OpenGL

Ricardo Dutra da Silva
Elisa de Cássia Silva Rodrigues

Universidade Tecnológica Federal do Paraná
Universidade Federal de Itajubá

2020

Transformações e Matrizes

- O OpenGL usa matrizes de transformações.
- As matrizes são passadas para vertex shader que as aplica em cada vértice.
- O OpenGL usa matrizes homogêneas.

Matrizes de Transformação

- Em OpenGL basta criar uma matriz `float`, de ordem 4×4, como já estudado.
- Usaremos em C++ a biblioteca OpenGL Mathematics (GLM) que:
  - é baseada na especificação da GLSL, possuindo tipos como `mat4`, `vec4`, etc.;
  - define funções para transformação;
  - define operações sobre matrizes e vetores.
- A discussão que segue é baseada em C++;
- Funções para Python são definidas no arquivo `lib/utils.py`.

Matriz Identidade

mat4

Cria uma matriz identidade de ordem 4×4. Parâmetro: 1.0f para indicar que é identidade.

Exemplo

```
glm::mat4 I = glm::mat4(1.0f);
```

Matriz Identidade

- Para imprimir a matriz inclua o header:

```
#include <glm/gtx/string_cast.hpp>
```

- Use o cast para string:

```
std::cout << glm::to_string(I) << std::endl;
```

Exemplo

```
mat4x4((1.000000, 0.000000, 0.000000, 0.000000), (0.000000,
1.000000, 0.000000, 0.000000), (0.000000, 0.000000, 1.000000,
0.000000), (0.000000, 0.000000, 0.000000, 1.000000))
```

Matriz de Translação

`translate`

Translada uma matriz M usando um vetor v. Parâmetros:

- M matriz transladada.
- v vetor de translação (dx, dy, dz).

Exemplo

```
glm::mat4 T = glm::translate(
  glm::mat4(1.0f),
  glm::vec3(1f,2f,3f)
);
```

Matriz de Translação

- Note que a matriz de saída está transposta.

Exemplo

```
mat4x4(
(1.000000, 0.000000, 0.000000, 0.000000),
(0.000000, 1.000000, 0.000000, 0.000000),
(0.000000, 0.000000, 1.000000, 0.000000),
(1.000000, 2.000000, 3.000000, 1.000000)
)
```

- O OpenGL usa matrizes baseadas em colunas.
- As funções para Python em `lib/utils.py` definem matrizes baseadas em linhas. Elas são explicitamente transpostas antes de enviar para o OpenGL.

Matriz de Rotação

rotate

Multiplica uma matriz M por uma matriz de rotação de ângulo *alfa* em torno de um eixo $v = (x, y, z)$. Parâmetros:

- M matriz a ser multiplicada;
- *alfa* ângulo em radianos;
- v vetor que define o eixo de rotação.

Exemplo

```
glm::mat4 T = glm::rotate(
  glm::mat4(1.0f),
  glm::radians(45.0f),
  glm::vec3(1.0f,0.0f,1.0f)
);
```

Matriz de Rotação no Eixo x

Exemplo

```
glm::mat4 T = glm::rotate(
   glm::mat4(1.0f),
   glm::radians(45.0f),
   glm::vec3(1.0f,0.0f,0.0f)
);
```

Exemplo

```
mat4x4(
(1.000000,  0.000000, 0.000000, 0.000000),
(0.000000,  0.707107, 0.707107, 0.000000),
(0.000000, -0.707107, 0.707107, 0.000000),
(0.000000,  0.000000, 0.000000, 1.000000)
)
```

Matriz de Rotação no Eixo y

Exemplo

```
glm::mat4 T = glm::rotate(
   glm::mat4(1.0f),
   glm::radians(45.0f),
   glm::vec3(0.0f,1.0f,0.0f)
);
```

Exemplo

```
mat4x4(
(0.707107, 0.000000, -0.707107, 0.000000),
(0.000000, 1.000000,  0.000000, 0.000000),
(0.707107, 0.000000,  0.707107, 0.000000),
(0.000000, 0.000000,  0.000000, 1.000000)
)
```

Matriz de Rotação no Eixo z

Exemplo

```
glm::mat4 T = glm::rotate(
   glm::mat4(1.0f),
   glm::radians(45.0f),
   glm::vec3(0.0f,0.0f,1.0f)
);
```

Exemplo

```
mat4x4(
( 0.707107, 0.707107, 0.000000, 0.000000),
(-0.707107, 0.707107, 0.000000, 0.000000),
( 0.000000, 0.000000, 1.000000, 0.000000),
( 0.000000, 0.000000, 0.000000, 1.000000)
)
```

Matriz de Escala

scale

Escala uma matriz M usando um vetor de fatores para cada eixo $v = (x, y, z)$. Parâmetros:

- M matriz em que a escala é aplicada;
- v vetor que define os fatores.

