[CSE4170: 기초 컴퓨터 그래픽스]

중 간 고 사

(담당교수: 임 인 성)

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

1. 2차원 아핀변환인 이동변환 $T(t_x, t_y)$, 크기변환 $S(s_x, s_y)$, 그리고 회전변환 $R(\theta)$에 대한 3행 3열 행렬들을 고려하자.

   (a) $R(\theta) \cdot S(-1, 1) = S(-1, 1) \cdot R(\alpha)$ 식을 만족시켜주는 α 값은 무엇인가?

   (b) $T(t_x, t_y) \cdot S(-1, 1) = S(-1, 1) \cdot T(\beta, \gamma)$ 식을 만족시켜주는 β와 γ 값은 무엇인가?

   (c) $R(\theta) \cdot T(t_x, t_y) = T(\delta, \epsilon) \cdot R(\theta)$ 식을 만족시켜주는 δ와 ϵ 값은 무엇인가?

2. (1번 문제를 고려하면서) 다음 질문에 답하라.

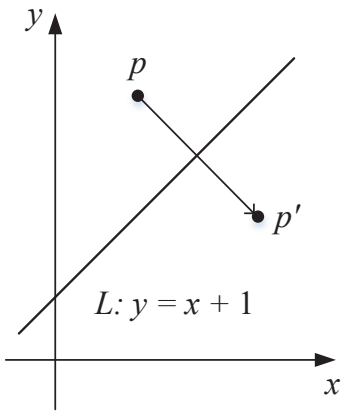

Figure 1: 직선에 대한 반사

   (a) 그림 1에서와 같이 2차원 공간의 점 $p = (x\ y\ 1)^t$를 직선 $L : y = x + 1$에 대하여 반사 시켜 $p' = (x'\ y'\ 1)^t$로 변환 해주는 3행 3열 행렬 M을 $M = N \cdot T(0, -1)$과 같이 표현한다고 할 때, 이때의 3행 3열 아핀변환 행렬 N을 $T(t_x, t_y)$, $S(s_x, s_y)$, 그리고 $R(\theta)$ 등의 기본 아핀변환을 통하여 합성하라.

   (b) 위에서 구한 행렬 M은 $M = S(s_x, s_y) \cdot T(t_x, t_y) \cdot R(90)$과 같이 세 개의 행렬의 곱으로 표현할 수 있다. 이때 여기에 들어갈 t_x, t_y, s_x, 그리고 s_y 값을 기술하라.

3. 그림 2에서와 같이 왼쪽의 윈도우의 내용을 오른쪽 윈도우 안으로 매핑을 해주는 2차원 아핀변환에 대한 3행 3열 행렬 M을 $T(t_x, t_y)$, $S(s_x, s_y)$, 그리고 $R(\theta)$ 등의 기본 아핀변환의 합성을 통하여 구하라 (여기서 왼쪽의 윈도우는 각 변의 길이가 2이고 중심이 원점이 사각형인데, (i) 합성 과정을 반드시 기술한 후, (ii) 최종 결과 행렬을 기술할 것).

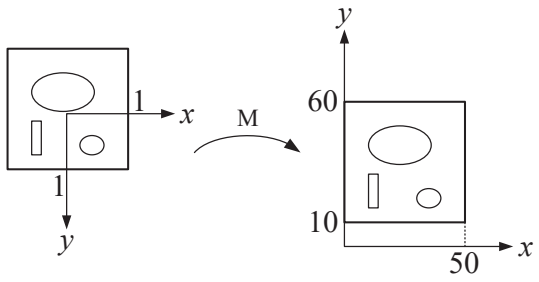

Figure 2: 2차원 윈도우 매핑 변환

4. 그림 3은 주어진 법선 벡터 (normal vector) $n = (n_x\ n_y\ n_z\ 0)^t$에 대하여 4행 4열 행렬 M이 의미하는 어떤 아핀변환을 가하는 과정을 보여주고 있다.

