UNIVERSIDADE FEDERAL DE GOIÁS

Bruno Rocha de Souza
Matheus da Silva Rodrigues
Túlio Travain Paim

Transformações de coordenadas homogêneas com o uso de matrizes de transformação.

22/09/2017

Introdução :

O objetivo do programa é realizar transformações em objetos tridimencionais (Rotação, Translação e Escala). O programa recebe o nome do arquivo de entrada e de saida.

No arquivo de entrada está especificado a quantidade de vértices do objeto, as coordenadas x,y,z dos pontos, seguidos pelas transformações a serem realizadas. É dada a primeira letra da transformação e seus valores, no formato:

T *deslocamentoEmX deslocamentoEmY deslocamentoEmZ (Translação)*

S *fatorEmX fatorEmY fatorEmZ (Escala)*

R *eixoDeRotação AnguloEmGraus*

Primeiramente, é lido o numero de coordenadas, e criado uma lista alocada dinâmicamente com as coordenadas em uma matriz *4x1(sento a quarta coordenada T = 1 para a realização do trabalho).* Após isso, é gerada uma matriz *4x4* para cada tipo de transformação. A primeira é multiplicada pela segunda, a resultante pela terceira e assim por diante.Ao chegar no final do arquivo ficamos com uma matriz *4x4* com todas as transformações a serem aplicadas. Após isso a lista de coordenadas e percorrida e cada ponto é multiplicado pela *matriz transformação*, gerando a nova coordenada transformada, que é impressa no arquivo de saída.

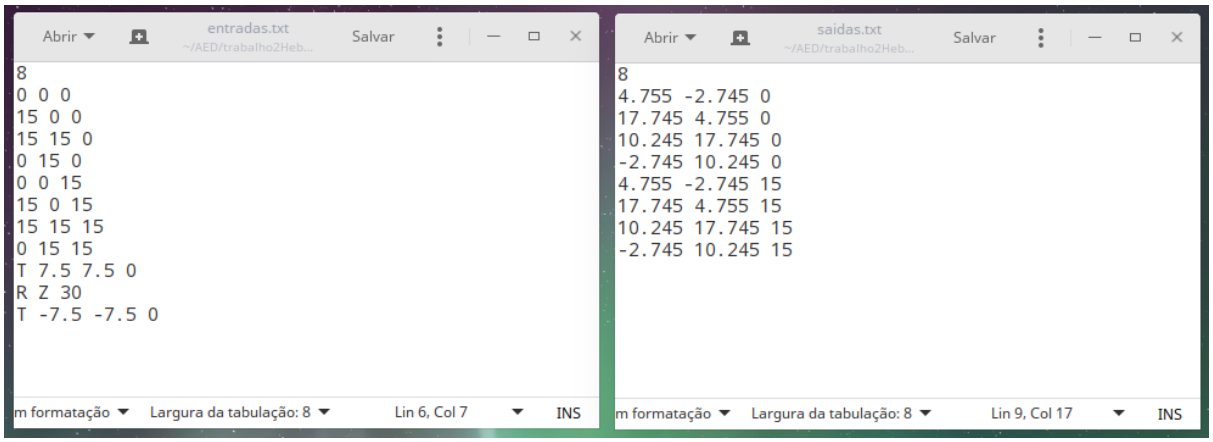

Figura 1: Arquivo de entrada como descrito. Arquivo de saída com as coordenadas transformadas

Implementação :

O programa possui o aquivo principal *main.c,* o arquivo com as funções implementadas *T3D.c* e o arquivo com a biblioteca *T3D.h.*

