# Homework #1: vector, matrix, transformation

김준호

**Abstract**

컴퓨터 그래픽스에서는 물체의 위치(position)와 이동(translation)/회전(rotation)/확대축소(scaling)와 같은 변환(transformation)을 표현하기 위해, 벡터와 행렬 연산이 필수적이다. 본 과제에서는 OpenGL Mathematics(GLM)을 이용해 벡터, 행렬, 변환 예제를 실습하는 것을 목표로 한다.

## 1 과제 소개

### 1.1 GLM 설치

OpenGL Mathematics (GLM)은 OpenGL Shading Language (GLSL) 기반의 그래픽스 소프트웨어를 위한 C++ 수학 라이브러리이다. 다음의 명령어를 통해 Ubuntu 18.04 환경에서 glm을 쉽게 설치할 수 있다.

```
sudo apt-get install libglm-dev
```

## 2 과제 목표

본 과제에서는 모던 OpenGL 프로그래밍에서 자주 활용되는 선형변환을 살펴보고 GLM을 사용해 직접 구현해보는 것을 목표로 한다. GLM에서 제공하는 다양한 함수들을 활용하여 선형변환을 실습한다.

### 2.1 열기준(Column-major) 행렬

본 과제에서 사용하는 OpenGL은 열기준(column-major)행렬을 사용한다. 예를 들면 $4 \times 4$ 행렬의 경우 다음과 같이 16개의 행렬 요소가 순차적으로 배열에 저장되어 있다고 가정한다.

$$\begin{bmatrix} m_{00} & m_{04} & m_{08} & m_{12} \\ m_{01} & m_{05} & m_{09} & m_{13} \\ m_{02} & m_{06} & m_{10} & m_{14} \\ m_{03} & m_{07} & m_{11} & m_{15} \end{bmatrix} \tag{1}$$

다음은 컴퓨터 그래픽스 프로그래밍에서 활용되는 각종 선형변환과 이에 대응하는 $4 \times 4$ 행렬을 소개하고 있다. 각 행렬을 반환하는 GLM 함수들을 사용하고 이를 main.cpp 소스코드 내에 구현하는 것이 본 과제의 수행 내용이다.

### 2.2 이동변환 (translation) 행렬

$x, y, z$축으로 각각 $d_x, d_y, d_z$ 만큼 이동하는 이동변환은 다음과 같은 $4 \times 4$ 행렬로 표현된다.

$$\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 0 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

본 과제에서는 이동변환 행렬을 반환하는 다음 함수를 사용한다.

```
template<typename T>
detail::tmat4x4<T> translate (detail::tmat4x4<T> const &m, detail::tvec3<T> const &v)
```

*국민대학교 소프트웨어학부, junho@kookmin.ac.kr*

### 2.3 회전변환(rotation) 행렬

단위벡터 $\mathbf{u} = (u_x, u_y, u_z)$를 회전축으로 하여, 원점을 중심으로 $\theta$만큼 회전하는 회전변환은 다음과 같은 $4 \times 4$ 행렬로 표현된다.

$$\begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta & 0 \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta & 0 \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$\theta$는 라디언(radian)으로 표현된 각도이다. 본 과제에서는 회전변환 행렬을 반환하는 다음 함수를 사용한다.

```
template<typename T>
detail::tmat4x4<T> rotate (detail::tmat4x4<T> const &m,
                           T const &angle,
                           detail::tvec3<T> const &axis)
```

### 2.4 확대축소(scaling) 행렬

$x, y, z$ 축으로 각각 $s_x, s_y, s_z$ 만큼 원점을 기준으로 크기를 변화시키는 확대축소변환은 다음과 같은 $4 \times 4$ 행렬로 표현된다.

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

본 과제에서는 확대축소변환 행렬을 반환하는 다음 함수를 사용한다.

```
template<typename T>
detail::tmat4x4<T> scale (detail::tmat4x4<T> const &m, detail::tvec3<T> const &v)
```

### 2.5 시점변환(view transform) 행렬

3차원에서 카메라의 자세와 관계된 시점변환은 다음과 같이 두 행렬의 곱으로 표현된 $4 \times 4$ 행렬로 표현된다.

$$\begin{bmatrix} \text{cam-x-axis}_x & \text{cam-x-axis}_y & \text{cam-x-axis}_z & 0 \\ \text{cam-y-axis}_x & \text{cam-y-axis}_y & \text{cam-y-axis}_z & 0 \\ \text{cam-z-axis}_x & \text{cam-z-axis}_y & \text{cam-z-axis}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \text{-cam-pos}_x \\ 0 & 1 & 0 & \text{-cam-pos}_y \\ 0 & 0 & 1 & \text{-cam-pos}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

본 과제에서는 시점변환 행렬을 반환하는 다음 함수를 사용한다.

```
template<typename T>
detail::tmat4x4<T>          lookAt (detail::tvec3<T> const &eye,
                                    detail::tvec3<T> const &center,
                                    detail::tvec3<T> const &up)
```

### 2.6 투영변환(projection matrix) 행렬

3차원에서 카메라의 투영변환은 직교투영(orthographic projection)인지 원근투영(perspective projection)인지에 따라 다음과 같은 $4 \times 4$행렬로 표현된다.