Matriz de Escala

Exemplo

```
glm::mat4 S = glm::scale(
  glm::mat4(1.0f),
  glm::vec3(0.3, 0.5, 1.0)
);
```

Exemplo

```
mat4x4(
(0.300000, 0.000000, 0.000000, 0.000000),
(0.000000, 0.500000, 0.000000, 0.000000),
(0.000000, 0.000000, 1.000000, 0.000000),
(0.000000, 0.000000, 0.000000, 1.000000)
)
```

Composição de Transformações

- A GLM permite multiplicar diretamente as matrizes.

Exemplo

```
glm::mat4 T = glm::translate(glm::mat4(1.0f),
                             glm::vec3(0.5f,-0.5f,0.0f));
glm::mat4 R = glm::rotate(glm::mat4(1.0f), glm::radians(45.0f),
                          glm::vec3(0.0f,0.0f,1.0f));
glm::mat4 S = glm::scale(glm::mat4(1.0f),
                         glm::vec3(0.3, 0.5, 1.0));
glm::mat4 M = T*R*S;
```

Exemplo

```
mat4x4(
( 0.212132,  0.212132, 0.000000, 0.000000),
(-0.353553,  0.353553, 0.000000, 0.000000),
( 0.000000,  0.000000, 1.000000, 0.000000),
( 0.500000, -0.500000, 0.000000, 1.000000)
)
```

Composição de Transformações

- Alternativamente, a matriz de composição pode ser usada como o primeiro parâmetro de cada função.

Exemplo

```
glm::mat4 M = glm::mat4(1.0f);
M = glm::scale(M, glm::vec3(0.3, 0.5, 1.0));
M = glm::rotate(M, glm::radians(45.0f), glm::vec3(0.0f,0.0f,1.0f));
M = glm::translate(M, glm::vec3(0.5f,-0.5f,0.0f));
```

Exemplo

```
mat4x4(
( 0.212132,  0.212132, 0.000000, 0.000000),
(-0.353553,  0.353553, 0.000000, 0.000000),
( 0.000000,  0.000000, 1.000000, 0.000000),
( 0.500000, -0.500000, 0.000000, 1.000000)
)
```

Programa

- O programa `transform.cpp` desenha um retângulo sobre o qual são aplicadas transformações.
- Precisamos definir uma variável para receber a matriz de transformação no shader.
- Depois passamos a matriz de transformação na função de desenho.

Vertex Shader

- Definimos uma variável `uniform` que recebe a matriz (linha 7).
- No main ela multiplica as coordenadas do vértice (linha 11).

Vertex shader

```
1  #version 330 core
2  layout (location = 0) in vec3 position;
3  layout (location = 1) in vec3 color;
4
5  out vec3 vColor;
6
7  uniform mat4 transform;
8
9  void main()
10 {
11    gl_Position = transform * vec4(position, 1.0);
12    vColor = color;
13 }
```

Desenho

- Na função `display` a matriz de transformação é criada e passada para o shader.

Vertex shader

```
void display()
{
    ...
    glm::mat4 T = glm::translate(...);
    glm::mat4 Rz = glm::rotate(...);
    glm::mat4 S = glm::scale(...);
    glm::mat4 M = T*Rz*S;

    unsigned int loc = glGetUniformLocation(program, "transform");
    glUniformMatrix4fv(loc, 1, GL_FALSE, glm::value_ptr(M));
    ...
}
```

Desenho

glGetUniformLocation

Recupera a localização de uma variável shader. Parâmetros:

- program: programa dos shaders;
- var: variável recuperada.