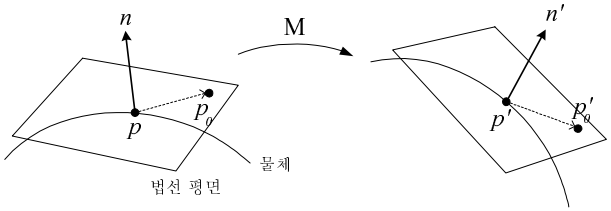

Figure 3: 법선 벡터의 변환

   (a) 이 아핀변환을 가하여 얻은 벡터 $n' = (n'_x\ n'_y\ n'_z\ 0)^t$을 어떤 4행 4열 행렬 N에

대하여 $n' = N \cdot n$과 같이 표현한다고 할 때, 이 행렬 N을 M 행렬을 사용하여 표현하라(힌트: 다음의 글을 참조할 것).

> 그림 3에서와 같이 주어진 점 $p = (x\ y\ z\ 1)^t$에서의 법선 벡터가 n이라고 하자. 이때 점 p에서의 법선 평면 상의 임의의 점 $p_0 = (x_0\ y_0\ z_0\ 1)^t$에 대하여 $n^t \cdot (p_0 - p) = 0$와 같은 관계가 성립한다. p, p_0, 그리고 n이 변환 행렬 M에 의해 각각 $p' = (x'\ y'\ z'\ 1)^t$, $p'_0 = (x'_0\ y'_0\ z'_0\ 1)^t$, n'으로 변환이 된다고 하면, $p' = M \cdot p$와 $p'_0 = M \cdot p_0$로부터 p와 p_0를 구해 위의 관계식에 대입하여 다음과 같은 식을 얻게 된다. $\cdots$

(b) 위 문제의 아핀변환 M이 강체변환(rigid-body transformation)이라고 하자. 이때 M을 다음과 같이 정의할 때,

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

이 행렬의 왼쪽-위쪽의 3행 3열 부행렬 $M_{3\times3}$의 세 개의 열벡터를 각각 a_1, a_2, a_3라 할 때, 이 세 개의 벡터가 만족하는 수학적인 성질을 벡터의 내적을 사용하여 정확히 기술하라.

(c) 문제 (b)의 조건하에, 문제 (a)의 행렬 N의 왼쪽-위쪽의 3행 3열 부행렬 $N_{3\times3}$의 내용을 정확히 기술하라.

5. 다음은 두 프레임간의 변환에 관한 문제이다.

(a) 그림 4의 A 소는 점 $(0, 0, -2)$를 원점으로 하는 자신의 프레임을 기준으로 세상 좌표계에 존재하고 있는데, 이 프레임의 각 축의 방향이 축 옆에 기술되어 있다. 이때 이 프레임을 (0, 0, -1) 방향과 (0, 1, 0) 방향이 각각 세상 좌표계의 x_w축과 y_w 방향과 일치하는 방식으로 세상 좌표계와 일치시켜주려한다. 이때 필요한 4행 4열 행렬 M_a의 내용을 기술하라.

(b) 한편 B 소는 점 $(0, 0, 2)$를 중심으로 하는 프레임을 기준으로 세상 좌표계에 존재하고 있는데, A 소의 각 꼭지점들을 B 소의 대응되는 점으로 매핑해주는 4행 4열 행렬

