

[CSE4170: 기초 컴퓨터 그래픽스]

중 간 고 사

(담당교수: 임 인 성)

2017년 4월 21일

- 답은 연습지가 아니라 답안지에 기술할 것. 답안지 공간이 부족할 경우, 답안지 뒷면에 기술하고, 해당 답안지 칸에 그 사실을 명기할 것.

1. 다음 단답식 문제에 답하라.

- (a) 임의의 4행 4열 아핀 변환 행렬 M 은 다음과 같이 부행렬을 사용하여 표현할 수 있다. (여기서 $M_{3 \times 3}$ 은 3행 3열 행렬, $v_{3 \times 1}$ 은 3행 1열 행렬, 그리고 $u_{1 \times 3}$ 은 1행 3열 행렬임)

$$M = \begin{bmatrix} M_{3 \times 3} & v_{3 \times 1} \\ u_{1 \times 3} & 1 \end{bmatrix}$$

이때 $u_{1 \times 3}$ 의 세 원소 값을 기술하라.

- (b) 위 문제의 행렬 M 의 역행렬은 다음과 같이 표현할 수 있다.

$$M^{-1} = \begin{bmatrix} \hat{M}_{3 \times 3} & \hat{v}_{3 \times 1} \\ u_{1 \times 3} & 1 \end{bmatrix}$$

이때 $\hat{M}_{3 \times 3}$ 과 $\hat{v}_{3 \times 1}$ 의 값을 위 문제의 부행렬들을 사용하여 정확히 표현하라.

- (c) 3차원 공간의 두 벡터 $p = (p_x \ p_y \ p_z)^t$ 와 $q = (q_x \ q_y \ q_z)^t$ 의 외적을 $r = p \times q = (r_x \ r_y \ r_z)^t$ 이라 할 때 r_x 값을 기술하라.
- (d) (앞 문제에 이어서) 다음 조건을 만족시켜주는 3행 3열 행렬 M_c 의 내용을 기술하라.

$$\begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = M_c \begin{pmatrix} q_x \\ q_y \\ q_z \end{pmatrix}$$

- (e) 3차원 투영공간의 점 $(6.0, -6.0, -9.0, -3.0)$ 에 해당하는 아핀공간의 점의 좌표 (x, y, z) 를 기술하라.
- (f) 2차원 투영 공간에서의 두 직선 $2X + 3Y + 5W = 0$ 과 $2X + 4Y + 7W = 0$ 의 교점에 대한 아핀 공간에서의 좌표를 기술하라.

- (g) 투영 참조점이 무한대점 (point at infinity)에 위치한 투영 변환의 이름은 무엇인가?
- (h) 투영 참조점이 투영 평면에서 유한 거리만큼 떨어진 투영 변환의 이름은 무엇인가?
- (i) 만약 M 이 x, y , 그리고 z 축으로 각각 2배씩 확대한 후, y 축으로 -1만큼 이동시켜주는 4행 4열 아핀 변환 행렬이라 할 때, 이 행렬의 역행렬 M^{-1} 을 기술하라.

2. 그림 1에 도시한 바와 같이 왼쪽의 윈도우의 내용을 오른쪽 윈도우 안으로 매핑을 해주는 2차원 아핀변환에 대한 3행 3열 행렬 M 을 이동변환 $T(t_x, t_y)$, 크기변환 $S(s_x, s_y)$, 그리고 회전변환 $R(\theta)$ 등의 기본 아핀변환의 합성을 통하여 구하라. 이때 (i) 합성 과정을 반드시 기술한 후, (ii) 최종 결과 행렬을 기술할 것.

