

[CSE4170: 기초 컴퓨터 그래픽스]

중간고사

(담당교수: 임 인 성)

2018년 4월 27일 (금요일)

- 답은 연습지가 아니라 답안지에 기술할 것. 답안지 공간이 부족할 경우, 답안지 뒷면에 기술하고, 해당 답안지 칸에 그 사실을 명기할 것. 문제지와 연습지는 수거하지 않음.
 1. 그럼 1에 도시한 바와 같이 왼쪽의 윈도우의 내용을 오른쪽 윈도우 안으로 매핑을 해주는 2차원 아핀 변환에 대한 3행 3열 행렬 M 을 이동 변환 $T(t_x, t_y)$, 크기 변환 $S(s_x, s_y)$, 그리고 회전 변환 $R(\theta)$ 등의 기본 아핀 변환의 합성을 통하여 구하라. (답안 작성 시 (i) 합성 과정을 반드시 기술한 후, (ii) 최종 결과 행렬을 기술할 것)

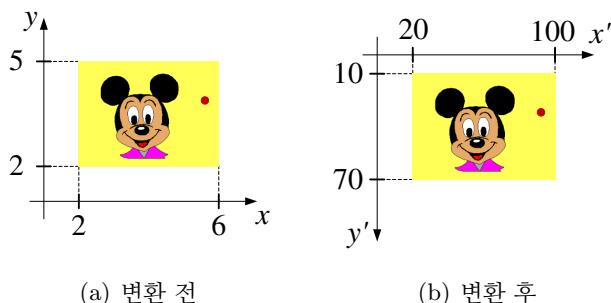


Figure 1: 2차원 윈도우 매핑

- 그림 2에는 2차원 공간의 두 점 $(5, 0)$ 과 $(0, 5)$ 를 지나는 직선이 주어져 있다. 이제 임의의 한 점 (x, y) 를 이 직선 둘레로 반사(reflection)를 시키되 직선 까지의 거리가 두 배가 되는 방식으로 변환을 하려 한다. 이러한 반사 변환에 해당하는 3행 3열 아핀 변환 행렬 M 을 $T(t_x, t_y)$, $S(s_x, s_y)$, 그리고 $R(\theta)$ 등의 기본 아핀 변환을 통하여 합성하라. (답안 작성 시 먼저 M 을 위의 기본 변환 행렬의 곱으로 표현한 후, 각 3행 3열의 행렬의 9개의 원소값들을 정확히 기술할 것. 또한 편의상 $\frac{1}{\sqrt{2}}$ 값은 α 로 표기할 것)
 - 2차원 아핀 변환인 이동변환 $T(t_x, t_y)$, 크기변환 $S(s_x, s_y)$, 그리고 회전변환 $R(\theta)$ 에 대한 3행 3열

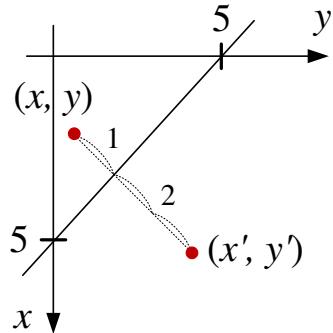


Figure 2: 변형된 2차원 반사

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, i , 그리고 i 값은 무엇인가?

(d) 위 문제의 아핀 변환 행렬 M_{2D} 의 9개의 원소 값을 정확히 기술하라.

4. 그림 3은 3차원 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. 여기서 $\text{PRP} = (e_x e_y e_z)^t$, $u = (u_x u_y u_z)^t$, $v = (v_x v_y v_z)^t$, 그리고 $n = (n_x n_y n_z)^t$ 이고, 내적 연산 \circ 에 대해 $u \circ u = v \circ v = n \circ n = 1$ 이고 $u \circ v = v \circ n = n \circ u = 0$ 이며, 이 때 각 기하 데이터의 의미는 수업시간에 배운 것을 가정한다.

(a) 이 두 점의 좌표간의 관계를 표현해주는 아래의 식에서 필요한 3행 3열 행렬의 원소들을

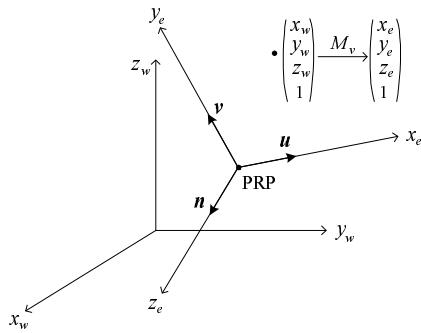


Figure 3: OpenGL 시스템에서의 뷰잉 변환

기술하라. (즉 b_{11} 부터 b_{33} 까지)

$$\begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} = \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix} + \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \end{pmatrix}$$

