Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de
Computação Gráfica

Ano Letivo de 2021/2022

Relatório Fase 2
Transformações Geométricas

Grupo

Pedro Araújo	a90614
Pedro Fernandes	a84313
João Cardoso	a94595
Rita Gomes	a87960

4 de Abril de 2022

Índice

1 Introdução

O presente relatório serve de suporte ao trabalho realizado no âmbito da unidade curricular de Computação Gráfica. Este trabalho está dividido em quatro fases, sendo que neste relatório será analisada toda a formalização e modelação envolvida na realização da segunda fase.

O objetivo desta fase é o seguinte: a partir do engine elaborado na fase anterior, realizar um tratamento das transformações geométricas (*rotação, translação e escalamento*) lidas a partir de um ficheiro em formato .xml, para que posteriormente seja desenhado um modelo estático idêntico ao sistema solar, incluindo o sol, planetas e respetivas luas.

Para permitir a correta renderização dos novos modelos é necessário efetuar alterações no engine, de maneira a que este seja capaz de processar o novo formato do ficheiro .xml, assimo como a estrutura de dados necessária para armazenar em memória as informações cruciais para a renderização das cenas.

2 Formalização do problema

Como previamente mencionado, nesta fase será necessário gravar não só as primitivas lidas a partir de um ficheiro .xml como também o conjunto de operações associadas às mesmas.

Existem campos denomidados por "group"que agregam estas informações, sendo que há a possibilidade de haver outros "groups"associados ao mesmo, dos quais herdam as informações do parente. Deste modo, iremos precisar de implementar estruturas de dados capazes de sustentar todo o conjunto de dados.

Relativamente ao parsing do ficheiro .xml, o mesmo terá de ser alterado de modo a que possa processar ordenamente e corretamente todos os parâmetros presentes, pois por se tratar de transformações geométricas é preciso ter cuidado com a ordem que as operações são feitas para não prejudicar o resultado final pretendido. Este processo é deveras importante para que, posteriormente, nos permita desenhar todas as primitivas de acordo com os parâmetros requeridos no ficheiro.

Assim sendo, todas as novas operações passarão por um processo de "conversão"para funções glut que possibilitem o desenho das primitivas.

3 Estruturas de dados utilizadas

Como foi visto anteriormente, as informação a ser lida acerca das transformações presente no campo *group* terá que ser processada de forma hierárquica, de forma a que um nodo pai possa possuir inúmeros nodos filhos. Note que o pai contém informações relevantes que são associadas a todos os filhos.

Para concretizar isto foi criado a estrutura "**transformation**"que possui toda a informação associada às transformações geométricas das figuras e o desenho das mesmas.Esta *struct* baseia-se num vetor de listas ligadas, correspondente à mesma struct, que constitui a função de armazenar as informações dos nodos filhos.

A *struct* possui também um vetor de todas as operações relativas ao nodo, incluindo as transformações geométricas e as informações relevantes para o desenho da figura. Por último, contém ainda uma variável booleana que possui a informação relativa à existência dos nodos filhos relativos.

4 Arquitetura da Solução

4.1 *Transformações Geométricas*

No capítulo 3 foi exposta a estrutura necessária para guardar as novas informações presentes nesta fase do projeto. A principal modificação foi relativa à inclusão das transformações geométricas.

Como já foi visto, as operações de cada nodo "group"lido serão guardadas num vetor. Iremos agora explicitar com maior detalhe como foram desenvolvidas estas operações.

Os requisitos desta fase envolvem a obrigatoriedade de existirem 3 tipos de transformações geométricas: *Translação, Escala e Rotate.* Foi criada a classe abstrata *operations*, e também as respectivas classes associadas: *opRotate*, *opTranslate* e *opScale* para permitir armazenamento destas operações.

Como funções virtuais associadas à classe abstrata, de seguida foram criadas as funções *run* e *print*.

A função *run* possui a finalidade de executar as operações/transformações da sua respetiva classe implementadora da função. A função *print* foi criada com o intuito de fazer *debug*, imprimindo como output os valores guardados na classe.

De notar que também foi associada à classe *operations* a classe *opDraw*, a qual possui toda a informação relativa ao desenho das figuras, deste modo é generalizada qualquer transformação/operação realizada no programa.

