

Relatório Computação Gráfica - Fase 2

Marco Sousa^{1,2[62608]}, José Malheiro^{1,2[93271]}, and Miguel Fernandes^{1,2[94269]}

¹ Universidade do Minho

² Licenciatura em Engenharia Informática, Braga, Portugal

Resumo Por forma a permitir a construção de um mundo gráfico virtual, é necessário a possibilidade de transformações geométricas dos objetos a serem construídos. Assim, procedeu-se à utilização das funções que permitem transformações geométricas, como `glTranslate`, `glRotate` e `glScale` para construir um modelo do sistema solar a partir da definição hierárquica do sol, planetas e luas.

Procedeu-se, ainda, à utilização de VBO e implementação de uma câmara FPS para navegar pelo mundo.

Keywords: OpenGL · GLUT · Glew · c++ · Transformations · FPS

1 Introdução

1.1 Contextualização

No seguimento da fase I do projeto da disciplina de *Computação Gráfica* da Licenciatura em Engenharia Informática da Universidade do Minho, foi proposto a aplicação de transformações geométricas ao motor previamente desenvolvido, assim como o desenvolvimento de um modelo do sistema solar.

1.2 Breve Descrição do Enunciado Proposto

O principal foco da fase a que este relatório se remete é a aplicação de um conjunto de transformações geométricas, tais como translação, rotação e escala de forma hierárquica a um conjunto de modelos. Assim, a estrutura dos modelos é efetuada através da definição de grupos e respetivos subgrupos, conforme exposto no enunciado.

Os grupos *filhos* deverão adquirir as transformações geométricas dos *pais*, i.e. as transformações são aplicados a todos os modelos e subgrupos. As transformações geométricas devem ser aplicadas na mesma ordem que são definidas no ficheiro de *input*.

Por último, foi proposto o desenvolvimento de um modelo **estático** do sistema solar, incluindo o sol, planetas e luas definidas de forma hierárquica.

2 Trabalho Realizado

Funcionalidades implementadas:

1. Transformações geométricas aplicadas de forma hierárquica (2.1)
 - (a) Possibilidade de aplicar múltiplas vezes a mesma transformação
 - (b) Garantia de aplicação das transformações pela mesma ordem que a definida no ficheiro de *input*
2. Navegação com câmara *FPS* (2.2)
 - (a) Acréscimo de escala de navegação
3. Implementação de VBO's
4. Construção do modelo do sistema solar (2.4)

2.1 Transformações Geométricas

Como consequência da arquitetura definida na fase *I*, a implementação desta fase foi facilitada. Em particular, para aplicar as transformações geométricas de forma hierárquica, optou-se por criar uma nova `class Transform` que se pode encontrar ao nível da `class Group`. Ao implementar desta forma, permitiu que um conjunto de transformações fosse aplicada em todo o *scope* do **Grupo** (seja modelos ou subgrupos). Assim, pode-se encontrar a *Transform* ao nível do objeto Group:

CGDraw Source Code 1.1: Group

```

12 class Group {
13
14 private:
15     Transform _transform;
16     Models _model;
17     vector<Group> _groups;
18     XMLElement* _xml_elem;

```

Nesta classe (*Transform*), a primeira abordagem apenas permitia que se tivesse uma transformação de cada tipo, sendo estas aplicadas pela ordem: translação, rotação e escala. Após uma análise mais detalhada do enunciado e mediante as dificuldades encontradas na construção do modelo do *sistema solar*, verificou-se que seria necessário a *engine* ser capaz de aplicar múltiplas transformações do mesmo tipo, no mesmo conjunto de elementos e que a sua ordem de aplicação poderia ser diferente daquela definida de forma estática. Assim, procedeu-se à alteração da `class Transform` para albergar um `std::vector<transformation>`.

CGDraw Source Code 1.2: Transformation

```

17 struct transformation {
18     point p; // transformation x, y, z

```

```

19     float a; // angle (use in rotate)
20     type t; // type: translate, rotate, scale
21 };

```

Em termos concretos, pode-se encontrar a estratégia utilizada representada no seguinte diagrama de sequência na figura 1. Em suma, consiste em chamar de forma recursiva o método `void draw()` ao longo da hierarquia do Grupo, Modelo e Subgrupos.

2.2 Câmera *FPS*

Optou-se por implementar a câmera do tipo *FPS*. A sua implementação consistiu em utilizar uma biblioteca ("cartesian") desenvolvida pelo grupo para facilitar os cálculos sobre pontos 3D.

