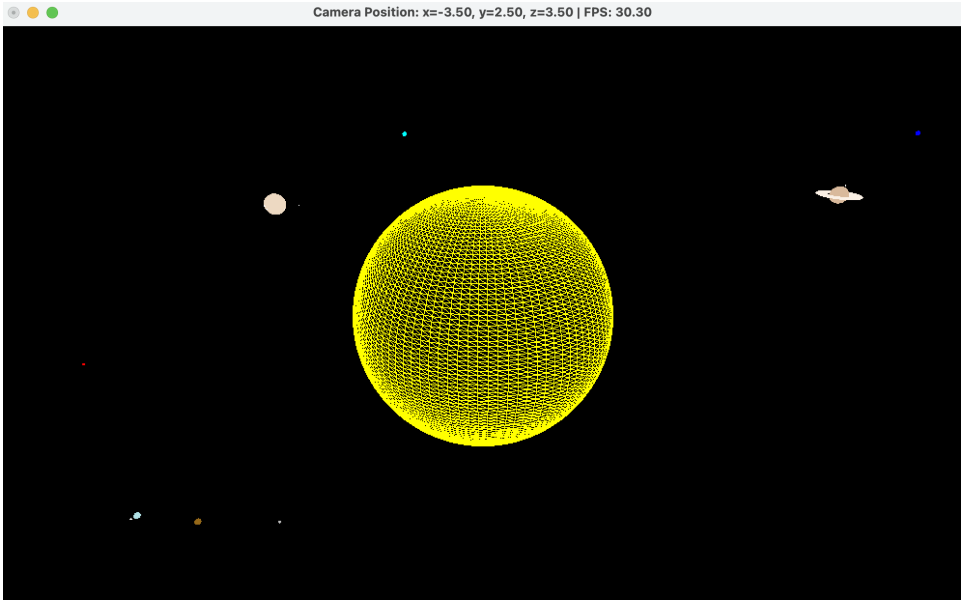

Figura 4.2: *solar_system.xml*

Capítulo 5

Testes

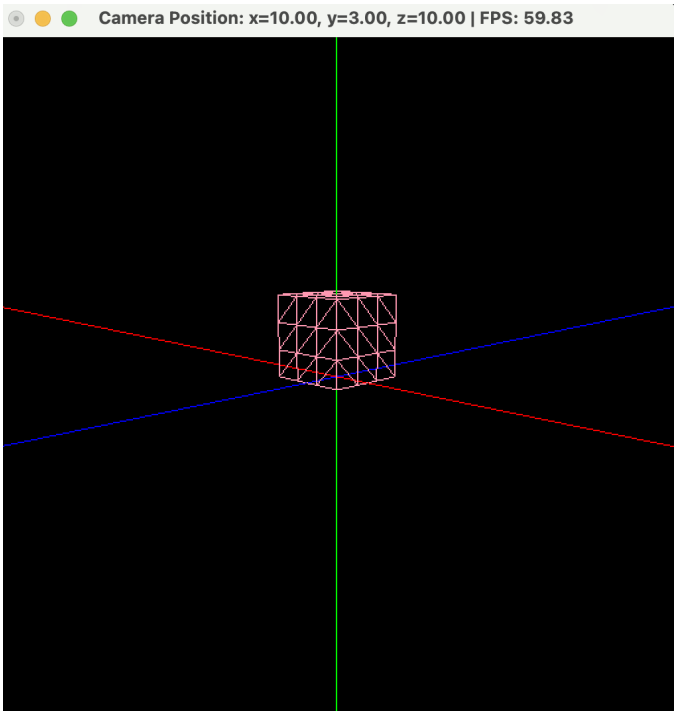

Figura 5.1: *test_2_1.xml*

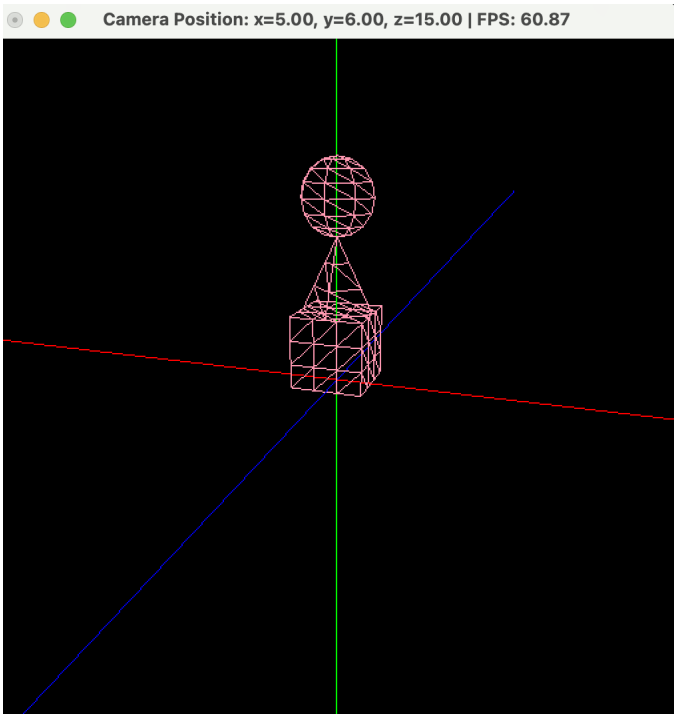

Figura 5.2: *test_2_2.xml*

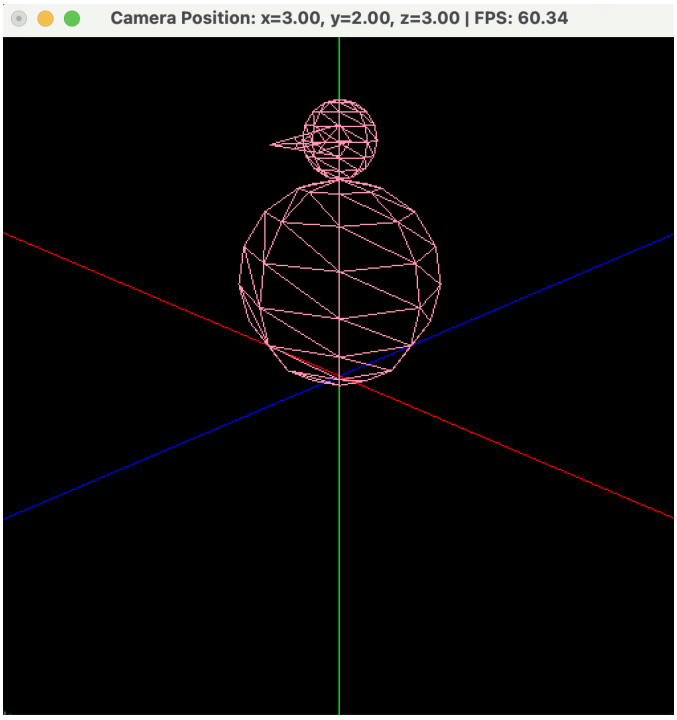

Figura 5.3: *test_2_3.xml*

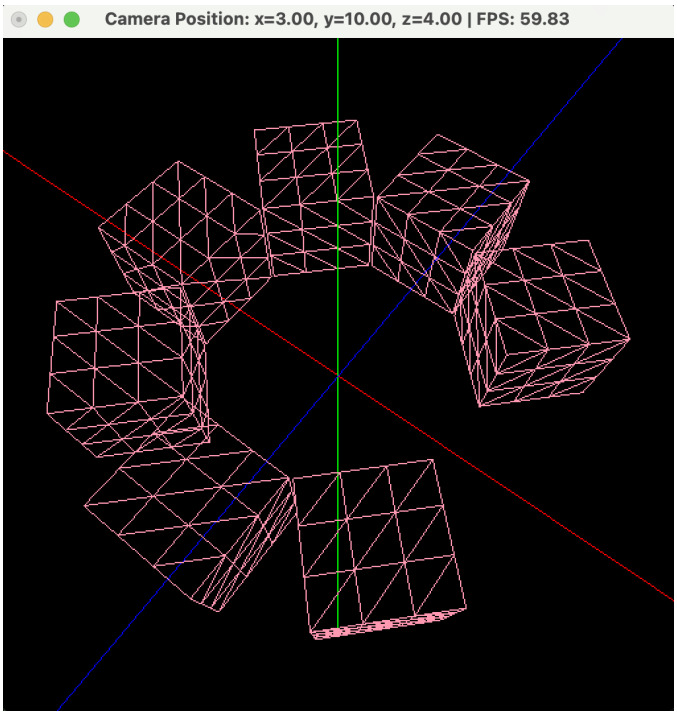

Figura 5.4: *test_2_4.xml*

Capítulo 6

Conclusão

Nesta Segunda Fase do trabalho, foi possível consolidar os conhecimentos práticos e teóricos das aulas de Computação Gráfica. Aprendeu-se a combinar o desenho de primitivas através de uma organização hierárquica e implementação de transformações geométricas. Foi consolidado o conhecimento sobre a herança de transformações geométricas e a importância da ordem com que as transformações são aplicadas.

Os conhecimentos adquiridos nesta fase dão continuidade à fase anterior e são essenciais para as etapas subsequentes, onde irão ser exploradas técnicas mais avançadas de Computação Gráfica.