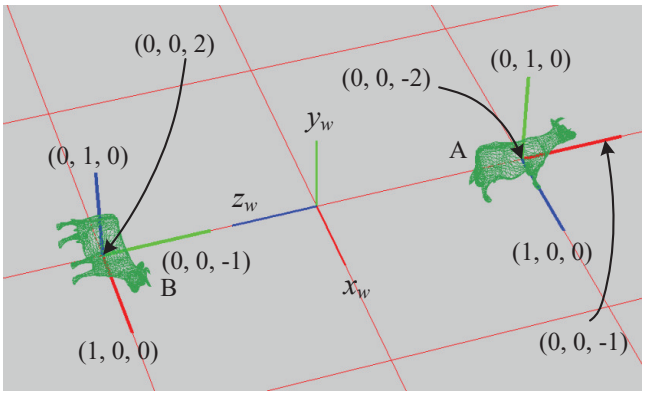

Figure 4: 두 프레임간의 변환

M_b를 $M_b = T_2 \cdot R \cdot T_1$와 같이 두 개의 이동변환 행렬 T_1, T_2와 한 개의 회전변환 행렬 R의 곱으로 표현하라(반드시 이 세 개의 4행 4열 행렬의 내용을 정확히 기술할 것).

(c) 위 문제의 R 행렬은 임의의 점을 $(-1, n_y, n_z)$ 벡터가 가리키는 직선 둘레로 α도만큼 회전시켜주는 회전변환에 해당한다. 이때 n_y와 n_z, 그리고 α 값을 기술하라. (참고: 그림 5에 주어진 회전변환 행렬을 참조하고, 각도는 0도와 180도 사이의 각으로 기술할 것)

6. 다음은 간단한 모델링 변환에 관한 문제이다.

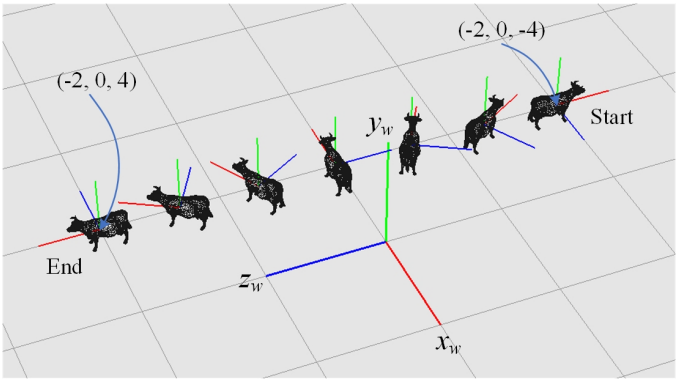

Figure 6: 간단한 모델링 변환

그림 6에는 세상 좌표계의 $(-2, 0, -4)$ 지점에서 $(-2, 0, 4)$ 지점까지의 직선 경로를 따라 균일한 속도로 이동 및 회전을 하는 소의 모습이 도시되어 있다. 소 물체는 자신의 모델링 좌표계의 x_m축 방향을 바라보고 있고, 등은 y_m축 방향을 향하고 있는데, y_m축 둘레로 총 180도 회전을 하면서 이동하고 있다. 또한 소의 모델링 좌표

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

Figure 5: 회전 변환 행렬

```
for (int i = 0; i <= 180; i += 30) {
   float angle = (float)i;
   ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(t_x, t_y, t_z));
   ModelViewMatrix = glm::rotate(ModelViewMatrix, (α)*TO_RADIAN,
            glm::vec3(n_x, n_y, n_z));
   ModelViewProjectionMatrix = ProjectionMatrix * ModelViewMatrix;
   glUniformMatrix4fv(loc_ModelViewProjectionMatrix_simple, 1, GL_FALSE,
            &ModelViewProjectionMatrix[0][0]);
   glLineWidth(2.0f);
   draw_axes();
   glLineWidth(1.0f);
   draw_cow(0.3f, 0.3f, 0.3f); // cow color = (0.3f, 0.3f, 0.3f)
}
```

Figure 7: 간단한 모델링 변환 코드

계의 원점에 해당하는 점은 위의 두 점간의 직선 상에서 이동을 하고 있다. 그림 7은 이렇게 움직이는 소를 그려주는 프로그램의 일부가 주어져 있는데, 이 코드가 올바르게 작동하기 위하여 t_x, t_y, t_z, α, n_x, n_y, 그리고 n_z에 들어갈 값을 정확히 기술하라. (참고: 변수 `ViewMatrix`와 `ProjectionMatrix`에는 그 이름이 의미하는 행렬이 이미 계산되어 지정이 되어 있음)

7. 다음은 카메라의 설정에 관한 문제이다. 그림 8에는 OpenGL Compatibility Profile에서 제공하는 `gluLookAt(*)` 함수의 호출에 대하여 뷰잉변환 행렬을 계산하는 과정(이하 "이 그림")이 주어져 있으며, 그림 9에는 glm::lookAt() 함수에 대한 구현 코드(이하 "이 함수")가 주어져 있는데, 이들을 참조하며 답하라.