Durante o *parsing* do ficheiro *XML*:

1. `set_camera_pos(X, Y, Z)` - define a posição onde a câmera se encontra, a partir do elemento `<position x="X" y="Y" z="Z" />`
2. `set_camera_lookat(X, Y, Z)` - define a direção da câmera, a partir do elemento `<lookAt x="X" y="Y" z="Z" />`
 - (a) Cria-se um vetor entre os pontos *lookat* e *position*
 - (b) Efetua-se a normalização do vetor para que seja unitário
 - (c) Armazena-se em duas estruturas distintas o valor em cartesiano e em polar (este último) permite posteriormente oscilar os ângulos α e β para movimentar a câmera.
3. `set_camera_up(X,Y,Z)` - define a orientação do vetor vertical da câmera a partir do elemento `<up x="X" y="Y" z="Z" />`

Após efetuar o *parsing* do *xml*, tem-se o estado inicial de navegação. A partir deste, foi utilizado os métodos `move_camera(CAMenum t)` e `move_lookat(double alpha, double beta)`.

A movimentação do *lookat* consiste em incrementar o valor do α conforme o movimento do rato ao longo da janela. Quanto ao primeiro - `move_camera(CAMenum)` efetua um movimento para a frente, trás, direita e esquerda. Em qualquer um dos casos, é utilizado o vetor *lookat* multiplicado por um escalar para movimentar a posição da câmera. No caso de movimentar para a esquerda e direita, como a referência é para onde a câmera está a olhar, utilizou-se uma rotação de ± 90 deg e, posteriormente, movimentou-se a câmera. Ao utilizar o vetor *lookAt* garante-se que o movimento é uniforme e congruente com a sua direção.

Nota: Atualmente esta funcionalidade encontra-se com um *bug*. A primeira interação do rato com o ecrã gera uma pequena deslocação do *lookat*. Após esta, encontra-se uma interação fluída do utilizador.

2.3 Vertex Buffer Object (VBO)

Optou-se por implementar, desde já, a utilização de VBO's para melhorar a performance da renderização.

A implementação foi efetuada ao nível da `class Model`, que representa a definição de um modelo do sistema e implicou pequenas alterações, nomeadamente:

- Remoção da variável de instância `_points` que apontava para um `std::vector`
- Acréscimo de um `GLuint _buffer` para apontar para um buffer gerado na memória da placa gráfica

Após estas alterações, tratou-se de alocar um buffer na memória da placa gráfica e popular esse buffer com os pontos a serem desenhados, ficando o índice do buffer disponível como variável de instância. Este procedimento decorre durante o *parsing* do documento XML e respetiva leitura dos pontos do modelo:

CGDraw Source Code 1.3: Gerar e popular VBO

```

46         glGenBuffers(1, &_buffer);
47         glBindBuffer(GL_ARRAY_BUFFER, _buffer);
48         glBufferData(GL_ARRAY_BUFFER, points.size() * sizeof(float),
↪ points.data(), GL_STATIC_DRAW);

```

Depois de ter o *buffer* disponível, a sua renderização ocorre durante a execução do método `_draw`:

CGDraw Source Code 1.4: Renderizar VBO

```

73         glBindBuffer(GL_ARRAY_BUFFER, _buffer);
74         glVertexPointer(3, GL_FLOAT, 0, 0);
75         glDrawArrays(_type, 0, _total_points * 3);

```

Futuramente, na próxima fase, pretende-se a implementação de VBO's com índices, por forma a otimizar a *performance*.

2.4 Modelo do Sistema Solar

Conforme requisitado pela equipa docente, procedeu-se à construção de um modelo do sistema solar utilizando a sintaxe *XML* previamente definida, assim como as figuras geométricas geradas pela aplicação **generator**, desenvolvido na fase I.

Assim, a construção do modelo consistiu na criação, de forma hierárquica, do sol, planetas e algumas luas.

A principal dificuldade prendeu-se com a escolha de uma escala adequada. Atendendo às grandes dimensões de diâmetro dos planetas e ainda maior distância entre estes, foi necessário definir escalas diferentes para ambas as medidas.

A escala do diâmetro e distância do sol foi obtida através da fórmula:

$$\begin{aligned} terra_{diâmetro}^{scale} &= 10u \\ planeta_x^{scale} &= \frac{terra_{diâmetro}^{scale} \cdot planeta_x^{real}}{terra_{diâmetro}^{real}} \\ distancia_x^{scale} &= \frac{x^{scale} \cdot x_{real}^{distance}}{x_{diâmetro}^{real}} \cdot 500^{-1} \end{aligned}$$

Depois de definida a escala, a sua concretização foi relativamente linear. Através da utilização da *sphere* gerada no *generator*, aliado ao poder da translação e *scale*, permitiu que se construísse os vários planetas na posição previamente calculada com a escala definida.

