Universidade do Minho

Licenciatura em Engenharia Informática

Computação Gráfica

Grupo 30

Trabalho Prático - **Fase 2**

Transformações Geométricas

Joana Alves (A93290)

Maria Cunha (A93264)

Vicente Moreira (A93296)

Abril 2023

Conteúdo

1 Introdução

Este relatório foi desenvolvido no âmbito da segunda fase de do trabalho da unidade curricular de Computação Gráfica da Licenciatura em Engenharia Informática, tendo como objetivo a evolução do *engine* criado, a partir da integração e suporte de transformações geométricas a serem aplicadas sobre as primitivas geométricas desejadas.

Este documento detalha as várias modificações efetuadas para alcançar o objetivo, assim como as decisões tomadas pelo grupo. Por fim, foram efetuados vários testes para verificar o bom funcionamento destas transformações.

2 Modificações para o Suporte de Transformações

2.1 Modificações à Estrutura de Dados

O primeiro passo que o grupo tomou para a adição do suporte de transformações geométricas foi a adição de novas variáveis que armazenassem a informação relativa aos três tipos de transformações geométricas. Estas variáveis como representam as transformações a serem aplicadas a um determinado grupo, foi decidido que seriam adicionadas à classe **grupo.h** já existente, tendo as variáveis os seguintes nomes e valores por defeito:

- **Translação -** *(float)* transX, transY, transZ (Valor por defeito 0)
- **Rotação -** *(float)* rotAngle, rotX, rotY, rotZ (Valor por defeito 0)
- **Escala -** *(float)* scaleX, scaleY, scaleZ (Valor por defeito 1)

Outra variável adicionada foi a **transformOrder**, constituída por um array de 3 caracteres, que irão representar a ordem de aplicação das transformações. Este array é inicializado com os valores **{'x', 'x', 'x'}**, sendo que o caracter 'x' representa nenhuma transformação a ser aplicada. Para os três tipos de transformações, os caracteres **'t'**, **'r'** e **'s'** são usados, respetivamente, para indicar uma translação, rotação e escala. Assim, dando um exemplo, um array constituído por **{'r', 't', 'x'}** indica que a ordem das transformações do grupo será: rotação → translação.

2.2 Leitura das Transformações

Para a leitura das transformações presentes no ficheiro *XML*, foi adicionada uma nova função (***loadTransforms***) à lógica de inicialização do grupo. Esta função percorre por ordem de aparição todos os *child nodes* presentes no nodo ***transform***, analisando o nome dos nodos que percorre para verificar se este corresponde a alguma das transformações geométricas possíveis. Caso seja encontrado um nodo relativo a uma transformação geométrica, são lidos e atualizados os atributos respetivos nos dados do grupo e registada a ordem de aparição deste nodo.

2.3 Aplicação das Transformações na Cena

Para a aplicação das transformações, foi apenas necessário alterar a lógica de *render* do grupo, mais em específico, a função ***drawGroup***. Anteriormente o grupo seguia uma lógica simples, começando por desenhar todos os modelos no seu grupo e, por fim, desenhar todos os seus subgrupos.

Com a adição de transformações geométricas nos grupos e, possivelmente, nos vários subgrupos, foi necessária a adição de novos comandos ***glPushMatrix()*** e ***glPopMatrix()*** no início e no fim da função, respetivamente, de forma a preservar a matriz de modelo aquando do desenho dos subgrupos. Também foi adicionada antes do desenho dos modelos e subgrupos a lógica de leitura do *array* **transformOrder**, e a posterior aplicação das transformações lidas através dos comandos de **OpenGL** correspondentes.

Apresentamos, de seguida, o novo código da função **drawGroup**, de forma a demonstrar as modificações referidas e mostrar que, apesar desta ser a função vital de *render* da cena do *engine*, esta tem um implementação relativamente simples.

