Universidade Lusófona de Humanidades e Tecnologias

ULHT260-2267
Licenciatura em Engenharia Informática
Computação Gráfica
Nuno Cruz Garcia – p6040@ulusofona.pt

Aula Prática 04b

Temas
Transformações Geométricas

Esta ficha prática é dedicada às transformações geométricas que vimos na última aula teórica. Até agora temos abordado objectos estáticos. Esta ficha aborda as transformações geométricas introduzidas na aula teórica, e que quando aplicadas continuamente no tempo nos possibilitam animar cenas 3D.

Como sabemos, as transformações geométricas são representadas por matrizes, e são aplicadas ao objecto pela multiplicação de todos os pontos do objecto pela matriz da transformação. A biblioteca GLM (OpenGL Mathematics) oferece suporte a este tipo de operações com matrizes, e está disponível aqui. É uma biblioteca apenas de *headers* (como a stb_image da ficha 4a), e por isso não é necessária qualquer instalação. Apenas download e fazer o *include "path/to/glm.hpp"*.

Há duas ideias importantes a reter da aula teórica: 1) a combinação de diferentes transformações é feita através da multiplicação das matrizes que descrevem cada transformação individualmente, e 2) a ordem da multiplicação de matrizes é inversa à ordem a que as transformações são aplicadas ao objecto.

Um escalamento seguido de uma rotação é definido da seguinte forma:

```
glm::mat4 trans = glm::mat4(1.0f);
trans = glm::rotate(trans, glm::radians(90.0f), glm::vec3(0.0, 0.0,
1.0));
trans = glm::scale(trans, glm::vec3(0.5, 0.5, 0.5));
```

O tipo de dados *mat4* é disponibilizado pela biblioteca *glm*. A primeira linha declara e inicializa uma matriz identidade 4x4, que é usada nas linhas seguintes. A segunda devolve a matriz de rotação de 90º (2º argumento - em radianos), em torno do eixo do Z (3º argumento – uma rotação em torno do eixo dos Z significa que a coordenada Z não muda). A terceira linha devolve a matriz de escalamento com um factor de escala igual a 0.5 para cada dimensão.
Como a matriz *trans* é usada em ambas as funções a biblioteca *glm* faz automaticamente a multiplicação, e por isso o resultado da última chamada é já a combinação da rotação e escalamento.

A ideia é passar esta matriz para dentro do *vertex shader* onde pode ser aplicada a todos os vértices do objecto. Como vimos em fichas anteriores, passar valores da aplicação c++ para dentro de um *shader* pode ser feito através de variáveis com o tipo *uniform*.

Dentro do *vertex shader* deve ser declarada uma variável

```
uniform mat4 transform;
```

e dentro da main() deve ser feita a multiplicação com a variável das coordenadas do vértice

```
gl_Position = transform * vec4(aPos, 1.0f);
```

Na aplicação c++ de renderização actualizamos a variável *uniform* da matriz. Os dois passos são semelhantes ao que foi feito com os *uniforms* das fichas anteriores: 1) fazer uma query sobre o endereço da variável que nos interessa, e 2) utilizar a função *glUniform* para enviar os dados para esse endereço.

```
unsigned int transformLoc = glGetUniformLocation(ourShader.ID,
                                                 "transform");
glUniformMatrix4fv(transformLoc, 1, GL_FALSE, glm::value_ptr(trans));
```

Em resumo:
. foi criada uma matriz de transformação usando a biblioteca *glm*, que é a matriz que resulta de um escalamento seguido de uma rotação.
. foi declarada uma variável uniform no *vertex shader* para receber esta matriz
. esta matriz é usada dentro do *vertex shader* para transformar as coordenadas dos vértices, através da sua multiplicação com a variável que tem as coordenadas dos vértices.

Exercício 1
Tomemos como base o código das outras fichas que constrói um triângulo ou outro polígono qualquer. Verifique que o polígono está centrado na origem.
Aplique as transformações acima mencionadas de rotação e escalamento e verifique o resultado. O que acontece se a ordem das operações for trocada?

Exercício 2
Posicione o triângulo fora do centro da janela, por exemplo num dos cantos.
O que acontece quando é aplicada apenas a transformação de escalamento? E de rotação?

Exercício 3
A operação de translação é feita com a função:

```
trans = glm::translate(trans, glm::vec3(x, y, z));
```

Aplique as tranformações de escalamento e rotação acima referidas sem que o centro do objecto se desloque. Para isto, como vimos na aula teórica, aplique a transformação de translação para a origem em primeiro lugar, e de novo para a posição do objecto por último. Ou seja:

```
Translate(a,b,c)
Rotate(...)
Scale(...)
Translate(d,e,f)
```

Que valores devem ter {a,b,c,d,e,f}?

Exercício 4
Faça o objecto rodar em torno do seu centro de forma contínua no tempo. Para isso, o ângulo de rotação deve ser obtido usando a função *glfwGetTime()*. A matriz *trans* deve ser inicializada dentro do *loop* de renderização, assim como todas as transformações necessárias.

References:

https://learnopengl.com/
https://open.gl/