   (a) 이 그림에서 이 함수의 인자 중의 하나인 3차원 벡터 `eye`에 해당하는 기호의 이름을 기술하라.

   (b) 이 그림의 u, v, 그리고 n 벡터는 각각 카메라를 기준으로 오른쪽, 위쪽, 그리고 바라보는 반대 방향에 대한 단위 벡터(unit vector)들이다. 이 함수의 `Line (a)` 문장 수행 후 벡터 `f`에 저장되는 값을 위의 벡터들을 사용하여 정확히 기술하라.

   (c) 문맥 상 이 함수의 `Line (b)`와 `Line (c)` 문장의 `smile(*,*)` 함수는 두 개의 3차원 벡터를 인자로 받아 어떠한 계산을 해줄까?

   (d) 이 함수의 `Line (d)` 문장 수행 결과 4행 4열 행렬 `Result`에는 어떤 값이 저장이 될까?

   (e) 문맥 상 이 함수의 `Line (e)`와 `Line (f)`의 빈곳에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라.

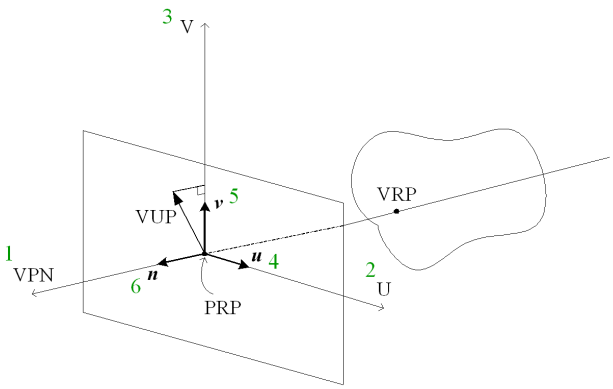

Figure 8: `gluLookAt(*)` 함수를 통한 카메라의 설정

8. 다음은 원근투영 변환에 관한 문제이다.

   그림 10에서와 같이 COP(Center of Projecton)가 $(1,0,0)$이고 PP(Projection Plane)이 $x = 9$

```
namespace glm {
  ...
  template <typename T, precision P> GLM_FUNC_QUALIFIER tmat4x4<T, P>
     lookAt(tvec3<T, P> const &eye, tvec3<T, P> const &center, tvec3<T, P> const &up) {
        tvec3<T, P> const f(normalize(center - eye)); // Line (a)
        tvec3<T, P> const s(normalize(smile(f, up))); // Line (b)
        tvec3<T, P> const u(smile(s, f)); // Line (c)
        tmat4x4<T, P> Result(1); // Line (d)
        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] = -dot(s, eye);
        Result[3][1] =               ; // Line (e)
        Result[3][2] =               ; // Line (f)
        return Result;
     }
}
```

Figure 9: glm::lookAt() 함수의 구현

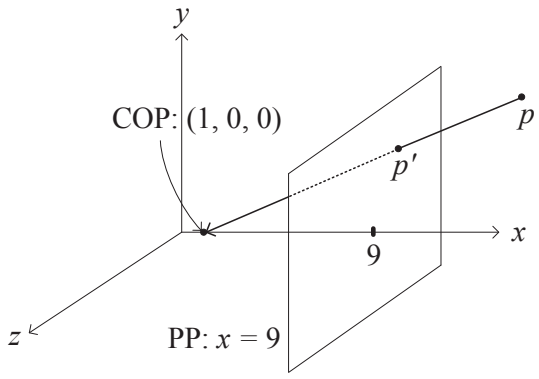

Figure 10: 원근 투영 변환

인 상황에서, 주어진 점 $p = (x\ y\ z\ 1)^t$을 $p' = (x'\ y'\ z'\ 1)^t$로 변환해주는 4행 4열의 원근투영 변환행렬 M_P를 기술하라.