4.2 *Processamento dos ficheiros XML*

Para ler o ficheiro .xml e consequente processamento do mesmo foi necessário modificar o engine realizado na primeira fase deste trabalho. Para isso, o grupo optou por criar uma nova função *readGroups*, que está responsável por ler toda informação relativa ao campo *group* do ficheiro .xml.

De uma forma resumida, esta função percorre cada elemento filho do campo *group* raiz (1º grupo encontrado) e analisa cada valor associado a este.

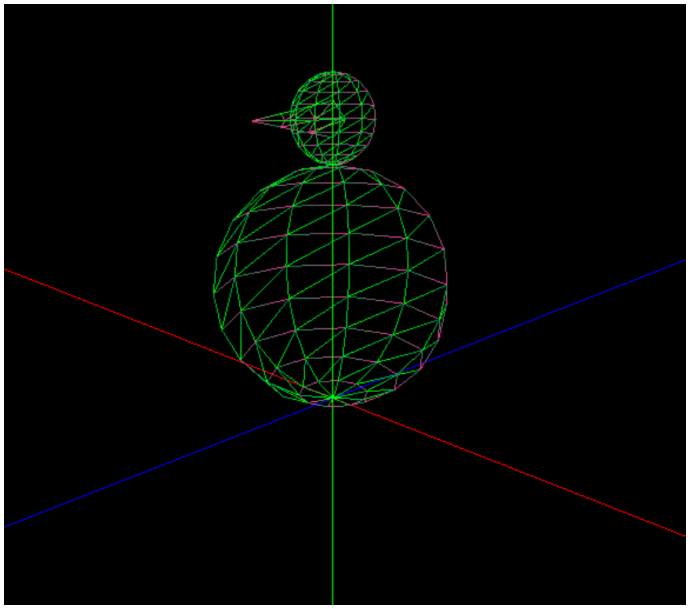
Quando o valor é do tipo *"translate", "rotate"* ou *"scale"* consideramos que se trata

de uma transformação e armazenamos os valores na estrutura responsável por guardar as operações. Caso o valor do elemento em questão seja *models* vamos analisar cada ficheiro .3d associado as figuras, guardar os pontos/vértices e, por fim, armazenamos as informações na mesma *struct* das operações.

Quando o elemento filho de um *group* for do tipo *group* chamamos recursivamente a função *readGroups*, sendo que a cada iteração é criado uma nova *struct* para o nodo filho. No final, é acrescentado ao vector de transformações do nodo pai com todos os "subgroups"associado ao memso.

Para a execução das transformações e desenho das figuras, foi criada a função *drawFigures*. Esta é uma função que é iterada recursivamente, de modo a que a cada iteração é executada a função *glPushMatrix* e depois são realizadas as operações relativas ao nodo. Se neste existirem nodos filhos, é realizada uma nova iteração com cada filho e, caso seja a última linhagem de *group* é finalizado com a função *glPopMatrix*. Resumindo, o objetivo desta função é garantir que as transformações sejam realizadas de forma ordenada, que sejam herdadas as informações para cada filho e, após os nodos mais extremos realizarem as suas operações, ser retirado com *glPopMatrix* de forma a que não tenha impacto nos nodos irmãos.

A imagem abaixo exemplifica o output gerado a partir do ficheiro xml teste 3.



4.3 *Sistema Solar*

Para fins de demonstração das funcionalidades do programa implementado foi solicitado o desenho de um sistema solar. Para a sua elaboração foi necessário escrever o ficheiro sistemaSolar.xml , que possui todas as transformações e ficheiros .3d necessários.