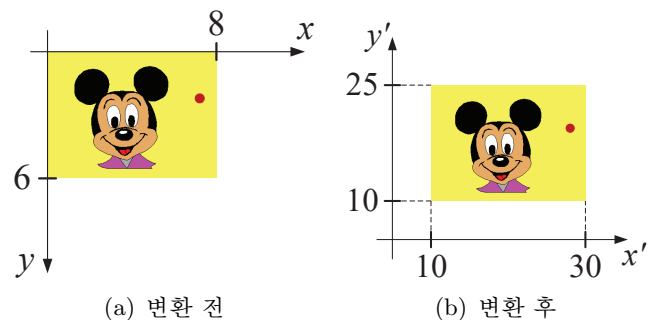


Figure 1: 2차원 윈도우 매핑

3. 2차원 공간에 주어진 한 점을 직선 $y = -x + 1$ 을 중심으로 반사 (reflection)시켜주는 3행 3열 아핀 변환 행렬 M 을 $T(t_x, t_y)$, $S(s_x, s_y)$, 그리고 $R(\theta)$ 등의 기본 아핀변환을 통하여 합성하라.
4. 2차원 아핀 변환인 이동변환 $T(t_x, t_y)$, 크기변환 $S(s_x, s_y)$, 그리고 회전변환 $R(\theta)$ 에 대한 3행 3열 행렬들을 고려하자.

- (a) $S(1, -1) \cdot T(0, t_y) = T(a, b) \cdot S(1, -1)$ 식을 만족시켜주는 a 와 b 값은 무엇인가?
- (b) $T(0, t_y) \cdot R(90^\circ) = R(90^\circ) \cdot T(c, d)$ 식을 만족시켜주는 c 와 d 값은 무엇인가?
- (c) $M_{2D} = R(90^\circ) \cdot S(1, -1) \cdot T(0, 200) \cdot R(90^\circ) = R(e^\circ) \cdot T(f, g) \cdot S(h, i)$ 를 만족시켜주는 e, f, g, h , 그리고 i 값은 무엇인가?
- (d) 위 문제의 아핀 변환 행렬 M_{2D} 의 원소 값을 정확히 기술하라.
5. 3차원 공간의 점 $p = (p_x \ p_y \ p_z)^t$ 와 벡터 $n = (n_x \ n_y \ n_z)^t$ 를 주어진 점 $q = (q_x \ q_y \ q_z)^t$ 와 벡터 $m = (m_x \ m_y \ m_z)^t$ 에 의해 정의되는 직선 L 둘레로 θ 도 만큼 돌려주는 회전 변환을 고려하자.

- (a) 먼저 2차원 공간에서의 3행 3열 회전 변환 행렬 $R(\theta)$ 를 기술하라.
- (b) 다음 $q = (1 \ 0 \ 0)^t$ 이고 $m = (0 \ 0 \ 1)^t$ 일 경우의 회전에 해당하는 4행 4열 아핀 행렬 M 의 내용을 정확히 기술하라. ($c = \cos \theta$ 와 $s = \sin \theta$ 와 같이 c 와 s 를 사용하여 답할 것)
- (c) 다음은 glm 함수를 사용하여 점 q 와 벡터 m 에 대하여 점 p 를 변환시켜주는 코드인데, 변환 행렬 M 을 계산해주는 두 문장에 오류가 있다. 이 두 문장을 제거한 후, 다시 해당 문법에 맞게 구현하라. (답을 기술할 때 편의상 `glm::`은 제거해도 무방함)

```
glm::vec3 p, n, q, m;
float theta; // angle
glm::mat4 M;

: // Set up p, n, q, m & theta here.
M = glm::translate(glm::mat4(1.0f), q);
M = glm::rotate(M, theta*TO_RADIAN, m);
p = glm::vec3(M*glm::vec4(p, 1.0f));
```

- (d) (위의 문제에서 M 행렬을 정확히 계산하였다는 가정하에) 벡터 n 도 마찬가지로 다음과 같이 변환하려하니 문제가 발생하였다. 아래 문장의 내용을 어떻게 수정해야 할지 밝혀라.

```
n = glm::vec3(M*glm::vec4(n, 1.0f));
```

6. 다음과 같은 4행 4열 행렬 M 에 의해 정의되는 3차원 아핀 변환을 고려하자.

$$M = \begin{bmatrix} a_{11} & a_{12} & a_{13} & v_1 \\ a_{21} & a_{22} & a_{23} & v_2 \\ a_{31} & a_{32} & a_{33} & v_3 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