#### *2.6.1 직교투영(orthographic projection) 행렬*

3차원 카메라의 직교투영변환은 다음과 같은 $4 \times 4$ 행렬로 표현된다. 단, $l, r, b, t, n, f$는 각각 left, right, bottom, top, near, far의 약어이다.

$$\begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

본 과제에서는 직교투영변환 행렬을 반환하는 다음과 같은 함수를 사용한다.

```
template<typename T>
detail::tmat4x4<T> ortho (T const &left, T const &right,
                          T const &bottom, T const &top,
                          T const &zNear, T const &zFar)
```

### 2.6.2 원근투영(perspective projection) 행렬

3차원 카메라의 원근투영변환은 다음과 같은 $4 \times 4$행렬로 표현된다. 단, $l, r, b, t, n, f$는 각각 left, right, bottom, top, near, far의 약어이다.

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{7}$$

본 과제에서는 원근투영 행렬을 반환하는 다음과 같은 함수를 사용한다.

```
template<typename T>
detail::tmat4x4<T> frustum (T const &left, T const &right,
                            T const &bottom, T const &top,
                            T const &near, T const &far)

template<typename T>
detail::tmat4x4<T> perspective (T const &fovy, T const &aspect, T const &near, T const &far)
```

## 3 과제 가이드

본 과제는 GLM의 벡터, 행렬, 그리고 선형변환 함수들을 사용하여 main.cpp 프로그램을 완성하는 것이 목표이다. main.cpp의 `//TODO` 부분에 코드를 추가하여 과제를 수행한다. 자세한 방법은 아래와 같다.

1) $\mathbf{x} = (3, 5, 7)$ 벡터를 생성한다.
2) $\mathbf{x}$와 같은 벡터 $\mathbf{y}$를 생성한다.
3) $\mathbf{y} = \mathbf{x} + \mathbf{y}$를 수행한다.
4) $\mathbf{x}, \mathbf{y}$ 두 벡터간의 dot product를 계산한다.
5) $\mathbf{x} = (1, 0, 0), \mathbf{y} = (0, 1, 0)$으로 재설정하고 두 벡터간의 cross product로 $\mathbf{z}$를 생성한다.
6) $4 \times 4$ identity matrix $A$를 생성한다.
7) $A$를 다음과 같은 행렬이 되도록 변경한다.

$$\begin{bmatrix} 1 & 2 & -1 & -1 \\ 2 & 1 & 0 & 2 \\ 3 & -2 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

8) $A$의 전치행렬인 $B$행렬을 생성한다.
9) 다음의 조건을 만족하는 변환행렬을 완성한다.
   - $(1, -1, 2)$로 이동변환을 수행하는 행렬
   - $(1, 2, -1)$을 회전축으로 하여 $90°$ 회전변환을 수행하는 행렬
   - $x, y, z$축으로 각각 $2, 1, 1.5$ 만큼 크기를 변화시키는 행렬
   - 카메라가 $(0, 0, -5)$위치에 있고 카메라의 up-vector가 $(0, 1, 0)$일 때 $(0, 0, 0)$위치를 바라보도록 시점을 변환하는 행렬
   - 아래의 파라미터를 가지는 직교투영 행렬
     - left: 1, right: -1, bottom: 1, top: -1, z-near: 1, z-far: -1
   - 아래의 파리미터를 가지는 절두체 행렬
     - left: -0,1, right: 0.1, bottom: -0.1, top: 0.1, near: 0.1, far: 1000
   - 시야각 $60°$, 0.001 10000사이에 있는 물체를 그리기 위한 원근 투영 행렬. 이 때 aspect값은 1.0이다.

## 4 과제 제출방법(매우 중요!!)

- 본 과제는 개인과제이며, 각자 자신의 코드를 완성하도록 한다.
- 공지된 마감 시간까지 과제 코드를 가상대학에 업로드하도록 한다.
- 과제 코드는 **Ubuntu 18.04 LTS 환경에서 make 명령으로 컴파일 가능**하도록 작성한다.
- 과제 코드는 다음의 파일들을 하나의 압축파일로 묶어 **tar.gz** 파일 형식이나 표준 **zip**파일 형식으로만 제출하도록 한다. 이때, 압축파일의 이름은 반드시 'OOOOOOOO_HW01.tar.gz (OOOOOOOO은 자신의 학번)'과 같이 자신의 학번이 드러나도록 제출한다.
  1) 소스코드 및 리소스 파일들
  2) Makefile
- 과제에 관한 질문은 오피스아워를 활용하도록 한다. 오피스아워 이외의 시간에 도움을 받으려면 교육조교(teaching assistant, TA)에게 메일로 약속시간을 정한 후, 교육조교가 있는 연구실로 방문하도록 한다.