De salientar a utilização da figura geométrica ***torus*** para a representação dos anéis de saturno.

Em suma, foi construído um grupo que corresponde à origem, onde é definido o sol. Posteriormente, há vários subgrupos que identificam cada um dos planetas. Os planetas Terra e Marte têm, ainda, outros subgrupos que permitem a criação das suas luas. Optou-se por não representar as luas de outros planetas, atendendo à sua quantidade. O facto de se ter representado o sistema solar de forma hierárquica, permite que, nas fases que se aproximam seja mais fácil a implementação de animações relacionadas com cada planeta e/ ou a totalidade do sistema solar.

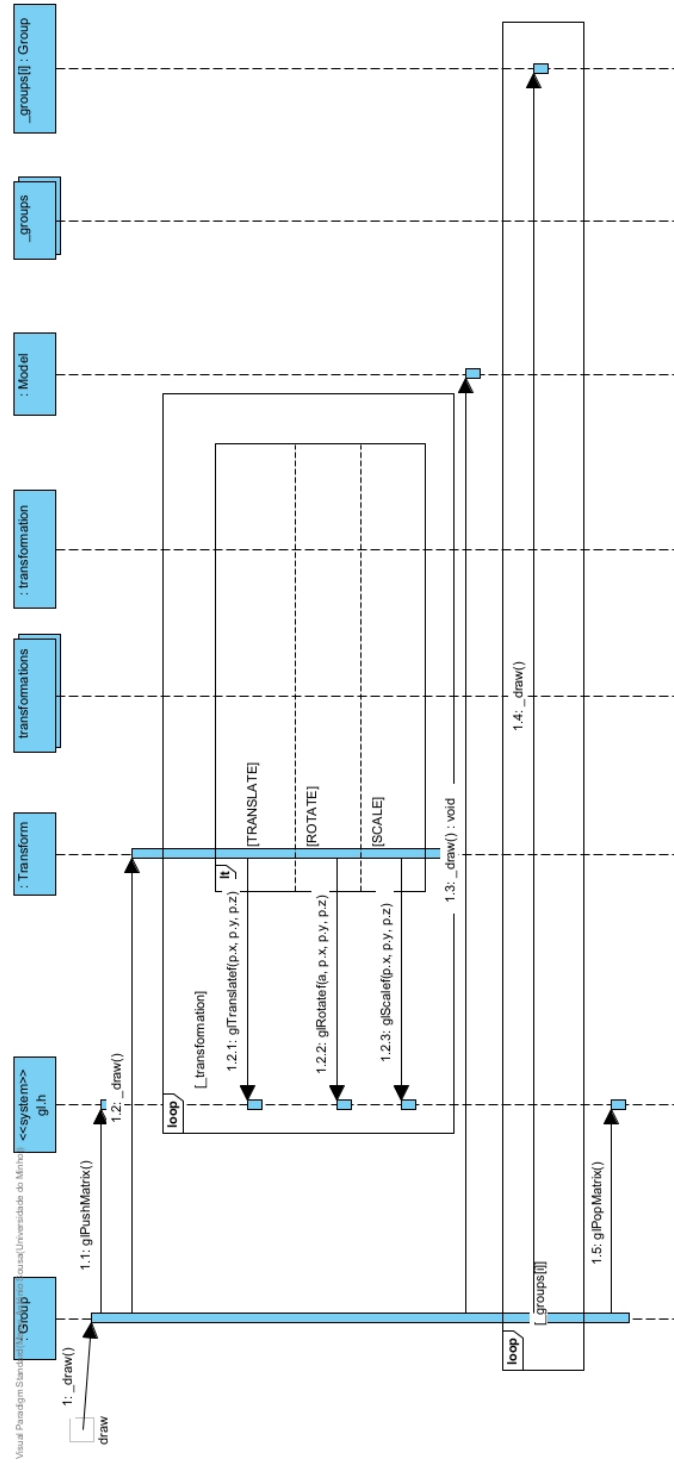


Figura 1: Diagrama de sequência das transformações geométricas

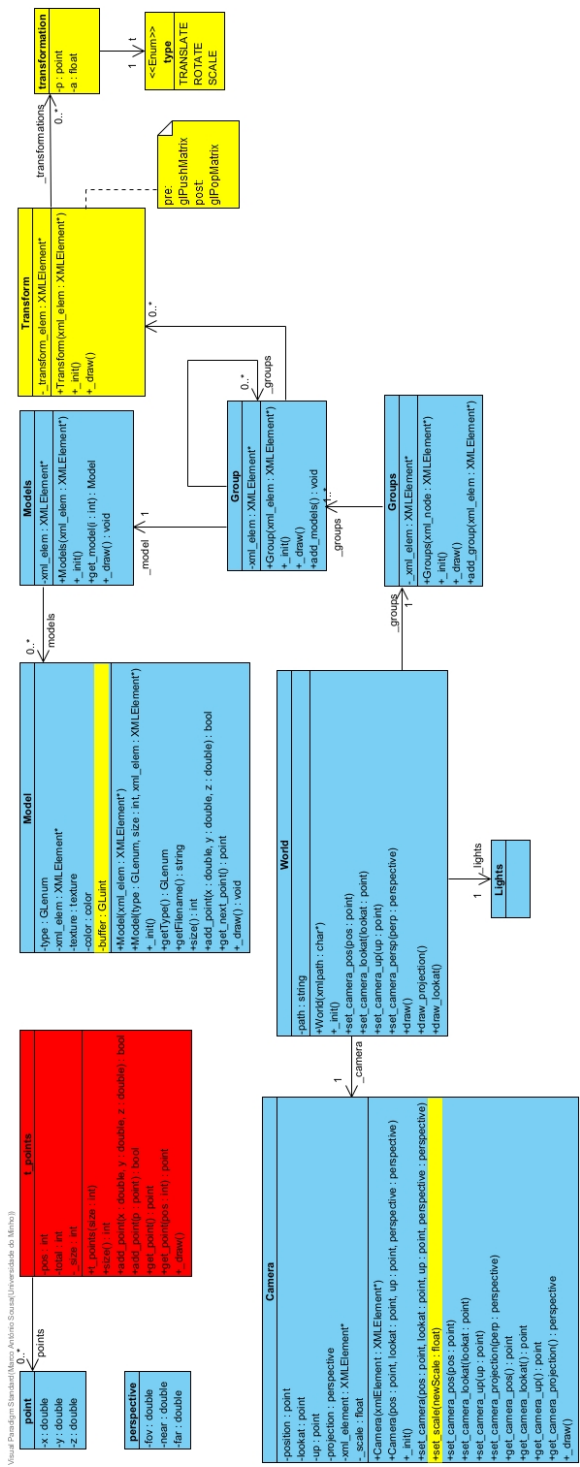


Figura 2: Arquitetura da Aplicação - Alterações em relação à fase 1 a amarelo.

3 Conclusão

Com a conclusão da fase *I*, verificou-se que a implementação da atual fase se tornou bastante expedita pelo facto de se ter desenvolvido uma arquitetura que permite a implementação de novas funcionalidades de forma quase *cirúrgica*, i.e. é possível determinar o exato local onde se terá de efetuar alterações para que se obtenha o resultado pretendido.

Assim, com a presente fase, foi possível implementar todos os requisitos definidos pela equipa docente e, ainda, acrescentar algumas funcionalidades extra. Em particular, nesta fase foi implementado:

- Transformações geométricas
- Geração do torus
- Navegação com câmara FPS
- Utilização de VBO
- Construção do modelo do sistema solar

Assim, pretende-se na próxima fase implementar a utilização de VBOs com recurso a índices, tal como todas as funcionalidade propostas pela equipa docente.

Anexos

Modelo do Sistema Solar

CGDraw Source Code 1.5: Modelo do Sistema Solar

```

1 <world>
2     <!-- a world can have a camera, a set of lights, and a
   ↪ single group -->
3     <camera>
4         <position x="2500" y="400" z="1800" />
5         <lookAt x="0" y="0" z="-1174" />
6         <up x="0" y="1" z="0" /> <!-- optional, use these
   ↪ values as default-->
7         <projection fov="60" near="1" far="10000" /> <!--
   ↪ optional, use these values as default-->
8     </camera>
9
10    <group>
11        <!-- SOL -->
12        <group>
13            <transform>
14                <scale x="1090.1" y="1090.1"
   ↪ z="1090.1" />
15            </transform>
16            <models>