- (b) 위 그림이 나타내는 뷰잉 변환 행렬 M_v 를 고려하자.

$$M_v = \begin{bmatrix} a_{11} & a_{12} & a_{13} & c_1 \\ a_{21} & a_{22} & a_{23} & c_2 \\ a_{31} & a_{32} & a_{33} & c_3 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

카메라 시점에서 세상을 바라볼 때, 왼쪽에 해당하는 방향 $v_l = (x_l \ y_l \ z_l)^t$ 에 대한 세상 좌표계에서 좌표값을 행렬 M_v 의 원소 값들을 사용하여 표현하라.

- (c) 행렬 M_v 의 원소들 중 c_3 을 PRP , u , v , 그리고 n 들의 좌표값을 사용하여 표현하라. (필요할 경우 내적 연산 \circ 를 사용할 것)
- (d) 다음과 같은 glm 함수를 호출할 때 생성되는 뷰잉 변환 행렬 M_v 의 원소들을 기술하라. (주의: 편의상 이 코드에서 `glm::`은 생략함)
- ```
lookAt(vec3(-10.0f, 0.0f, 0.0f),
 vec3(-10.0f, 10.0f, 0.0f),
 vec3(1.0f, 0.0f, 0.0f));
```
5. 그림 4에는 2차원 세상 좌표계의 원점 근처에 모델링 좌표계에서 정의된 모자 물체가 도시되어 있다. 이때 이 그림의 내용처럼 점 (300, 100)을 중심으로 반지름이 200인 원을 따라 모자를 그리려 한다. 이를 다음과 같은 코드로 구현하려 할 때, 변수 a부터 h에 들어갈 값 또는 수식을 C/C++ 언어 문법에 맞게 정확히 기술하라. (주의: 편의상 이 코드에서 `glm::`은 생략함)

```
for (int i = 0; i < 9; i++) {
 ModelMatrix = translate(mat4(1.0f),
```

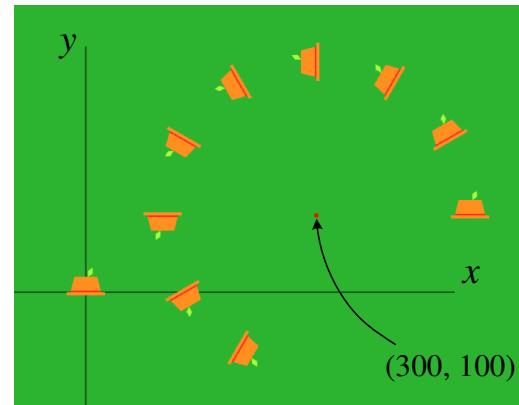


Figure 4: 2차원 모델링 변환

```
vec3(a, b, 0.0f));
ModelMatrix = rotate(ModelMatrix,
 c*TORADIAN, vec3(d, e, f));
ModelMatrix = translate(ModelMatrix,
 vec3(g, h, 0.0f));
ModelViewProjectionMatrix
= ViewProjectionMatrix * ModelMatrix;
glUniformMatrix4fv(loc_ModelViewProjectionMatrix,
1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
draw_hat();
}
```

6. 그림 5에는 그림 6에 도시한 바와 같이 3차원 세상 좌표계의  $(-6, 0, -4)$  지점에서  $(6, 0, 4)$  지점까지의 직선 경로를 따라 균일한 속도로 이동 및 회전을 하는 소를 렌더링 해주는 프로그램의 일부가 주어져 있다. 여기서 소 물체는 자신의 모델링 좌표계의  $x_m$ 축 방향을 바라보고 있고, 같은  $y_m$ 축 방향을 향하고 있는데,  $y_m$ 축 둘레로 총 180도 회전을 하면서 이동하고 있다. 또한 소의 모델링 좌표계의 원점에 해당하는 점은 위의 두 점간의 직선 상에서 이동을 하고 있다. 이 코드가 올바르게 작동하기 위하여  $\alpha_0$ 에서  $\alpha_6$ 까지 7개의 지점에 들어갈 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라. (참고: 변수 `ViewMatrix`와 `ProjectionMatrix`에는 그 이름이 의미하는 행렬이 이미 계산되어 지정되어 있음)
7. 그림 7은 한 변의 길이가 1이고 서로 수직인 세 선분에 의해 정의되는 두 프레임 Frame A와 Frame B를 도시하고 있는데, 지금 Frame A를 각 점이 아래와 같이 매핑이 되도록 Frame B로 변환하려 한다.