- (a) 만약 M 이 이는 어떤 방향이 가리키는 축 둘레로 각도 θ 만큼 돌려주는 회전 변환이라 할 때, $\cos \theta$ 값을 이 행렬의 원소들을 사용하여 표현하라. (힌트: 그림 2의 회전 변환 행렬을 참조할 것)
- (b) M 이 강제 변환일 경우 벡터 $a_2 = (a_{12} \ a_{22} \ a_{32})^t$ 의 세 원소들간에 성립해야할 조건을 기술하라.
- (c) M 이 강제 변환일 경우 두 벡터 $a_2 = (a_{12} \ a_{22} \ a_{32})^t$ 와 $a_3 = (a_{13} \ a_{23} \ a_{33})^t$ 사이에 어떤 조건을 만족시켜야 할 지 기술하라.
- (d) 임의의 벡터 $n = (n_x \ n_y \ n_z)^t$ 에 대하여 이 아핀변환을 가하여 얻은 벡터 $n' = (n'_x \ n'_y \ n'_z)^t$ 은 어떤 3행 3열 행렬 L 에 대해 $n' = L \cdot n$ 과 같이 행렬과 벡터의 곱으로 계산할 수 있다. M 이 강제 변환일 경우 L 행렬의 내용을 M 의 원소들을 사용하여 기술하라.
- (e) 만약 M 이 x, y , 그리고 z 축으로 각각 2배씩 크게 한 후 z 축 방향으로 1만큼 이동시켜주는 변환일 경우의 L 행렬의 원소들을 기술하라.
- (f) 만약 M 이 z 축 둘레로 90도만큼 돌려주는 회전 변환일 경우의 L 행렬의 원소들을 기술하라.

7. 다음은 원근 투영 변환에 관한 문제이다.

- (a) 그림 3에서와 같이 COP (Center of Projection)가 $(0, -1, 0)$ 이고 PP(Projection Plane)가 $y = d$ 인 평면일 경우, 주어진 점 $p = (x \ y \ z \ 1)^t$ 을 $p' = (x' \ y' \ z' \ 1)^t$ 로 원근투영 해주는 4행 4열의 원근투영 변환행렬 M_P 를 기술하라.

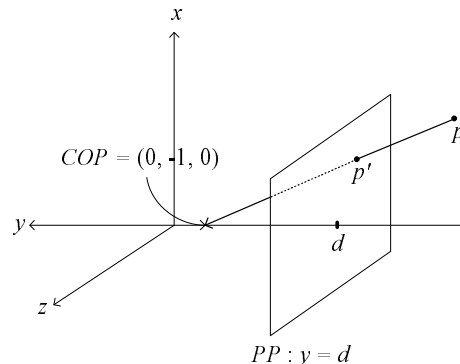


Figure 3: 원근 투영 변환

- (b) 위 문제에서의 원근 투영 변환 시 점 $(8.0, -12.0, 1.0)$ 의 깊이(depth) 값은 무엇인가?

$$R(\alpha, n_x, n_y, n_z) = \begin{bmatrix} \bar{n}_x^2(1-c) + c & \bar{n}_x\bar{n}_y(1-c) - \bar{n}_zs & \bar{n}_z\bar{n}_x(1-c) + \bar{n}_ys & 0 \\ \bar{n}_x\bar{n}_y(1-c) + \bar{n}_zs & \bar{n}_y^2(1-c) + c & \bar{n}_y\bar{n}_z(1-c) - \bar{n}_xs & 0 \\ \bar{n}_z\bar{n}_x(1-c) - \bar{n}_ys & \bar{n}_y\bar{n}_z(1-c) + \bar{n}_xs & \bar{n}_z^2(1-c) + c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 2: 회전 변환 행렬

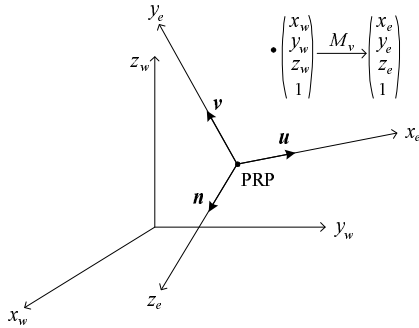


Figure 4: OpenGL 시스템에서의 뷰잉 변환

- (c) 다음은 어떤 원근 투영 변환에 해당하는 행렬이다.