```



```

17         <model file="sphere.3d" /> <!--
           ↳ generator sphere 1 8 8
           ↳ sphere.3d -->
18     </models>
19 </group>
20 <!-- Mercurio -->
21 <group>
22     <transform>
23         <translate x="0" y="0" z="-1180"
           ↳ />
24         <scale x="3.8" y="3.8" z="3.8" />
25     </transform>
26     <models>
27         <model file="sphere.3d" /> <!--
           ↳ generator sphere 1 8 8
           ↳ sphere.3d -->
28     </models>
29 </group>
30 <!-- Venus -->
31 <group>
32     <transform>
33         <translate x="0" y="0" z="-1259.8"
           ↳ />
34         <scale x="9.5" y="9.5" z="9.5" />
35     </transform>
36     <models>
37         <model file="sphere.3d" /> <!--
           ↳ generator sphere 1 8 8
           ↳ sphere.3d -->
38     </models>
39 </group>
40 <!-- Terra -->
41 <group>
42     <transform>
43         <translate x="0" y="0" z="-1324.3"
           ↳ />
44     </transform>
45     <group>
46         <transform>
47             <scale x="10" y="10"
               ↳ z="10" />
48         </transform>
49         <models>
50             <model file="sphere.3d" />
               ↳ <!-- generator sphere
               ↳ 1 8 8 sphere.3d -->

```

```

51         </models>
52     </group>
53     <!-- Lua -->
54     <group>
55         <transform>
56             <translate x="15" y="0"
57                 ↪ z="-15" />
58             <scale x="2.5" y="2.5"
59                 ↪ z="2.5" />
60         </transform>
61     </models>
62     <model file="sphere.3d" />
63 </models>
64 </group>
65 <!-- Marte -->
66 <group>
67     <transform>
68         <translate x="0" y="0" z="-1447.7"
69             ↪ />
70     </transform>
71     <group>
72         <transform>
73             <scale x="5.3" y="5.3"
74                 ↪ z="5.3" />
75         </transform>
76     </models>
77     <model file="sphere.3d" />
78     ↪ <!-- generator sphere
79     ↪ 1 8 8 sphere.3d -->
80 </models>
81 </group>
82 <!-- luas -->
83 <group>
84     <transform>
85         <translate x="5" y="0"
86             ↪ z="8" />
87     </transform>
88     <models>
89         <model file="sphere.3d" />
90         ↪ <!-- generator sphere
91         ↪ 1 8 8 sphere.3d -->
92     </models>
93 </group>
94     <transform>

```

```

87         <rotate angle="90"
           ↪ x="1" y="0"
           ↪ z="1" />
88     <translate x="5"
           ↪ y="0" z="8" />
89     <scale x="0.5"
           ↪ y="1" z="1.3"
           ↪ />
90     </transform>
91     <models>
92         <model
           ↪ file="sphere.3d"
           ↪ />
93     </models>
94 </group>
95 </group>
96 </group>
97 <!-- Jupiter -->
98 <group>
99     <transform>
100         <translate x="0" y="0" z="-2310.4"
           ↪ />
101         <scale x="112.1" y="112.1"
           ↪ z="112.1" />
102     </transform>
103     <models>
104         <model file="sphere.3d" /> <!--
           ↪ generator sphere 1 8 8
           ↪ sphere.3d -->
105     </models>
106 </group>
107 <!-- Saturno -->
108 <group>
109     <transform>
110         <translate x="0" y="0" z="-3337.8"
           ↪ />
111     </transform>
112     <group>
113         <transform>
114             <scale x="94.5" y="94.5"
               ↪ z="94.5" />
115         </transform>
116         <models>
117             <model file="sphere.3d" />
               ↪ <!-- generator sphere
               ↪ 1 8 8 sphere.3d -->

```

```

118         </models>
119     </group>
120     <group>
121         <transform>
122             <rotate angle="-45" x="1"
123                 ↪ y="0.45" z="1" />
124         </transform>
125         <models>
126             <!-- Ring A -->
127             <model file="ring_a.3d" />
128                 ↪ <!--generator torus
129                 ↪ 179 10 15 5 ring_a.3d
130                 ↪ -->
131             <!-- Ring B -->
132             <model file="ring_b.3d" />
133                 ↪ <!-- generator torus
134                 ↪ 199 10 15 5 ring_b.3d
135                 ↪ -->
136         </models>
137     </group>
138 </group>
139 <!-- Urano -->
140 <group>
141     <transform>
142         <translate x="0" y="0" z="-5593.9"
143             ↪ />
144         <scale x="40" y="40" z="40" />
145     </transform>
146     <models>
147         <model file="sphere.3d" /> <!--
148             ↪ generator sphere 1 8 8
149             ↪ sphere.3d -->
150     </models>
151 </group>
152 <!-- Neptuno -->
153 <group>
154     <transform>
155         <translate x="0" y="0" z="-8137.9"
156             ↪ />
157         <scale x="39" y="39" z="39" />
158     </transform>
159     <models>
160         <model file="sphere.3d" /> <!--
161             ↪ generator sphere 1 8 8
162             ↪ sphere.3d -->

```

```
150         </models>
151     </group>
152 </group>
153 </world>
```