- $(0, 0, -2) \rightarrow (0, 0, 2)$
- $(0, -1, 0) \rightarrow (1, 0, 0)$

```

for (int i = 0; i <= 180; i += 30) {
 float angle = (float)i;
 ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(alpha_0, alpha_1, alpha_2));
 ModelViewMatrix = glm::rotate(ModelViewMatrix, (alpha_3)*TO_RADIAN,
 glm::vec3(alpha_4, alpha_5, alpha_6));
 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);
}

```

Figure 5: 3차원 모델링 변환 코드

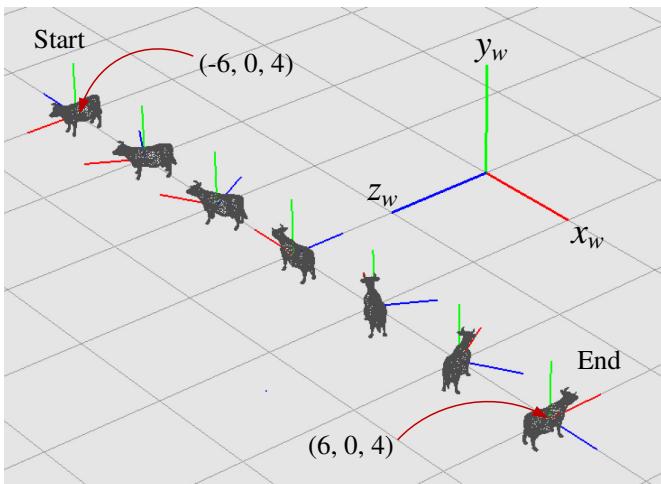


Figure 6: 간단한 모델링 변환

- $(\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}}) \rightarrow (0, 0, -1)$
- $p \rightarrow (0, 1, 0)$

(주의: 여기서 두 Frame의 원점의 좌표  $(0, 0, -2)$  와  $(0, 0, 2)$ 만 점의 좌표이고  $p$ 를 포함 나머지 좌표는 벡터를 나타냄)

- 이때 Frame A의  $p$ 의 좌표  $(x, y, z)$ 를 기술하라.
  - 이때 Frame A를 Frame B로 변환 해주는 아핀 변환에 해당하는 4행 4열 행렬  $M$ 을  $M = ABC$ 와 같이 세 개의 행렬의 곱으로 표현하여 할 때, 각 4행 4열 행렬의 16개 원소들을 정확히 기술하라. (주의:  $A$ 와  $C$ 는 동일한 부류의 기하 변환을 나타내야 함)
8. 다음은 카메라의 설정에 관한 문제이다. 그림 8에는 수업 시간에 배운 `glm::lookAt()` 함수에 대한 구

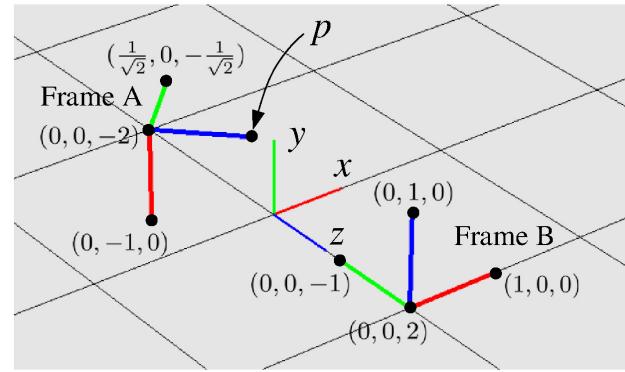


Figure 7: 두 프레임간의 변환

현 코드 (이하 “이 함수”)가 주어져 있고, 그림 9에는 OpenGL Compatibility Profile에서 제공하는 `gluLookAt(*)` 함수의 호출에 대하여 뷔잉 변환 행렬을 계산하는 과정 (이하 “이 그림”)이 주어져 있는데, 이들을 참조하여 답하라. 이 그림의 세 벡터  $u$ ,  $v$ , 그리고  $n$ 은 내적 연산  $\circ$ 에 대해  $u \circ u = v \circ v = n \circ n = 1$ 이고  $u \circ v = v \circ n = n \circ u = 0$ 을 만족하며, 이때 각 기하 데이터의 의미는 수업시간에 배운 것을 가정한다.