9. 그림 11에는 카메라의 위치와 방향을 설정해 주는 카메라 프레임이 도시되어 있다. 이 경우 OpenGL 시스템에서 사용하는 4행 4열 뷰잉변환 행렬 M_V를 기술하라.

10. 다음은 직교 투영 변환과 관련한 문제이다. 그림 12에서와 같이 3차원 공간의 두 점 $(1,1,1)$과 $(6,6,11)$에 의해 정의되는 직육면체의 내용을 다른 두 점 $(-1,-1,-1)$과 $(1,1,1)$에 의해 정의되는 정육면체의 영역으로 매핑해주는 4행 4열 아핀 변환 행렬 M_{ortho}를 기술하라. (주의: 변환 후 z축의 방향이 반대가 되도록 네 모서리를 마춰주어야 함)

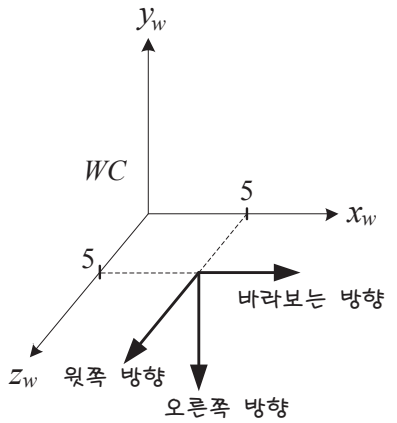

Figure 11: 카메라의 위치와 방향 설정

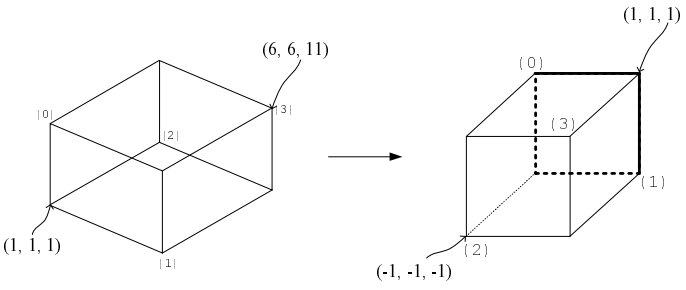

Figure 12: 직교 투영 변환

11. 시험지 뒤에 첨부한 프로그램은 적절한 모델링 변환을 통하여 자동차를 세상 좌표계로 그려주는 OpenGL 프로그램이다. 이 프로그램과 그림 13을 보면서 답하라.

    (a) 이 프로그램은 그림 13에 주어진 자동차에 대한 트리 구조를 어떤한 방식으로 탐색을 하고 있는가? 자료 구조 시간에 배운 용어를 사용할 것.

    (b) 프로그램 문맥상 13번 문장의 (A)에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라.

    (c) 프로그램 문맥상 13번 문장의 (B), (C), (D), 그리고 (E)에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라. (주의: glm 함수에서는 기본적으로 각도는 라디안을 사용하므로 이 프로그램에서 정의한 상수 `TO_RADIAN`을 적절히 사용할 것)

    (d) 이 프로그램에서 43번 문장 수행 후 변수 `ModelMatrix_CAR_WHEEL`에 저장되는 4행 4열의 내용을 그림 13의 행렬 기호들을 사용하여 표현하라.

    (e) `draw_wheel_and_nut()` 함수가 41번 문장에서 호출되어 수행이 되는 과정에서, 변수 `i`가 4일 때 14 문장 수행 결과 변수 `ModelMatrix_CAR_NUT`에 저장되는 행렬의 내용을 그림 13의 행렬 기호들을 사용하여 표현하라.