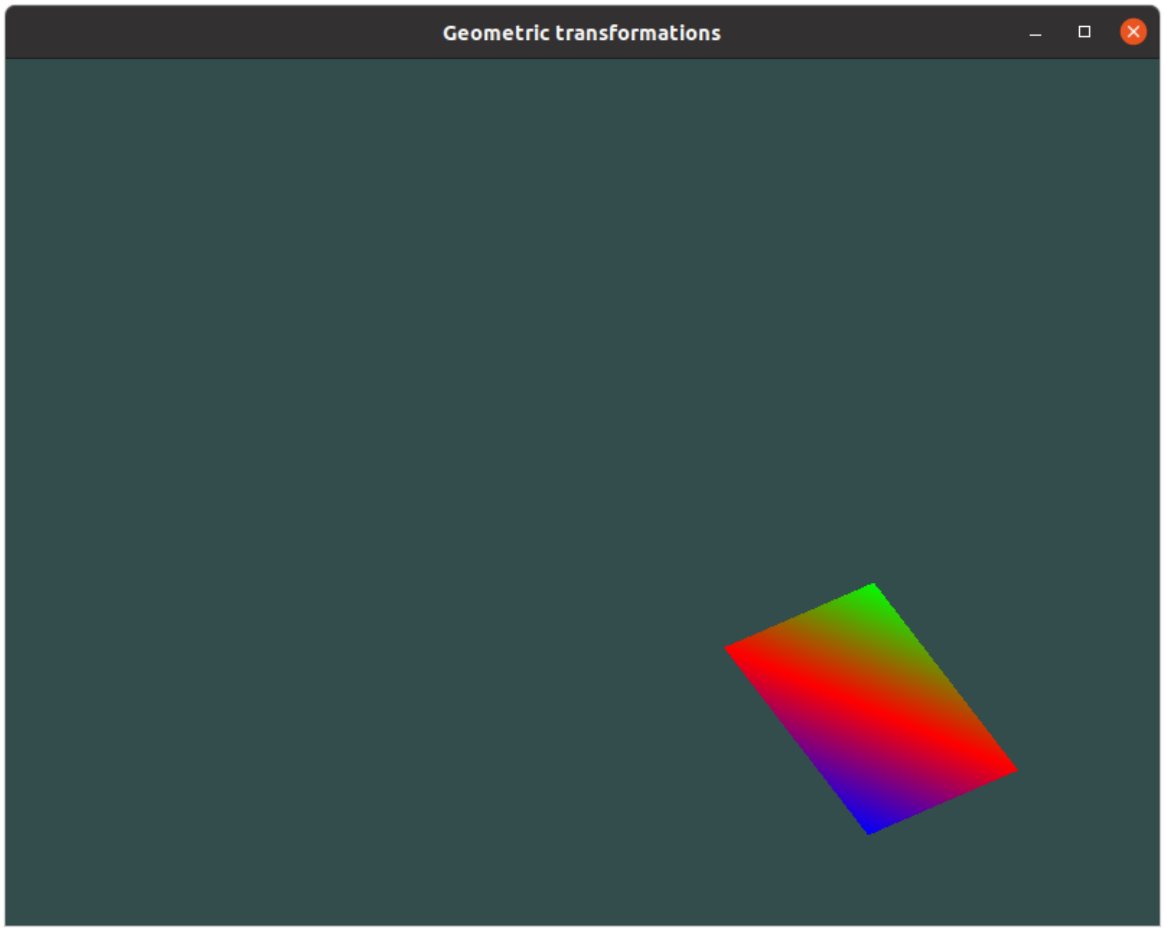
glUniformMatrix4fv

Passa a matriz float $4x4$ para o shader. Parâmetros:

- loc: Localização da variável;
- n: Número de matrizes;
- transpose: se precisa transpor a matriz;
- matriz: ponteiro para a matriz.

Desenho

- Objeto após a aplicação de escala, rotação e translação.



Animação

- O programa `transform2.cpp` tem dois modos de exibição, escolhidos no teclado:
  - 1: rotação em torno do centro do objeto;
  - 2: rotação em torno do eixo z.
- Para produzir uma animação da rotação, usamos uma função `idle`.

Animação

- Incrementando o ângulo de rotação e sinalizando redesenho.

Exemplo

```
void idle()
{
    angle = ((angle+angle_inc) < 360.0f) ? angle+angle_inc :
            360.0-angle+angle_inc;
    glutPostRedisplay();
}
```

glutIdleFunc

Executa uma função sempre que o OpenGL está ocioso (sem outros sinais para processar). Parâmetros:

- idle: função a ser executada.

Animação

- A diferença entre os modos de exibição consiste na ordem da composição das transformações de escala S, rotação R_z e translação T:
  - 1: rotação em torno do centro do objeto (TR_zS);
  - 2: rotação em torno do eixo z (R_zTS);

Exemplo

```
void display()
...
if (mode == 1)
    M = T*Rz*S;
else if (mode == 2)
    M = Rz*T*S;
...
```