- 이 그림에서 이 함수의 인자 중의 하나인 3차원 벡터 `center`에 해당하는 기호의 이름을 기술하라.
- 이 함수의 Line (a) 문장 수행 후 벡터 `f`에 저장되는 값을 위의 벡터들을 사용하여 정확히 기술하라.
- 문맥상 이 함수의 Line (c) 문장 수행 직후 `u.y`에 저장이 되는 값은 무엇일지 두 벡터 `s`와 `f`의 원소 `s.x, s.y, s.z`와 `f.x, f.y, f.z`을 사용하여 표현하라.

```

namespace glm {
 ...
 template <typename T, precision P> GLM_FUNC_QUALIFIER tmat4x4<T, P>
 lookAt(tvec3<T, P> const &eye, tvec3<T, P> const ¢er, 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; // Line (e)
 Result[0][1] = u.x; Result[1][1] = u.y; Result[2][1] = u.z; // Line (f)
 Result[0][2] = f.x; Result[1][2] = f.y; Result[2][2] = f.z; // Line (g)
 Result[3][0] = -dot(s, eye); // Line (h)
 Result[3][1] = ; // Line (i)
 Result[3][2] = ; // Line (j)
 return Result;
 }
}

```

Figure 8: glm::lookAt() 함수의 구현

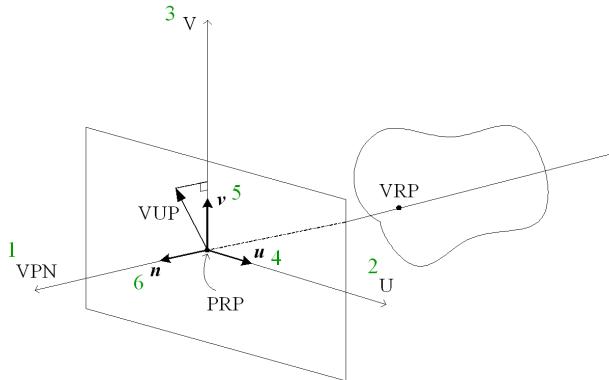


Figure 9: gluLookAt(\*) 함수를 통한 카메라의 설정

- (d) 이 함수의 Line (d) 문장 수행 결과 4행 4열 행렬 *Result*에는 어떤 값이 저장이 될까?
- (e) 현재 이 함수에는 심각한 오류가 내포되어 있다. 어떤 문장의 어떤 부분을 수정해야하는지, 해당 문장 기호(예를 들어, Line (c))를 명시하고 오류를 수정하라.
- (f) 문맥상 이 함수의 Line (i)와 Line (j)의 빈곳에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라.
9. 다음과 같은 4행 4열 행렬  $M$ 에 의해 정의되는 3

차원 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) 그림 10의 왼쪽의 직각 프레임을 오른쪽의 직각 프레임으로 회전시켜주는 4행 4열 행렬  $M$ 의 왼쪽-위쪽 3행 3열 부행렬에 해당하는 부분의  $a_{11}$ 에서  $a_{33}$ 까지의 9개의 원소값을 정확히 기술하라.

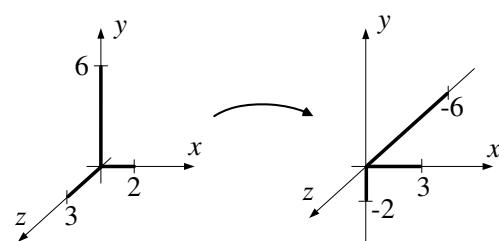


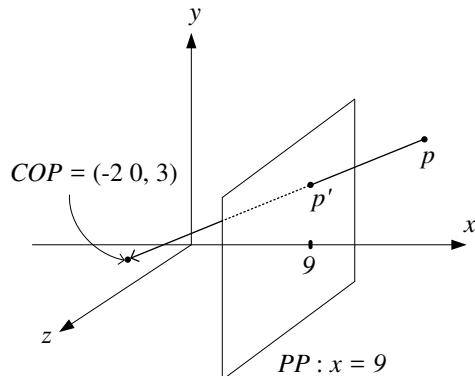
Figure 10: 3차원 공간에서의 회전 변환

- (b) 바로 위 문제에서의 회전 변환은  $(n_x, 1, n_z)$  벡터가 가리키는 직선 둘레로  $\alpha$ 도만큼 회전시켜주는 변환에 해당한다. 과연  $\alpha$ 가 몇 도인지 정확히 기술하라. (참고: 그림 11에 주어진 회전 변환 행렬을 참조하고, 각도는 0도와 180도 사이의 각으로 기술할 것)
- (c) 바로 위 문제에서의  $n_x$ 와  $n_z$  값을 기술하라.

