Universidade do Minho

Licenciatura em Engenharia Informática

GitHub do Projeto

Computação Gráfica - Fase 2

Grupo 43

Eduardo Pereira (A94881) Gonçalo Freitas (A96136)
Gonçalo Vale (A96923) José Pereira (A89596)

Ano Letivo 2023/2024




Conteúdo

1 Introdução

Este relatório documentou a fase 2 do trabalho prático da disciplina de Computação Gráfica. O objetivo dessa fase foi aprimorar o projeto da fase anterior, incorporando suporte para transformações de escala, translação e rotação. Adicionalmente, foi implementado suporte para hierarquias de grupos, viabilizando a construção de cenários complexos.

É importante mencionar que o foco deste trabalho concentrou-se no *Engine*, enquanto o *Generator* permaneceu inalterado.

2 *Generator*

Nesta etapa, não houve modificações no *generator*. Todas as alterações foram implementadas diretamente no XML de configuração. Portanto, apenas o *Engine* sofreu mudanças significativas.

3 Estruturas de Dados

Foi criada a estrutura *"Transforms"* para guardar os dados das transformações. Estas estruturas são distinguidas entre si através de uma string *type*, e os valores das transformações são guardados no vetor de *floats "arguments"*.

A estrutura de dados *Figura* agora tem um vetor de *Transforms* por cada grupo, que contém os *Transforms* do grupo e dos grupos "pais", de forma a que estas fiquem posicionadas corretamente em relação aos mesmos.

Outras alterações feitas à estrutura de dados *Figura* passam por mudar a forma como os pontos dos modelos são guardados, sendo agora guardados separadamente, por grupo, na própria estrutura *Figura*.

4 *Engine*

Durante esta etapa, o *Engine* experienciou modificações substanciais, habilitando a realização de transformações em conjuntos de objetos e a composição hierárquica de grupos dentro de outros grupos. Assim, facilita-se a aplicação recursiva de transformações em um grupo e seus elementos subordinados.

4.1 Transformaçoes

Nesta etapa, foram introduzidas novas operações aplicáveis a grupos específicos. O fragmento de código a seguir demonstra um exemplo em que um grupo é configurado para aplicar uma rotação de 30 graus a todos os seus elementos. Este grupo é composto pelo *OBJ1*, representando uma esfera. O *OBJ1* é posicionado na origem (0,0,0) e mantém uma escala unitária (1) em relação ao

modelo original. Por outro lado, o *OBJ2* é transladado para as coordenadas (41,5, 0, 0) e ajustado para 0,035

```
<group>
    <transform>
        <rotate angle="30"x="0"y="1"z="0"/>
    </transform>
    <!-- OBJ1 -->
    <group>
        <transform>
            <translate x="0"y="0"z="0"/>
            <scale x="1.0"y="1.0"z="1.0"/>
        </transform>
        <models>
            <model file="../3d_files/sphere.3d" />
        </models>
    </group>

    <!-- OBJ2 -->
    <group>
        <transform>
            <translate x="41.60"y="0"z="0"/>
            <scale x="0.003505"y="0.003505"z="0.003505"/>
        </transform>
        <models>
            <model file="../3d_files/sphere.3d" />
        </models>
    </group>
</group>
```

4.1.1 Translação

Dentro de um grupo, é possível realizar uma operação de translação, a qual desloca os objetos do grupo (incluindo seus descendentes) para as coordenadas especificadas no XML da operação. Essencialmente, esta operação modifica as coordenadas centrais do grupo.

A execução desta função ocorre durante a análise sintática (*parsing*) dos objetos, conforme ilustrado abaixo:

```
glTranslatef(x, y, z);
```

Esta abordagem assegura que a translação seja aplicada diretamente aos parâmetros definidos, influenciando a posição dos objetos no espaço tridimensional.

4.1.2 Rotação

Dentro de um grupo, pode-se efetuar uma operação de rotação, que faz girar o conjunto inteiro do grupo em torno de um eixo definido pelas coordenadas (x, y, z).

```
glRotatef(angle, x, y, z);
```

É crucial destacar que esta rotação é aplicada de maneira uniforme a todos os elementos presentes no grupo, abrangendo também quaisquer subgrupos herdeiros, garantindo assim a coesão na transformação aplicada.

4.1.3 Escala

Dentro de um grupo, é possível realizar operações de escala, incluindo ampliação e redução. Essa transformação afeta todos os componentes do grupo, abrangendo tanto os objetos individuais quanto os subgrupos herdeiros.

A operação de escala permite a modificação proporcional das dimensões dos objetos em diferentes eixos, possibilitando, assim, ajustes na esticagem e nas proporções dos objetos de acordo com as coordenadas especificadas.

```
glScalef(x, y, z);
```

Esse procedimento facilita a personalização e ajuste das dimensões dos objetos, conferindo flexibilidade significativa na modelagem e apresentação dos elementos dentro do grupo.

4.2 Grupos aninhados

Como mencionado anteriormente, é possível estabelecer grupos dentro de outros grupos, criando assim uma dependência hierárquica e recursiva. Essa funcionalidade permite a composição de estruturas complexas de grupos, sobre as quais as transformações previamente discutidas podem ser aplicadas de maneira coesa.