Além das funções requeridas pelo trabalho foram implementadas as seguintes funções :

```
/**
*Inicia e aloca um nó em uma lista dinâmica
@return Retorna um ponteiro para um matriz do tipo Mat4x1.
*/
Mat4x1 * lista_inicia();

/**
*Insere as coordenadas para cada ponto da lista que forma o objeto.
@param Mat4x1 * COORD : Ponto a ser preenchido.
@param double x,y,z : Coordenadas a serem preenchidas no ponto.
*/
void lista_insere(Mat4x1 * COORD, double x, double y, double z);
/**
*Lê uma linha de coordenadas e armazena estas em um vetor.
@param FILE * p : Arquivo onde as coordenadas estão inseridas.
@param double * vetor : Vetor para ser armazenada as coordenadas.
*/
double * ler(FILE * p, double * vetor);

/**
*Gera a matriz final de transformação.
@param char * fName : Arquivo que possui todas as transformações.
@return Mat4x4 : Retorna uma matriz final com a multiplicação de todas as
matrizes de transformação.
*/
Mat4x4 pegaMatrizes(char * fName);
/**
*Percorre a lista de pontos de um objeto e multiplica esses pontos pela matriz
de transformação, alterando o valor desses pontos.
@param Mat4x1 * cord : Ponteiro para a lista de objetos.
@param Mat4x4 transforma : Matriz de transformação a ser multiplicada
pelos pontos.
*/
void perMult(Mat4x1 * cord, Mat4x4 transforma);
/**
*Libera uma lista de tipo Mat4x1
```

```
@param Mat4x1 * lista : Lista a ser liberada da memória.
*/
void lista_libera(Mat4x1 * lista);
```

Todos os códigos estão devidamente explicados e podem ser encontrados na seguinte url: https://github.com/tuliopaim/trabalho2Hebert/tree/master/Final

Arquivo *main.c:*

https://github.com/tuliopaim/trabalho2Hebert/blob/master/Final/main.c

Arquivo *T3D.c:*

https://github.com/tuliopaim/trabalho2Hebert/blob/master/Final/T3D.c

Biblioteca *T3D.h*

https://github.com/tuliopaim/trabalho2Hebert/blob/master/Final/T3D.h

Estudo da Complexidade :

A complexidade da maioria das funções deste programa foram *O*(1), pois se tratava de manipulação de matrizes com tamanhos constantes. Porém houve algumas funções com complexidade *O*(n) e uma com uma complexidade *O*(n.m), pois os pontos e as transformações eram variáveis.

A complexidade das funções estão devidamente comentadas no arquivo T3D.c, mas será realizado uma separação das complexidade das funções neste trabalho.

Complexidade *O*(1) :

```
Mat4x4 Trans(Mat4x4 M, double deltaX, double deltaY, double deltaZ);
Mat4x4 Escala(Mat4x4 M, double FX, double FY, double FZ);
Mat4x4 Rot(Mat4x4 M, int eixo, double ang);
Mat4x4 MatComp(Mat4x4 M1, Mat4x4 M2);
Mat4x1 MatTransf(Mat4x4 M, Mat4x1 P);
double * ler(FILE * p, double * vetor);
Mat4x1 * lista_inicia();
```

Complexidade *O*(n) :

```
void Imprime(Mat4x1 *Obj, char* fName);
void Cria(Mat4x1 *Obj, char * fName);
void lista_insere(Mat4x1 * COORD, double x, double y, double z);
void perMult(Mat4x1 * cord, Mat4x4 transforma);
void lista_libera(Mat4x1 * lista);
```

Complexidade O(n.m) :

```
Mat4x4 pegaMatrizes(char * fName);
```

Conclusão :

A principal dificuldade na realização do trabalho foi a difícil compreensão do que deveria ser realizado. Depois de realizados vários testes conseguimos alcançar o resultado sem maiores problemas.

Primeiramente foram implementadas as funções base do programa. Porém ao testar manualmente as coordenadas, a ordem de multiplicação das matrizes foi implementada errôneamente.

Depois de várias tentativas, descobrimos que ao multiplicar a coordenada por cada uma das matrizes de transformação, na ordem inversa do que foi pedido na entrada, a saída foi correta. Porém, foi implementado da maneira "correta" por motivos de praticidade.

Bibliografia :

http://www.lcad.icmc.usp.br/~rosane/CG/TransfGeomAndersonIcaro.pdf