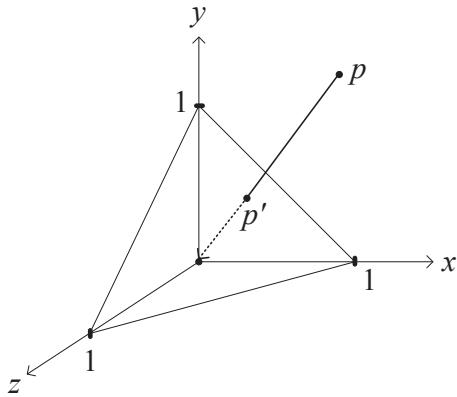
$$R(\alpha, n_x, n_y, n_z) = \begin{bmatrix} \bar{n}_x^2(1 - c) + c & \bar{n}_x\bar{n}_y(1 - c) - \bar{n}_z s & \bar{n}_x\bar{n}_z(1 - c) + \bar{n}_y s & 0 \\ \bar{n}_x\bar{n}_y(1 - c) + \bar{n}_z s & \bar{n}_y^2(1 - c) + c & \bar{n}_y\bar{n}_z(1 - c) - \bar{n}_x s & 0 \\ \bar{n}_x\bar{n}_z(1 - c) -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

Figure 11: 회전 변환 행렬

10. 다음은 원근 투영 변환에 관한 문제이다. 아래의 그림을 보면서 각 문항에서 요구하는 4행 4열 행렬을 기술하라. (참고: 이 행렬들은 3차원 공간에서 2차원 공간으로 변환을 나타냄)



(a) 문제 (a) 그림



(b) 문제 (b) 그림

Figure 12: 원근 투영 변환

- (a) 그림 12(a)에서와 같이 COP(Center of Projection)가  $(-2, 0, 3)$ 이고 PP(Projection Plane)이  $x = 9$ 인 상황에서, 주어진 점  $p = (x \ y \ z)^t$ 을  $p' = (x' \ y' \ z' \ 1)^t$ 로 변환해주는 4행 4열의 원근 투영 변환 행렬  $M_1$ 을 기술하라.
- (b) 그림 12(b)는 임의의 점  $p$ 를 이 점과 원점을 지나는 직선과 세 점  $(1, 0, 0)$ ,  $(0, 1, 0)$ , 그리고  $(0, 0, 1)$ 을 지나는 평면과의 교점  $p'$ 로 투영해주는 모습을 도시하고 있다. 이때 이에

대한 원근 투영 변환 행렬  $M_2$ 를 기술하라.

11. 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 13(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. 이 때 (B), (D), 그리고 (H)에 들어가 들어갈 내용을 이 함수의 문법에 맞게 정확히 기술하라.
- (b) (바로 위 문제에 이어) (I) 와 (J)에 들어갈 내용을 이 함수의 문법에 맞게 정확히 기술하라.
- (c) 사용자가 왼쪽 마우스 버튼을 누른 후 움직일 때 수행이 되는 모션 컬백 함수 내에서 마우스가 위-아래로 움직인 양 `delta_y`에 대해 다음과 같이 그림 13(b)에 정의된 함수를 호출하려 한다. (이 변수는 마우스가 화면에서 위쪽 방향으로 움직일 때 양수 값을 가지고, 반대로 아래로 움직이면 음수 값을 가짐)

```
cam_move_1(delta_y, camera.vaxis);
set_ViewMatrix_from_camera_frame();
```

지금 마우스가 화면의 위쪽에서 아래쪽으로 움직이고 있는 동안 지속적으로 이 함수가 호출될 경우 카메라는 어떻게 움직일지 정확히 기술하라.

```

void set_ViewMatrix_from_camera_frame(void) {
 ViewMatrix = glm::mat4(1.0f);

 ViewMatrix[0].x = (A);
 ViewMatrix[0].y = (B);
 ViewMatrix[0].z = (C);

 ViewMatrix[1].x = (D);
 ViewMatrix[1].y = camera.vaxis.y;
 ViewMatrix[1].z = (E);

 ViewMatrix[2].x = (F);
 ViewMatrix[2].y = (G);
 ViewMatrix[2].z = (H);

 ViewMatrix = glm::translate((I), (J));
}

```

(a) 뷰잉 변환 행렬의 설정

```

void cam_move_1(float del, glm::vec3 axis) {
 camera.pos += del*axis;
}

```

(b) 카메라의 조작 1

```

void cam_move_2(float angle, glm::vec3 axis) {
 glm::mat3 Matrix;

 Matrix = glm::mat3(glm::rotate(glm::mat4(1.0f),
 TO_RADIAN*angle, axis)); // Line (a)

 camera.uaxis = Matrix * camera.uaxis; // Line (b)
 camera.vaxis = Matrix * camera.vaxis; // Line (c)
 camera.naxis = Matrix * camera.naxis; // Line (d)
}