    (f) 이 프로그램의 35번 문장은 자동차의 운전석에 배치한 카메라의 위치와 방향을 기술해주는 프레임을 그려주는 역할을 한다 (x, y, 그리고 z축 각각이 u, v, 그리고 n축에 대응함을 상기할 것). 해당 부분의 코드를 볼 때 이 자동차의 앞쪽 방향은 자신의 모델링 좌표계를 기준으로 어느 축 방향을 향하고 있을까? "양의 x축" 또는 "음의 x축"과 같이 해당 축과 방향을 정확히 기술할 것.

    (g) 이 프로그램의 41번 문장과 46번 문장에서 그려주는 0번과 1번 바퀴와는 달리 52번 문장과 58번 문장에서 그려주는 2번과 3번 바퀴에 대해서는 크기변환(scale)이 수행되는 이유는 무엇일까?

12. 다음 문제에 답하라. 필요할 경우 CC, EC, MC, NDC, WC, 그리고 WdC (Window Coordinate) 등의 OpenGL 좌표계 이름을 사용하라.

    (a) 2차원 공간에서 원점을 지나고 길이가 1인 벡터 p 방향을 향하는 직선을 고려하자. 이때 한 점 q가 주어졌을 때, 이 점과 이 직선과의 최단 거리를 벡터의 내적 연산 및 벡터의 길이 연산자를 사용하여 표현하라. (참고: 내적 연산자 기호는 $\circ$, 그리고 벡터의 길이 연산자는 $\|\cdot\|$ 사용할 것)

    (b) 3차원 공간에서 두 벡터 $(1, 2, 3)$과 $(4, 5, 6)$에 의해 정의되는 삼각형의 면적을 구하라.

    (c) 2차원 공간의 두 직선 $3x + 2y + 5 = 0$과 $6x + 4y + 5 = 0$의 교점에 대한 투영 공간에서의 동차 좌표를 상수만 사용하여 기술하라.

    (d) 투영 참조점이 무한대점 (point at infinity)에 위치한 투영 변환의 이름은 무엇인가?

    (e) OpenGL의 뷰잉 파이프라인에서 항상 3차원 공간의 $[-1, 1]\times[-1, 1]\times[-1, 1]$의 영역만 고려하는 좌표계의 이름은 무엇인가?

    (f) OpenGL의 뷰잉 파이프라인에서 촬영한 필름을 현상한 후 인화지에 확대/인화하는 과정은 정확히 어느 좌표계에서 어느 좌표계로 보내주는 과정에 해당하는가?

    (g) OpenGL의 뷰잉 파이프라인에서 '카메라의 위치와 방향을 설정'해주는 변환은 정확히 어느 좌표계에서 어느 좌표계로 보내주는 변환인가?

    (h) 다음은 OpenGL Core Profile의 간단한 vertex shader의 예를 보여주고 있다.

```
#version 330
uniform mat4 u_Matrix;
uniform vec3 u_color;
layout (location = 0) in vec4 a_pos;
out vec4 v_color;
void main(void) {
  v_color = vec4(u_color, 1.0f);
  gl_Position = u_Matrix * a_pos;
}
```

    정상적인 렌더링이 수행될 경우 MC와 가장 관련이 높은 변수의 이름을 기술하라.

    (i) (위 문제에 이어) 원근 나눗셈 (perspective division)과 가장 관련이 있는 변수의 이름을 기술하라.

    (j) (위 문제에 이어) `u_Matrix`는 어떤 좌표계에서 어떤 좌표계까지의 기하변환 행렬을 저장하고 있을까?