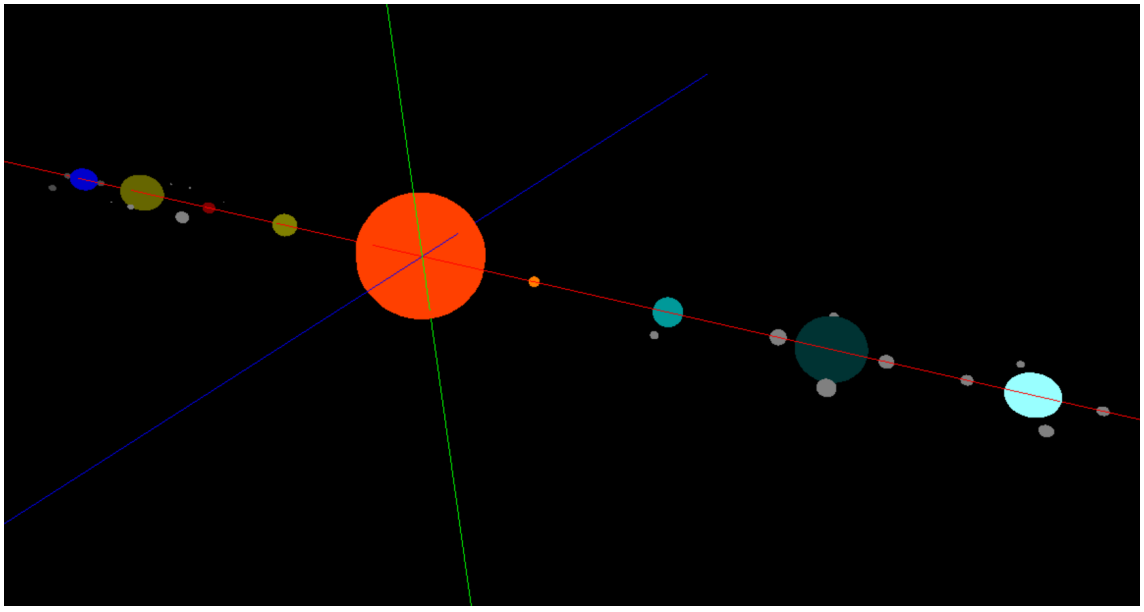
Com o intuito de suplementar trabalho proposto acrescentamos um parâmetro nos ficheiros xml que corresponde à adição de cores juntamente com o desenho da primitiva em questão, para que estas possam ser distinguidas pelas respetivas cores.

Devido ao facto de existirem centenas de luas no sistema solar, o grupo optou apenas por representar algumas delas, visto que o raciocínio é o mesmo para as restantes.

Desta forma, para a criação do ficheiro .xml foi necessário criar vários *groups* correspondente ao sol e aos oito planetas. Para representar as luas respetivas incluímos subgrupos nos grupos dos planetas. Cada um dos grupos deverá incluir a tag *models* que referencia o ficheiro *esferaSS.3d*, que foi uma figura gerada especialmente para a demo do sistema solar.

Desta forma, foi criado no *generator* a figura esférica com os seguintes parâmetros: 1 (raio), 20 (stacks) e 20 (slices), com o objetivo de criar esferas mais arredondadas, na tentativa de se assemelhar o máximo ao sistema solar real.

De seguida, apresentamos o resultado que obtemos para a representação do sistema solar.



5 Conclusão

Com a elaboração desta fase do projeto, o grupo conseguiu aprofundar conhecimento relativo ao processamento do ficheiros de configuração .xml como meio de desenhar as figuras do Sistema Solar.

Em termos de dificuldades sentidas, durante o desenvolvimento do projeto, estas surgiram principalmente na idealização e desenvolvimento de uma estrutura de dados capaz de armazenar as novas informações e garantir as restrições que previamente foram impostas. A dificuldade foi acentuada pelo facto de os elementos do grupo terem pouca familiaridade com a linguagem *C++* e, consequentemente, um investimento de tempo maior foi necessário para consolidação do conhecimentos necessário.

No entanto, o grupo de trabalho ganhou experiência, no que toca à elaboração de modelos exigentes, e com um maior grau de complexidade, ao se tornar necessário compreender o conceito da hierarquia ao construir as scenes e as consequentes transformações geométricas aplicadas aos elementos da scene.

O grupo considera que realizou esta etapa com sucesso na medida em que conseguimos implementar o que era proposto com sucesso e ainda melhorar algumas coisas da fase anterior, tais como os ângulos da posição da câmara.

Nas consequentes fases do projeto, é expectável que o modelo do sistema solar construído seja ainda melhor, mais detalhado e realista, o que enaltece o interesse do grupo.