```

(c) 카메라의 조작 2

Figure 13: 카메라의 조작

- (d) 사용자가 왼쪽 마우스 버튼을 누른 후 움직일 때 수행이 되는 모션 컬백 함수 내에서 마우스가 좌-우로 움직인 양 `delta_x`에 대해 다음과 같이 그림 13(c)에 정의된 함수를 호출하려 한다. (이 변수는 마우스가 화면에서 오른쪽 방향으로 움직일 때 양수 값을 가지고, 반대로 왼쪽으로 움직이면 음수 값을 가짐)

```
cam_move_2(delta_x, camera.vaxis);
set_ViewMatrix_from_camera_frame();
```

지금 마우스가 화면의 왼쪽에서 오른쪽으로 움직이고 있는 동안 지속적으로 이 함수가 호출될 경우 카메라는 어떻게 움직일지 정확히 기술하라.

- (e) 바로 위 문제에서와 같이 `cam_move_2()` 함수를 호출할 경우 그림 13(c)에 정의된 함수의 문장 중 수행을 하지 않아도 무방한 문장이 있을 경우 그에 해당하는 문장 모두에 대하여 문장 기호(예를 들어, Line (a)와 같이)를 사용하여 기술하라. (모든 문장이 필요할 경우 “없음”으로 답할 것)

## 12. 다음 단답식 문제에 답하라.

- (a) 3차원 공간에서의 세 점  $(0, 0, 3)$ ,  $(1, 0, 3)$ 과  $(3, 3, 4)$ 에 의해 정의되는 삼각형의 면적은?
- (b) 3차원 투영 공간의 점  $(6.0, -6.0, -9.0, -3.0)$ 에 해당하는 아핀 공간의 점의 좌표  $(x, y, z)$ 를 기술하라.
- (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 투영 평면에서 유한 거리만큼 떨어진 투영 변환의 이름은 무엇인가?
- (e) OpenGL의 뷰잉 파이프라인에서 ‘카메라의 위치와 방향을 설정’해주는 변환은 정확히 어느 좌표계에서 어느 좌표계로 보내주는 변환 인지, 수업 시간에 배운 좌표계 이름 CC, EC, MC, NDC, WC, 그리고 WdC (Window Coordinate)을 사용하여 기술하라.
- (f) OpenGL의 뷰잉 파이프라인에서 촬영한 필름을 현상한 후 인화지에 확대/인화하는 과정은 정확히 어느 좌표계에서 어느 좌표계로 보내주는 과정에 해당하는지 위 문제의 좌표계 이름을 사용하여 기술하라.
- (g) 다음과 같은 4행 4열 아핀 변환 행렬  $M$ 을  $M = R \cdot T$ 와 같이 회전 변환 행렬  $R$ 과 이동

변환 행렬  $T$ 의 곱으로 표현할 때,  $R$ 과  $T$ 의 내용을 기술하라.

$$M = \begin{bmatrix} 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 5 \\ 1 & 0 & 0 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

- (h) 위 문제의 행렬  $M$ 의 역행렬  $M^{-1}$ 을 기술하라. (16개 원소의 값들을 명기할 것)

- (i) 3차원 공간의 점을 위 문제의 아핀 변환  $M$ 을 통하여 기하 변환 할 때, ‘방향과 크기’를 가지는 임의의 벡터  $n = (n_x \ n_y \ n_z)^t$ 가  $M$ 에 의해 어떻게 변환이 될지 그 변환 후의 벡터  $n' = (n'_x \ n'_y \ n'_z)^t$ 의 좌표값을 기술하라.

13. 실시간 3D 렌더링 과정을 구성하는 다음의 단계들에 대하여 고려하자.

- Primitive processing (PP)
- Rasterization (R)
- Primitive assembly (PA)
- Fragment shading (FS)
- Vertex shading (VS)
- Raster operations (RO)

- (a) 위의 6개의 단계들을 보편적인 렌더링 과정에서 수행이 되는 순서대로 나열하라. (각 단계를 기술할 때 해당 팔호 안의 기호를 사용할 것)
- (b) 뷰 볼륨 절단(view volume clipping)이 수행되는 단계의 기호를 기술하라.
- (c) 위의 6개의 단계들 중 연속 공간의 정보들이 이산 공간의 정보로 변환이 되는 과정과 가장 관련이 있는 단계의 기호를 기술하라.
- (d) 은면 제거 계산(hidden surface elimination) 계산과 가장 관련이 있는 단계의 기호를 기술하라.
- (e) 카메라의 위치와 방향을 설정하는 과정이 직접적으로 영향을 미치는 단계의 기호를 기술하라.
- (f) 모델링 변환이 수행이 되는 과정의 기호를 기술하라.