    (k) 3차원 공간에서 원점을 지나고 벡터 $(0\ 0\ 1)^t$이 가리키는 방향에 의해 정의되는

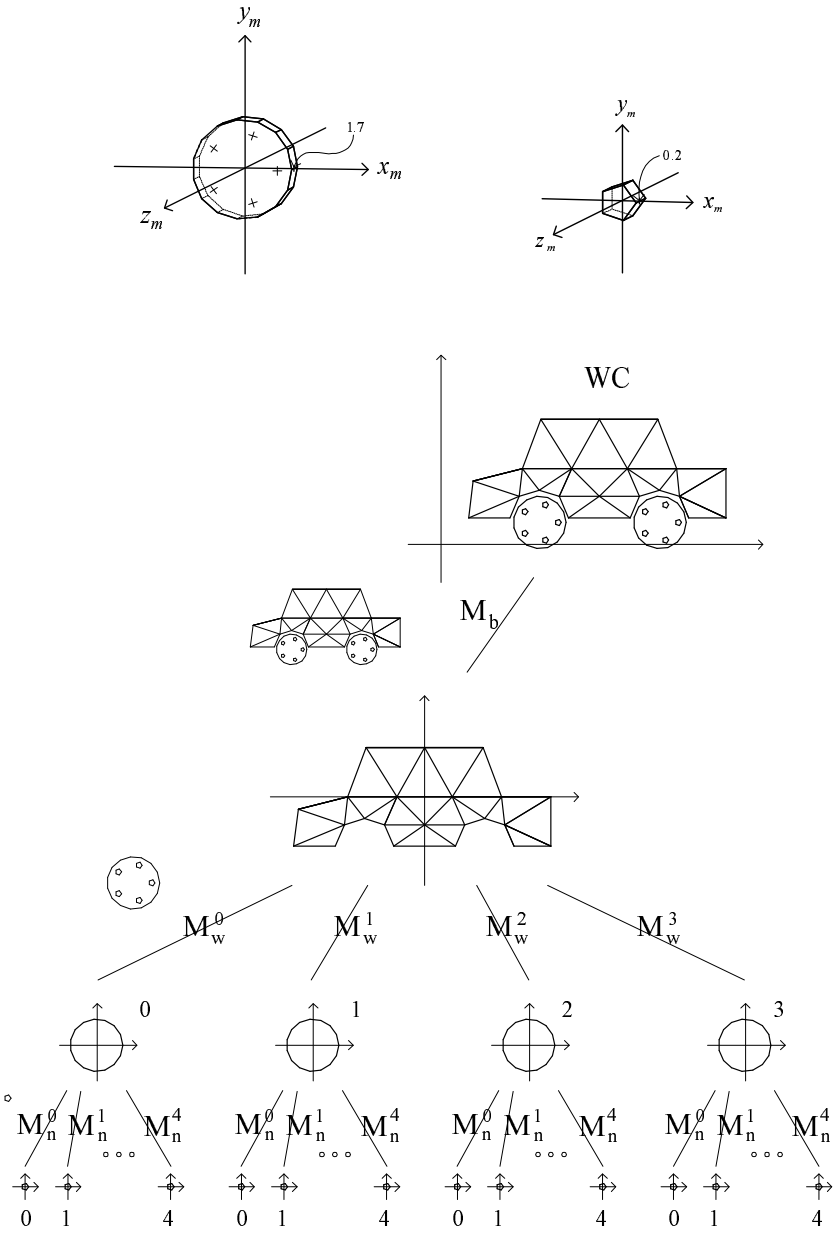

Figure 13: 자동차의 계층적 표현

직선 둘레로 반시계 방향으로 θ 각도만큼 회전시켜주는 4행 4열 아핀 변환 행렬 R을 기술하라.

(l) 다음 행렬 M_V가 OpenGL 렌더링 시스템의 '카메라의 위치와 방향'을 설정해주는 뷰잉 변환 행렬이라 하면, 이때 카메라는 정확히 '어느 지점'에서 '어느 방향'을 바라보고 있는 상태를 의미할까?

$$M_V = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & -1 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(m) 3차원 공간에서의 이동 변환, 크기 변환, 그리고 회전 변환에 대한 4행 4열 행렬 $T(t_x, t_y, t_z)$, $S(s_x, s_y, s_z)$, 그리고 $R(\theta, n_x, n_y, n_z)$를 고려하자. 위 행렬 M_V를 위의 기본 변환 행렬의 곱으로 표현하라.