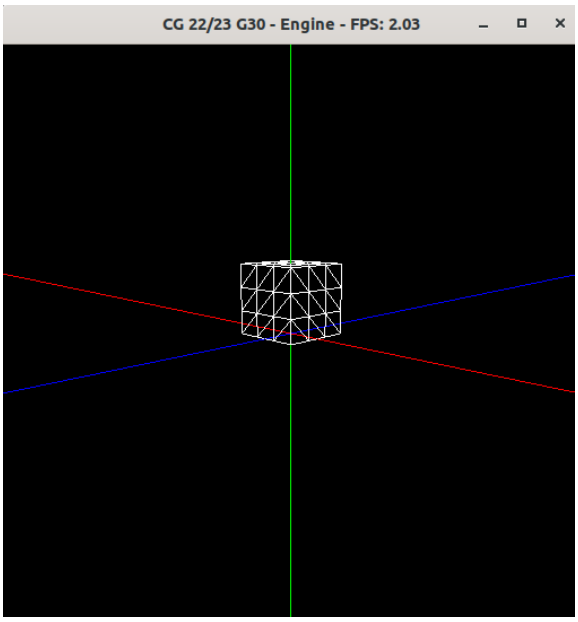
```
void group::drawGroup() {
    glPushMatrix();

    //Aplicar Transformações por Ordem
    for(int i = 0; i < 3; i++){
        if (this->transformOrder[i] == 't')
            glTranslatef(this->transX,this->transY,this->transZ);
        else if (this->transformOrder[i] == 'r')
            glRotatef(this->rotAngle,this->rotX,this->rotY,this->rotZ);
        else if (this->transformOrder[i] == 's')
            glScalef(this->scaleX,this->scaleY,this->scaleZ);
        else break;
    }

    //Desenhar Modelos
    for (auto it = models.begin(); it != models.end(); ++it){
        it->drawModel();
    }

    //Desenhar SubGrupos
    for (auto it = subgroups.begin(); it != subgroups.end(); ++it){
        it->drawGroup();
    }

    glPopMatrix();
}
```

3 Testes e Resultados

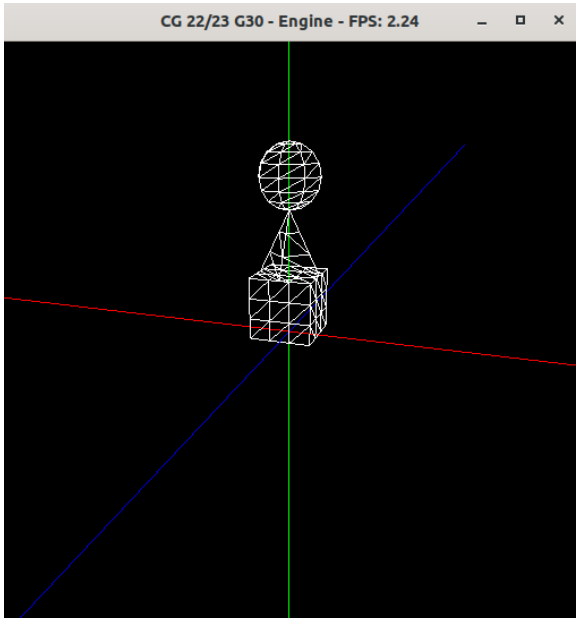
Nesta secção vamos apresentar os diversos testes executados pelo grupo de modo a conseguir avaliar o funcionamento do *engine* quanto à sua correção e exatidão de resultados.

3.1 Testes Docentes

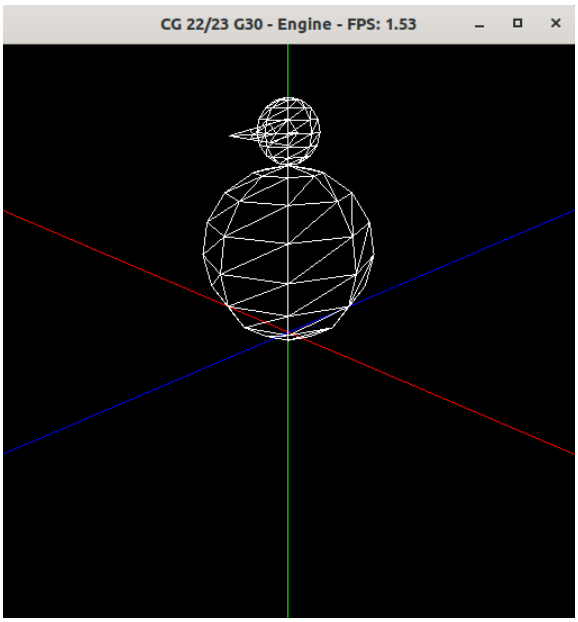
Alguns dos testes implementados foram cedidos pelos docentes de modo a verificar o comportamento correto do *engine*. Assim, apresentamos os resultados obtidos em cada um dos testes, sendo de notar que todos estes foram de encontro ao esperado.