$$\begin{bmatrix} \frac{9}{8} & 0 & 0 & -\frac{9}{8} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{8} & 0 & 0 & -\frac{1}{8} \end{bmatrix}$$

이 투영에서의 COP의 좌표값은 무엇일까?
(힌트: 이 행렬과 임의의 점 $(x \ y \ z \ 1)^t$ 를 곱하면 네 번째 좌표값을 통하여 이 점의 깊이
에 대한 정보를 얻을 수 있는데, COP점
자체의 깊이 값은 0이라는 사실을 고려할
것)

8. 그림 4는 세상 좌표계의 점 $(x_w \ y_w \ z_w)^t$ 를 눈 좌표계의 점 $(x_e \ y_e \ z_e)^t$ 로 바꾸어주는 OpenGL 렌더링 파이프라인에서의 뷰잉 변환에 관한 그림이다. 다음과 같은 뷰잉 변환 행렬을 보면서 답하라.

$$M_v = \begin{bmatrix} a_{11} & a_{12} & a_{13} & v_1 \\ a_{21} & a_{22} & a_{23} & v_2 \\ a_{31} & a_{32} & a_{33} & v_3 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

- (a) 카메라 시점 관점에서 볼 때, 오른쪽에 해당하는 방향 $v_r = (x_r \ y_r \ z_r)^t$ 에 대한 세상 좌표계에서 좌표값을 행렬 M_v 의 원소 값들을 사용하여 표현하라.
- (b) 카메라 시점 관점에서 볼 때, 바라보는 정방향 $v_v = (x_v \ y_v \ z_v)^t$ 에 대한 세상 좌표계에서 좌표값을 행렬 M_v 의 원소 값들을 사용하여 표현하라.

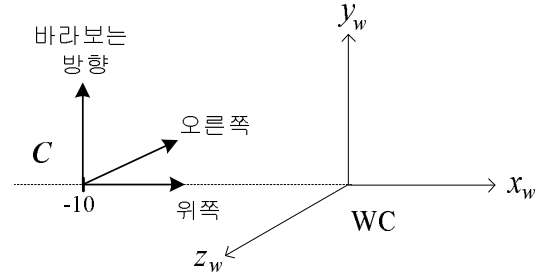


Figure 5: 카메라의 위치와 방향 설정

- (c) 카메라의 위치에 대한 기준점인 $PRP = (e_x \ e_y \ e_z)^t$ 에 대한 세상 좌표계에서 좌표값을 행렬 M_v 의 원소 값들을 사용하여 표현하라.
- (d) 그림 5에는 카메라의 위치와 방향을 설정해주는 카메라 프레임이 도시되어 있다. 이에 해당하는 M_v 를 기술하라.

9. 다음은 3차원 모델링 변환에 관한 문제이다.

- (a) 그림 6에는 그림 7에서와 같이 세상 좌표계의 원점 주변에 도시한 소 모델을 적절한 모델링 변환을 통하여 세상에 배치해주는 프로그램의 일부가 주어져 있다. 이 코드가 올바르게 작동하기 위하여 (A)에서부터 (H)까지 들어가야 할 값을 정확히 기술하라. (참고: 변수 ViewMatrix와 ProjectionMatrix에는 그 이름이 의미하는 행렬이 이미 계산되어 지정이 되어 있음)
- (b) (위 문제에서 구한 값들에 대하여) 다음 함수 호출이 리턴해주는 4행 4열 행렬을 정확히 기술하라. (힌트: 이 변환에 의해 기본 프레임이 어떻게 회전하는지 생각해 볼 것)
- ```
glm::rotate(glm::mat4(1.0f),
 (E)*TO_RADIAN,
 glm::vec3((F), (G), (H)))
```