서강대학교

```
float rotation_angle_car = 0.0f;
#define TO_RADIAN 0.01745329252f
#define rad 1.7f
#define ww 1.0f

void draw_wheel_and_nut(void) {
    int i;

    glUniform3f(loc_primitive_color, 0.000f, 0.808f, 0.820f); // color name: DarkTurquoise
    draw_geom_obj(GEOM_OBJ_ID_CAR_WHEEL); // draw wheel

    for (i = 0; i < 5; i++) {
        ModelMatrix_CAR_NUT = glm::rotate( (A), (B), glm::vec3( (C), (D), (E) ));
        ModelMatrix_CAR_NUT = glm::translate(ModelMatrix_CAR_NUT, glm::vec3(rad - 0.5f, 0.0f, ww));
        ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_NUT;
        glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);

        glUniform3f(loc_primitive_color, 0.690f, 0.769f, 0.871f); // color name: LightSteelBlue
        draw_geom_obj(GEOM_OBJ_ID_CAR_NUT); // draw i-th nut
    }
}

void draw_car_dummy(void) {
    glUniform3f(loc_primitive_color, 0.498f, 1.000f, 0.831f); // color name: Aquamarine
    draw_geom_obj(GEOM_OBJ_ID_CAR_BODY); // draw body
    glLineWidth(2.0f);
    draw_axes(); // draw MC axes of body
    glLineWidth(1.0f);

    ModelMatrix_CAR_DRIVER = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(-3.0f, 0.5f, 2.5f));
    ModelMatrix_CAR_DRIVER = glm::rotate(ModelMatrix_CAR_DRIVER, TO_RADIAN*90.0f, glm::vec3(0.0f, 1.0f, 0.0f));
    ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_DRIVER;
    glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
    glLineWidth(5.0f);
    draw_axes(); // draw camera frame at driver seat
    glLineWidth(1.0f);

    ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(-3.9f, -3.5f, 4.5f));
    ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
    glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
    draw_wheel_and_nut();  // draw wheel 0

    ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(3.9f, -3.5f, 4.5f));
    ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
    glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
    draw_wheel_and_nut();  // draw wheel 1

    ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(-3.9f, -3.5f, -4.5f));
    ModelMatrix_CAR_WHEEL = glm::scale(ModelMatrix_CAR_WHEEL, glm::vec3(1.0f, 1.0f, -1.0f));
    ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
    glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
    draw_wheel_and_nut();  // draw wheel 2

    ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(3.9f, -3.5f, -4.5f));
    ModelMatrix_CAR_WHEEL = glm::scale(ModelMatrix_CAR_WHEEL, glm::vec3(1.0f, 1.0f, -1.0f));
    ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
    glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
    draw_wheel_and_nut();  // draw wheel 3
}
void display(void) { // Display callback
    glm::mat4 ModelMatrix_big_cow, ModelMatrix_small_cow;
    glm::mat4 ModelMatrix_big_box, ModelMatrix_small_box;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    ModelViewProjectionMatrix = glm::scale(ViewProjectionMatrix, glm::vec3(5.0f, 5.0f, 5.0f));
    glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
    glLineWidth(2.0f);
    draw_axes();
    glLineWidth(1.0f);

    ModelMatrix_CAR_BODY = glm::rotate(glm::mat4(1.0f), -rotation_angle_car, glm::vec3(0.0f, 1.0f, 0.0f));
    ModelMatrix_CAR_BODY = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(20.0f, 4.89f, 0.0f));
    ModelMatrix_CAR_BODY = glm::rotate(ModelMatrix_CAR_BODY, 90.0f*TO_RADIAN, glm::vec3(0.0f, 1.0f, 0.0f));
    ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_BODY;
    glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
    draw_car_dummy();

    glutSwapBuffers();
}

```