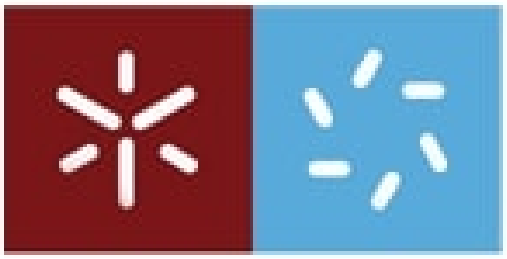

Universidade do Minho
Escola de Ciências

Computação Gráfica

Jorge Daniel Pereira Silva (a80931)
José Pedro Silva Ferreira (a96798)
Rui Alexandre Dias Neto (a80433)

2 de abril de 2024

Conteúdo

1 Introdução

O objetivo proposto nesta segunda fase fase do trabalho prático é criar cenas hierárquicas usando transformações geométricas, para isso só tivemos que alterar o **Engine**.

2 Engine

Nesta aplicação tivemos que criar uma nova classe e alterar a classe group e funções na main.

2.1 Classes

2.1.1 Transformation

Nesta classe temos 3 vetores para guardar as respetivas transformações(rotate,translate ou scale).Temos os construtores e os métodos *get* e *set*. Criamos também o método para dar parse ao XML, no qual verificamos se é um rotate , um translate ou um scale.Nos métodos *set* acrescentamos uma string transformation para identificarmos qual a transformação que estamos a tratar.

2.1.2 Group

Nesta Classe acrescentamos um vector de transformações,acrescentando os métodos *get* e *set*, e um vector de Group para identificarmos subgrupos.

2.2 Main

Neste ficheiro reorganizamos a função *DrawModels*,onde é passado como argumento um group, epercorremos as transformações os modelos e os subgrupos(Figura 1). Passamos parte do código que tínhamos nesta função para uma chamada *DrawModel*(Figura 2).

```
void drawModels(Group group) {
    glPushMatrix();

  for(Transformation transform : group.get_transforms()){
        string transformation = transform.get_transformation();

        if (transformation == "translate"){
            vector<float> translate = transform.get_translate();
            glTranslatef(translate[0],translate[1],translate[2]);
        } else if (transformation == "scale"){
            vector<float> scale = transform.get_scale();
            glScalef(scale[0],scale[1],scale[2]);
        } else if (transformation == "rotate"){
            vector<float> rotate = transform.get_rotate();
            glRotatef(rotate[0],rotate[1],rotate[2],rotate[3]);
        }
    }

    for (Model model : group.get_models()){
        drawModel(model);
  }

    for(Group subgroup : group.get_subgroups()){
        drawModels(subgroup);
    }

    glPopMatrix();
}
```

Figura 1: DrawModels

```
void drawModel(Model model){
    for (vector<int> triangleIndices : model.get_index()) {
    int index1 = triangleIndices[0];
    int index2 = triangleIndices[1];
    int index3 = triangleIndices[2];

    vector<float> point1 = model.get_points()[index1];
    vector<float> point2 = model.get_points()[index2];
    vector<float> point3 = model.get_points()[index3];

    glBegin(GL_TRIANGLES);
    //x y z of each point
    glVertex3f(point1[0], point1[1], point1[2]);
    glVertex3f(point2[0], point2[1], point2[2]);
    glVertex3f(point3[0], point3[1], point3[2]);
    glEnd();
  }
}
```

Figura 2: DrawModel

2.3 Demos

Decidimos criar uma pasta chamada demos onde temos o XML do sistema solar. Neste XML para desenhar os planetas e as luas usamos uma esfera de testes(Figura 3).

```
<group>
    <transform>
        <scale x="0.5" y="0.5" z="0.5" />
    </transform>
    <group> <!-- sol-->
        <transform>
                <scale x="202.02" y="202.02" z="202.02" />
        </transform>
        <models>
            <model file="sphere_1_10_10.3d" />
        </models>
    </group>
    <group> <!-- mercurio-->
        <transform>
            <translate x="350" y="0" z="0" />
            <scale x="0.71" y="0.71" z="0.71" />
        </transform>
        <models>
            <model file="sphere_1_10_10.3d" />
        </models>
    </group>
    <group> <!-- venus-->
        <transform>
            <translate x="410" y="0" z="0" />
            <scale x="1.68" y="1.68" z="1.68" />
        </transform>
        <models>
```

Figura 3: SystemXML

2.4 Resultados

Demonstramos aqui os ficheiros XML lidos e o seu resultado:

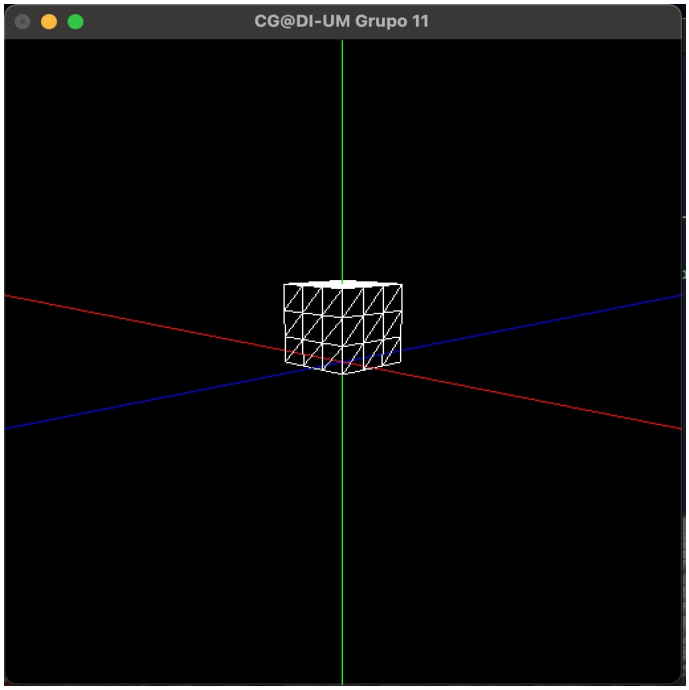

Figura 4: Teste1

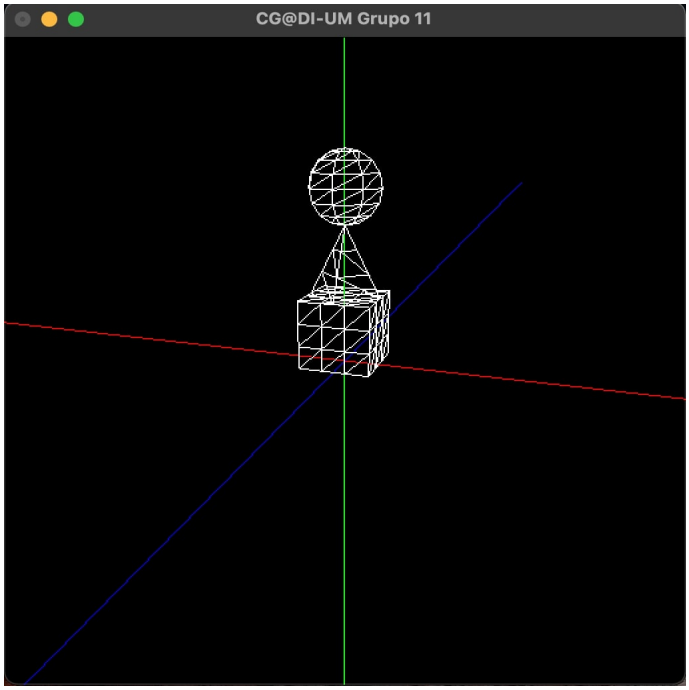

Figura 5: Teste2

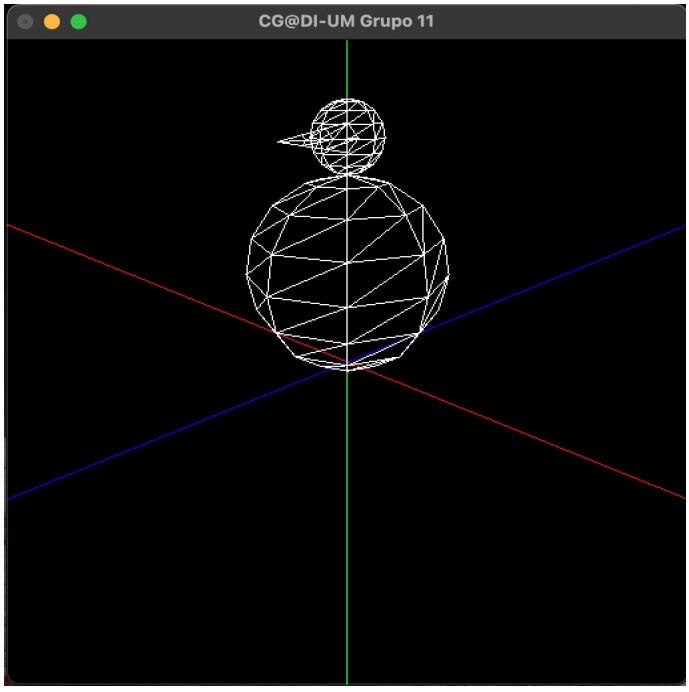

Figura 6: Teste3

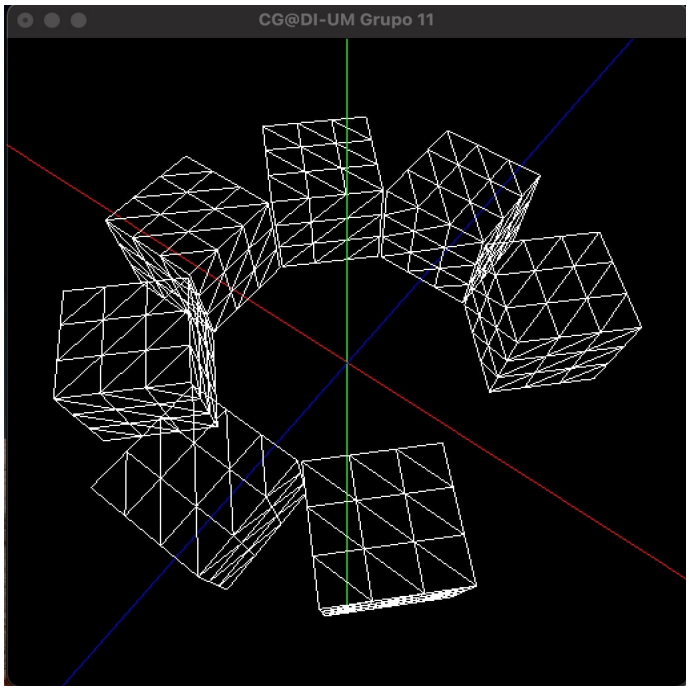

Figura 7: Teste4

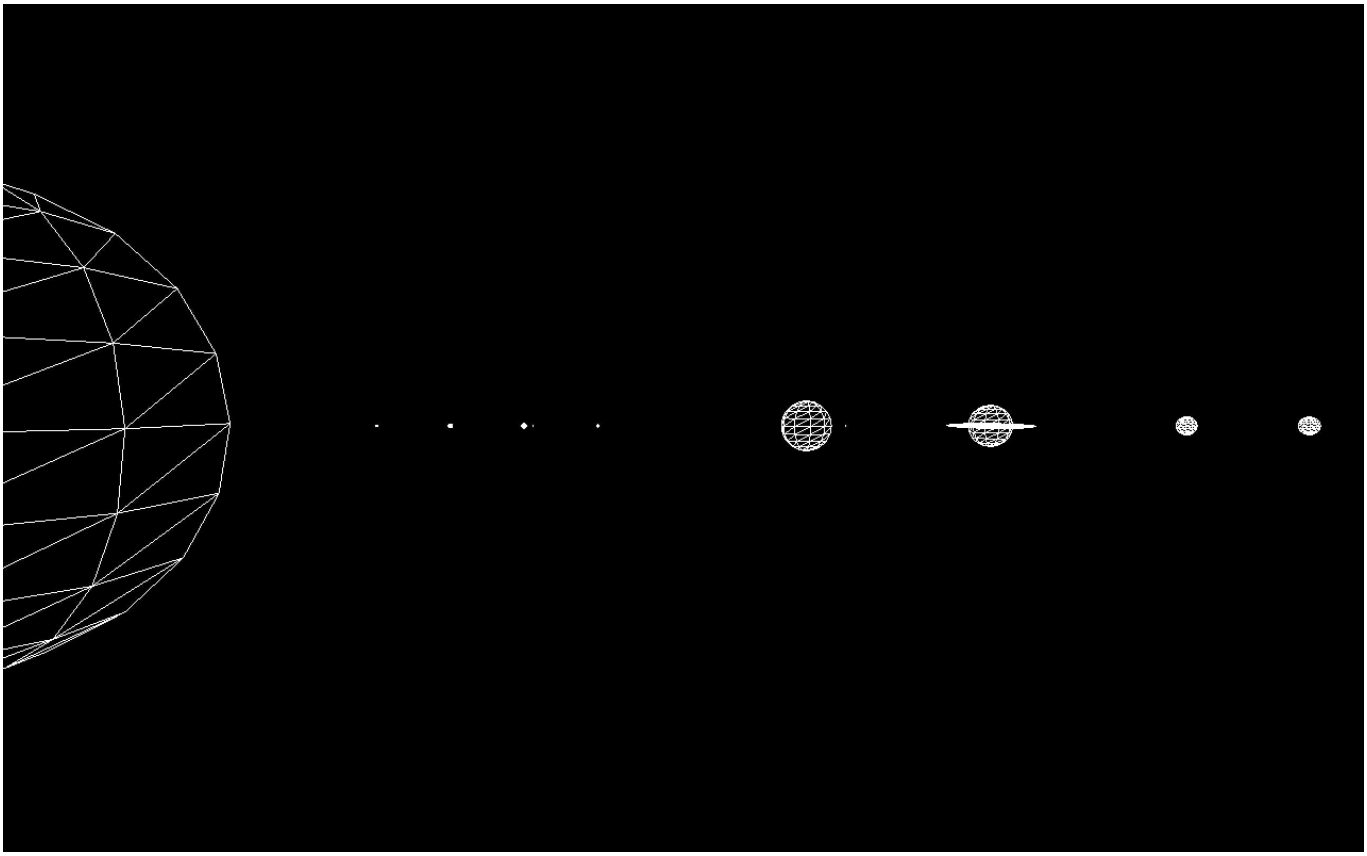

Figura 8: Sistema Solar

3 Conclusão

Após a realização da segunda fase,conseguimos obter uma melhor compreensão sobre como criar cenários complexos através de múltiplas transformações geométricas. O mais difícil nesta fase foi os pontos no XML do sistema solar onde sentimos muitas dificuldades. Concluindo, nesta fase conseguimos cumprir os requisitos necessários para completar a mesma.