# 서강대학교

[CSE4170: 기초 컴퓨터 그래픽스]

## 중간고사

(담당교수: 임인성)

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 만족시켜 주는 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  $x, y, z$ 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 원도우의 내용을 오른쪽 원도우 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.

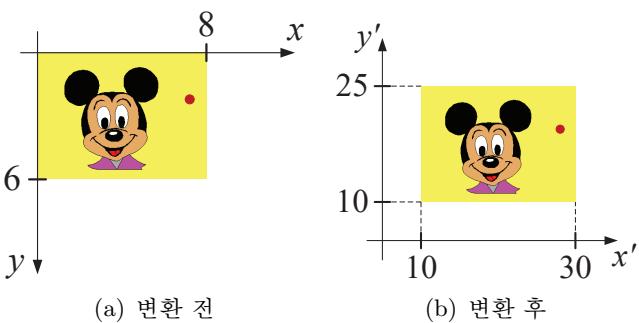


Figure 1: 2차원 원도우 매핑

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  $e, f, g, h, i$ , 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  $\theta$ 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 답할것)
  - (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_RADIANS, 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  $\theta$  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 강체 변환일 경우 벡터  $a_2 = (a_{12} \ a_{22} \ a_{32})^t$ 의 세 원소들간에 성립해야할 조건을 기술하라.
- (c)  $M$ 이 강체 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 할지 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 강체 변환일 경우  $L$  행렬의 내용을  $M$ 의 원소들을 사용하여 기술하라.
- (e) 만약  $M$ 이  $x, y, z$ 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.

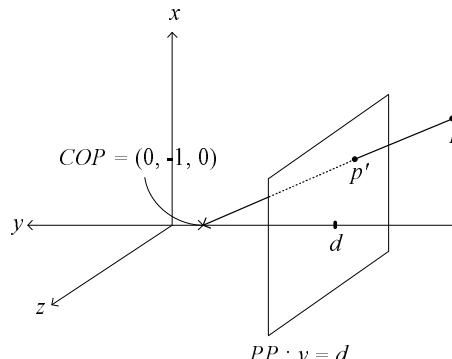


Figure 3: 원근 투영 변환

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

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

Figure 2: 회전 변환 행렬

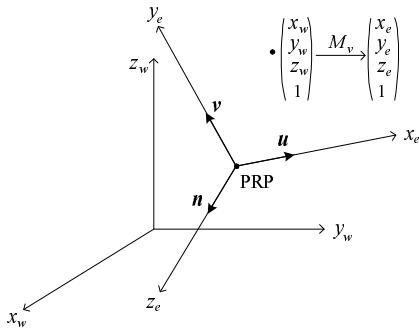


Figure 4: OpenGL 시스템에서의 뷰잉 변환

- (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 깊이에 대한 정보를 얻을 수 있는데, COP 점 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.

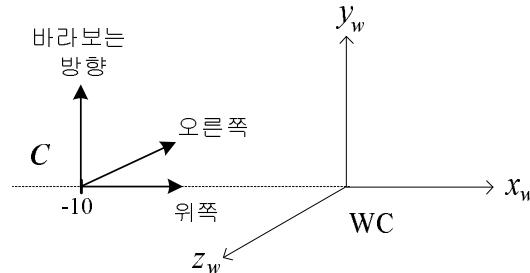


Figure 5: 카메라의 위치와 방향 설정

- (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 설정해 주는 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 프레임 간의 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

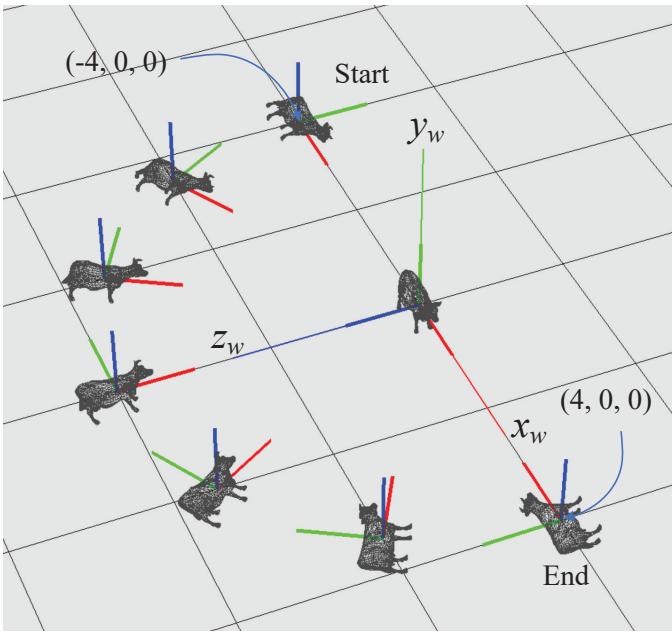


Figure 7: 3차원 모델링 변환

- (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.