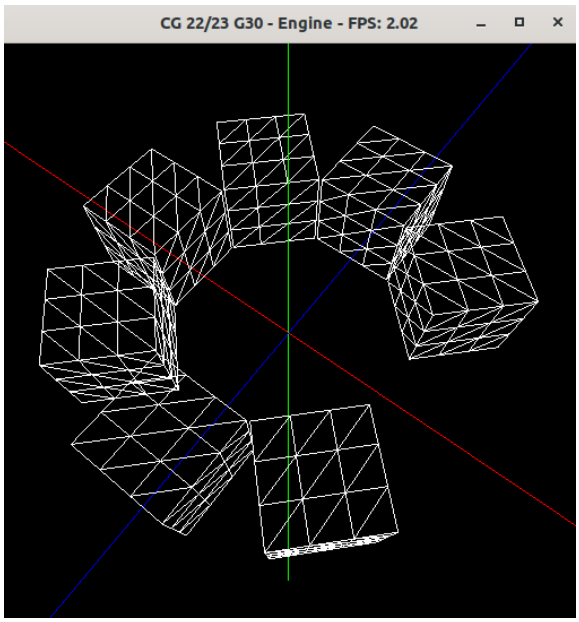
test_2_1



test_2_2



test_2_3



test_2_4

3.2 Testes Grupo

De seguida, o grupo criou alguns testes personalizados para testar o *engine* de forma mais profunda. Como requerido pelos docentes, um dos testes deverá simular o Sistema Solar dentro do **engine**, onde o Sol, os planetas e as luas são definidas segundo as hierarquias do ficheiro de **XML**.

Para o desenvolvimento da *demo*, visto que para representar um Sistema Solar à escala traz imensos desafios na sua visualização, foram feitas alterações nesta escala, sendo o objetivo principal da *demo* representar o sistema solar numa só janela, assim como possuir alguns detalhes de tamanho e distância entre planetas. Para a seleção das luas a representar, decidimos representar todas as luas de maior dimensão presentes na lista de luas do Sistema Solar por tamanho, até à lua *Triton*, ou seja, na *demo* estão representadas as luas *Ganymede*, *Titan*, *Callisto*, *Io*, 'Lua', *Europa* e *Triton*.

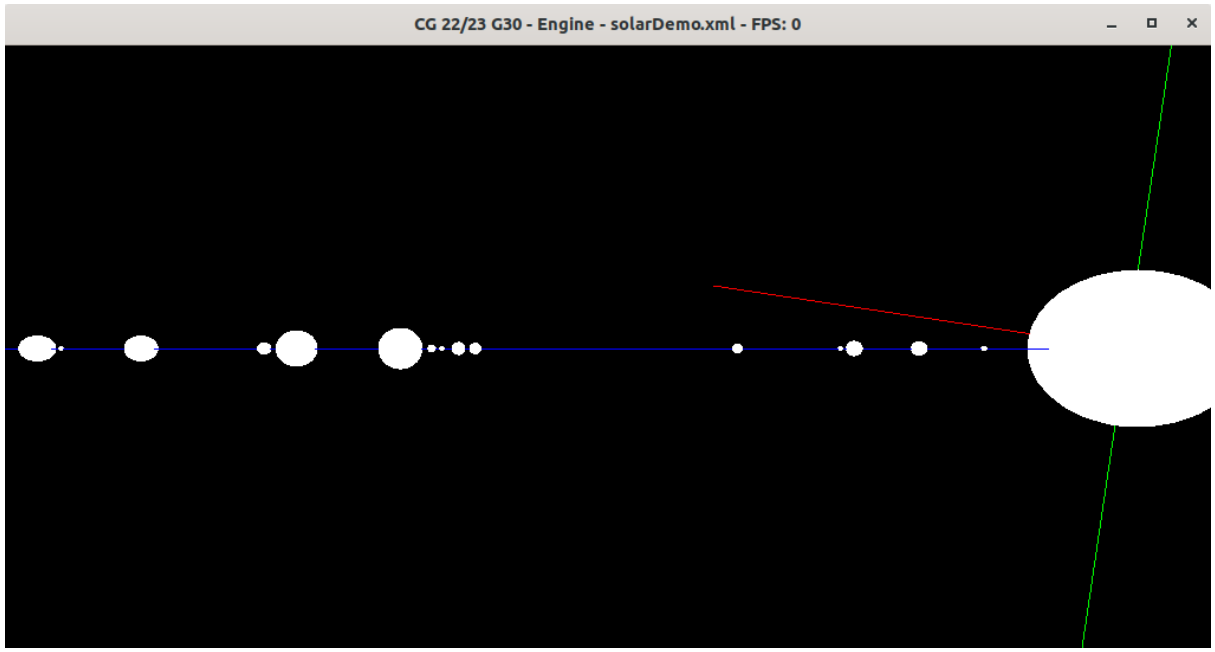


Figura 1: *Demo* Sistema Solar

4 Conclusão

Com a realização desta segunda fase do projeto, o grupo encontra-se satisfeito com o trabalho desenvolvido, tendo sido alcançados todos os objetivos estabelecidos quer pelos docentes, como pelo próprio grupo.

Em suma, aprofundamos o conhecimento adicional lecionado nas aulas quer teóricas quer práticas após a primeira fase do projeto, que nos ajudaram a perceber e a implementar técnicas de *render* e cálculo de matrizes como, por exemplo, aplicar *push* e *pop* da matriz do modelo no início e no final de cada iteração de *render*, respetivamente.