10. 다음은 3차원 공간에서의 두 프레임간의 변환에 관한 문제이다.

```

for (i = 0; i <= 180; i += 30) {
 float angle = (float) i;

 ModelViewMatrix = glm::rotate(ViewMatrix, (A)*TO_RADIAN, glm::vec3((B), (C), (D)));
 ModelViewMatrix = glm::translate(ModelViewMatrix, glm::vec3(-4.0f, 0.0f, 0.0f));
 ModelViewMatrix = glm::rotate(ModelViewMatrix, (E)*TO_RADIAN, glm::vec3((F), (G),
(H)));
 ModelViewProjectionMatrix = ProjectionMatrix * ModelViewMatrix;

 glUniformMatrix4fv(loc_MVPMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
 glLineWidth(5.0f);
 draw_axes();
 glLineWidth(1.0f);
 draw_cow(0.3f, 0.3f, 0.3f); // Cow color = (0.3, 0.3, 0.3)
}

```

Figure 6: 3차원 모델링 변환 코드

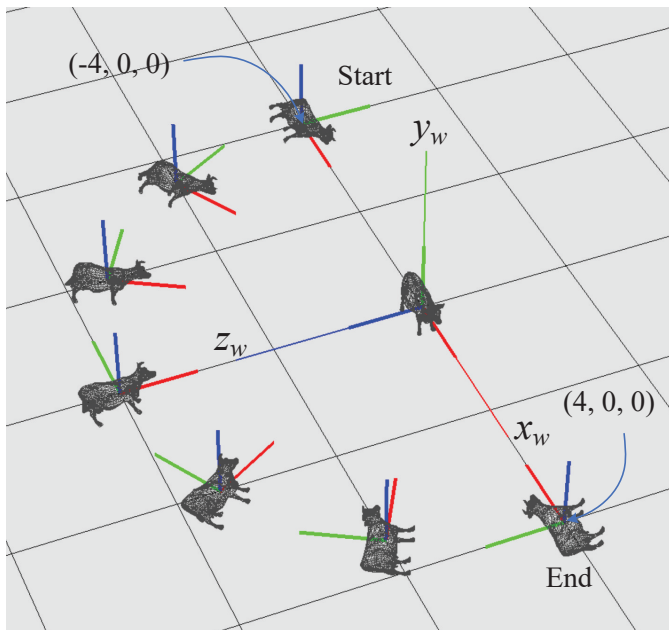


Figure 7: 3차원 모델링 변환

- (a) 그림 8의 A 소의 각 꼭지점들을 B 소의 대응되는 점으로 매핑해주는 4행 4열 행렬  $M_{cow}$ 를 다음과 같이 정의하자.

$$M_{cow} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & v_1 \\ a_{21} & a_{22} & a_{23} & v_2 \\ a_{31} & a_{32} & a_{33} & v_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

이때 왼쪽-위쪽 3행 3열 부행렬에 해당하는 부분의  $a_{11}$ 에서  $a_{33}$ 까지의 9개의 원소값을 정확히 기술하라.