A função abaixo é projetada para implementar essa característica de forma eficiente:

```
void drawFigures(vector<Figure> models) {
  for(Figure f : models){
    drawFigure(f.getPontos(), f.getTransforms());
    // No caso de um grupo ter outro grupo, excecutar
    if(!f.getFiguras().empty()){
      drawFigures(f.getFiguras());
    }
  }
}
```

Como ilustrado, a função drawFigures inicia percorrendo e aplicando todas as transformações aos modelos. Em seguida, verifica a presença de subgrupos dentro de cada figura. Caso existam, a função drawFigures é chamada recursivamente para estes subgrupos. Essa abordagem facilita a criação de uma arquitetura recursiva, possibilitando a formação de um número arbitrário de níveis de agrupamento de figuras.

4.3 Desenho da cena

Em seguida encontra-se uma parte do nosso código encarregue de desenhar os modelos, aplicando as transformações correspondentes. Para garantir que as transformações são aplicadas apenas à figura correspondente, é necessário utilizar *glPushMatrix e glPopMatrix* individualmente para cada modelo.

```
void drawFigure(vector<Ponto> points, vector<Transform> transforms){
    if(!points.empty()){
        if(transforms.empty()){
            glBegin(GL_TRIANGLES);
                glColor3f(1,1,1);
                for(Ponto p : points){
                    glVertex3f(p.x, p.y, p.z);
                }
            glEnd();
        }
        else{
            glPushMatrix();
            for(Transform t : transforms){
                if(strcmp(t.type.c_str(), "translate") == 0){
                    glTranslatef(t.arguments[0], t.arguments[1],
                    t.arguments[2]);
                }
                else if(strcmp(t.type.c_str(), "rotate") == 0){
                    glRotatef(t.arguments[0], t.arguments[1],
                    t.arguments[2], t.arguments[3]);
                }
                else if(strcmp(t.type.c_str(), "scale") == 0){
                    glScalef(t.arguments[0], t.arguments[1],
                    t.arguments[2]);
                }
            }

            glBegin(GL_TRIANGLES);
                glColor3f(1,1,1);
                for(Ponto p : points){
                    glVertex3f(p.x, p.y, p.z);
                }
            glEnd();

            glPopMatrix();
        }
    }
}
```

5 Sistema solar e resultados

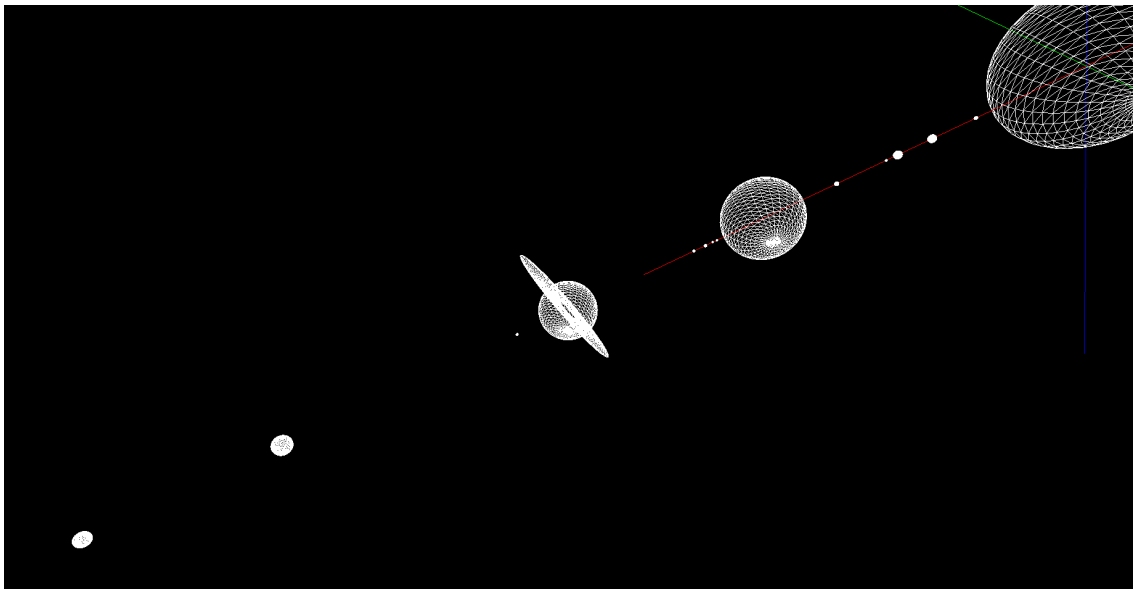

Figura 1: Representação so Sistema Solar através do programa criado

Como é possível ver na figura acima, o nosso programa é capaz de criar um esquema do sistema solar no qual são utilizadas e por isso demonstradas as transformações de modelos requeridas nesta fase do projeto.

Todos os exemplos providenciados pelo professor foram também testados e comprovados que funcionam corretamente.

6 Conclusão

Resumindo, tendo chegado a esta etapa do trabalho prático, nós demonstramos não só a retenção das capacidades desenvolvidas durante a primeira fase do projeto mas também a aquisição de novo conhecimento lecionada nas aulas práticas e teóricas entretanto.

Apesar de nos ter-mos deparado com certas complicações no que toca à leitura dos ficheiros .xml, conseguimos resolvê-las e crescer a nossa base de conhecimentos e capacidades em relação ao tópico, as quais serão indispensáveis para não só o resto desta cadeira, mas para toda a nossa carreira como engenheiros informáticos.