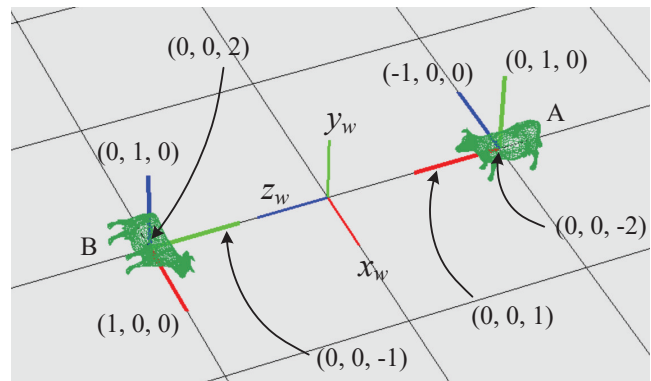


Figure 8: 두 프레임간의 변환

- (b)  $M_{cow}$ 를 구성하는 회전 변환은  $(n_x, 1, n_z)$  벡터가 가리키는 직선 둘레로  $\alpha$ 도만큼 회전시켜주는 회전변환에 해당한다. 과연  $\alpha$ 가 몇 도인지 정확히 기술하라. (참고: 그림 2에 주어진 회전변환 행렬을 참조하고, 각도는 0도와 180도 사이의 각으로 기술할 것)
- (c) 위 문제에서의  $n_x$ 와  $n_z$  값을 기술하라.
11. 다음은 카메라의 설정에 관한 문제이다. 그림 9가 암시하는 카메라의 위치와 방향 설정에 대한 뷰잉 변환 행렬을 구하는 코드가 그림 10에 주어져 있다.
- (a) 이 그림에서 이 함수의 인자 중의 하나인 3차원 벡터 **center**에 해당하는 기호의 이름을 기술하라.
- (b) 이 그림의  $u$ ,  $v$ , 그리고  $n$  벡터는 각각 카메라를 기준으로 오른쪽, 위쪽, 그리고 바

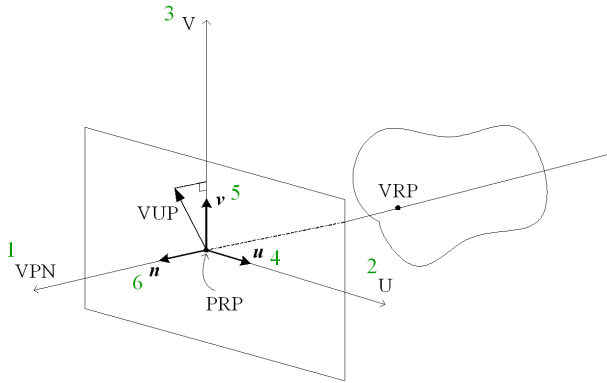


Figure 9: 카메라의 위치와 방향 설정

라보는 반대 방향에 대한 단위 벡터 (unit vector)들이다. 이 함수의 Line (a) 문장 수행 후 3차원 벡터  $f$ 에 저장되는 값을 위의 벡터들을 사용하여 정확히 기술하라.

- (c) 이 함수가 올바르게 작동하기 위하여 Line (b)의 (A)와 (B)에 들어갈 내용을 이 함수의 문법에 맞게 정확히 기술하라.
- (d) 현재 Line (c)부터 Line (e)의 세 문장에는 심각한 오류가 있다. 이를 올바르게 수정하라.
- (e) 다음은 `glm::translate(*, *)` 함수의 한 사용 예를 보여주고 있다.

```
glm::vec3 V;
glm::mat4 M;

V = glm::vec3(1.0f, -5.0f, 0.0f);
M = glm::mat4(1.0f);
M = glm::translate(M, V);
```

(바로 위 문제의 오류를 수정한 후) Line (c)부터 Line (e)의 세 문장은 이 함수를 사용하여 한 문장으로 대체하여도 올바르게 작동은 하는데, 과연 그 문장이 무엇인지 함수의 문법에 맞게 정확히 기술하라.

12. 다음은 3차원 공간에서의 카메라의 조작에 관한 문제이다. 현재 가상의 카메라는 다음과 같이 정의가 되어 있는데,

```
typedef struct _CAMERA {
 glm::vec3 pos;
 glm::vec3 uaxis, vaxis, naxis;
 float fov_y, aspect_ratio, near_clip,
 far_clip;
} CAMERA;
```

CAMERA camera;

각 변수의 의미는 수업 시간에 설명하였다.

- (a) 그림 11(a)의 코드에서 정의하고 있는 `set_ViewMatrix_from_camera_frame()` 함수는 전역 변수로 정의된 `camera`의 내용을 사용하여 `glm::mat4` 타입으로 선언한 뷰잉 변환 행렬 `ViewMatrix`를 설정하고 있다. 이때 (A), (E), 그리고 (G)에 들어갈 들어갈 내용을 이 함수의 문법에 맞게 정확히 기술하라.
- (b) (바로 위 문제에 이어) (I)에 들어갈 들어갈 내용을 이 함수의 문법에 맞게 정확히 기술하라.
- (c) 사용자가 왼쪽 마우스 버튼을 누른 후 움직일 때 수행이 되는 모션 콜백 함수 내에서 마우스가 좌우로 움직인 양 `delta`에 대해 다음과 같이 그림 11(b)에 정의된 함수를 호출하려 한다.

```
move_camera(delta, camera.naxis);
```

양수 값을 가지는 `delta` 값에 대해 매번 이 함수를 호출할 때마다 카메라는 어떻게 움직일 지 정확히 기술하라. (방향을 분명히 밝힐 것)

- (d) 그림 11(c)의 코드를 보자(여기서 `M`은 `glm::mat4` 타입의 변수임). 지금 사용자의 마우스 조작에 의해 카메라 프레임을 '카메라 기준점 `camera.pos`에서 카메라가 바라보는 앞쪽으로 정확히 `2.0f`만큼 떨어진 지점을 지나고 `camera.vaxis` 방향을 향한 직선' 둘레로 `del`도 만큼 회전시키려 한다. Line (a)에서 Line (c)의 세 문장은 이러한 카메라 프레임 회전을 위한 기하 변환 행렬을 계산하고 있는데, 아직 완성이 된 상태가 아니다. 원하는 카메라 조작이 이뤄지도록 하기 위해 각 문장의 어느 부분을 어떻게 수정해야 할 지 이 코드의 문법에 맞게 정확히 기술하라. (카메라 프레임에 대한 세 방향 벡터의 길이는 1로 가정하고, 해당 문장에 대해 고칠 것이 없을 경우 없다고 답할 것)
- (e) (바로 위 문제에서 코드를 정확히 수정했다는 가정 하에) 현재 Line (c) 이후의 코드의 내용에 큰 문제가 있다. 그것이 무엇인지 명확히 밝히고 올바르게 수정하라.

# 서강대학교

```

glm::mat4 mylookAt(glm::vec3 const & eye, glm::vec3 const & center,
 glm::vec3 const & up) {
 glm::vec3 f, s, u;
 glm::mat4 Result;

 f = glm::normalize(eye - center); // Line (a)
 s = glm::normalize(glm::cross((A) , (B))); // Line (b)
 u = glm::cross(f, s);

 Result = glm::mat4(1.0f);
 Result[0][0] = s.x; Result[1][0] = s.y; Result[2][0] = s.z;
 Result[0][1] = u.x; Result[1][1] = u.y; Result[2][1] = u.z;
 Result[0][2] = f.x; Result[1][2] = f.y; Result[2][2] = f.z;
 Result[3][0] = glm::dot(s, eye); // Line (c)
 Result[3][1] = glm::dot(u, eye); // Line (d)
 Result[3][2] = glm::dot(f, eye); // Line (e)
 return Result;
}

```

Figure 10: mylookAt() 함수의 구현

```

void set_ViewMatrix_from_camera_frame(void) {
 ViewMatrix[0] = glm::vec4((A) , (B) , (C) , 0.0f);
 ViewMatrix[1] = glm::vec4((D) , camera.vaxis.y, (E) , 0.0f);
 ViewMatrix[2] = glm::vec4((F) , (G) , (H) , 0.0f);
 ViewMatrix[3] = glm::vec4(0.0f, 0.0f, 0.0f, 1.0f);
 ViewMatrix = glm::translate(ViewMatrix, (I));
}

```

(a) 뷰잉 변환 행렬의 설정

```

void move_camera(float del, glm::vec3 axis) {
 camera.pos += del*axis;
}

```

(b) 카메라의 이동

```

M = glm::translate(glm::mat4(1.0f), camera.pos + 2.0f*camera.naxis); // Line (a)
M = glm::rotate(M, del*TO_RADIAN, camera.vaxis); // Line (b)
M = glm::translate(M, camera.pos + 2.0f*camera.naxis); // Line (c)

```

```

camera.uaxis = glm::vec3(M*glm::vec4(camera.uaxis, 1.0f));
camera.vaxis = glm::vec3(M*glm::vec4(camera.vaxis, 1.0f));
camera.naxis = glm::vec3(M*glm::vec4(camera.naxis, 1.0f));
camera.pos = glm::vec3(M*glm::vec4(camera.pos, 1.0f));

```

```

set_ViewMatrix_from_camera_frame();

```

(c) 카메라의 회전

Figure 11: 뷰잉 변환 행렬 ViewMatrix의 조작