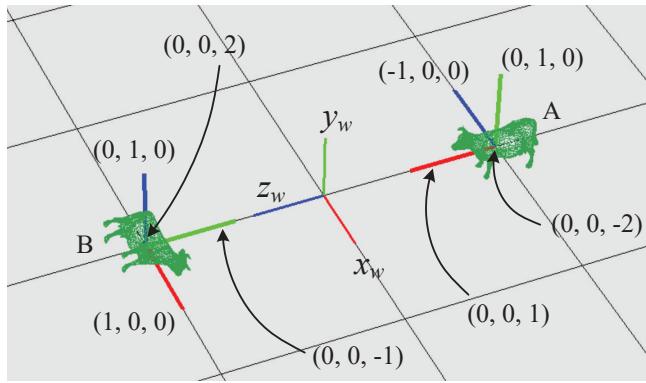


Figure 8: 두 프레임 간의 변환

- (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 구해주는 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

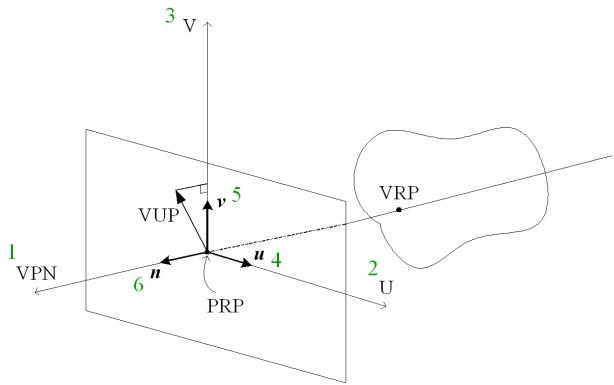


Figure 9: 카메라의 위치와 방향 설정

라보는 반대 방향에 대한 단위 벡터 (unit vector)들이다. 이 함수의 Line (a) 문장 수행 후 3차원 벡터 f에 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 대치하여도 올바르게 작동을 하는데, 과연 그 문장이 무엇인가? 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 뷰 임 변환 행렬 `ViewMatrix`를 설정하고 있다. 이 때 (A), (E), 그리고 (G)에 들어갈 내용을 이 함수의 문법에 맞게 정확히 기술하라.
- (b) (바로 위 문제에 이어) (I)에 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 움직일지 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 2.0f만큼 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 할지 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*T0_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

[CSE4170: 기초 컴퓨터 그래픽스]

## 중간고사

(담당교수: 임인성)

- 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.

- $R(\theta) \cdot S(-1, 1) = S(-1, 1) \cdot R(\alpha)$  식을 만족시켜주는  $\alpha$  값은 무엇인가?
- $T(t_x, t_y) \cdot S(-1, 1) = S(-1, 1) \cdot T(\beta, \gamma)$  식을 만족시켜주는  $\beta$ 와  $\gamma$  값은 무엇인가?
- $R(\theta) \cdot T(t_x, t_y) = T(\delta, \epsilon) \cdot R(\theta)$  식을 만족시켜주는  $\delta$ 와  $\epsilon$  값은 무엇인가?

2. (1번 문제를 고려하면서) 다음 질문에 답하라.

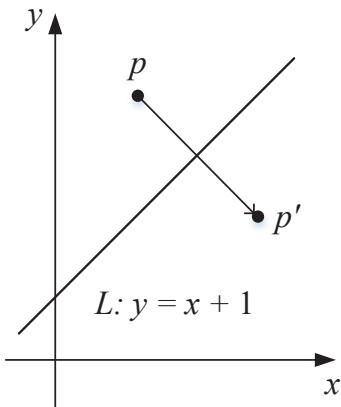


Figure 1: 직선에 대한 반사

- 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  $M$ 을  $M = N \cdot T(0, -1)$ 과 같이 표현한다고 할 때, 이때의 3행 3열 아핀변환 행렬  $N$ 을  $T(t_x, t_y), S(s_x, s_y)$ , 그리고  $R(\theta)$  등의 기본 아핀변환을 통하여 합성을 하라.

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

- 그림 2에서와 같이 왼쪽의 윈도우의 내용을 오른쪽 윈도우 안으로 매핑을 해주는 2차원 아핀변환에 대한 3행 3열 행렬  $M$ 을  $T(t_x, t_y), S(s_x, s_y)$ , 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).

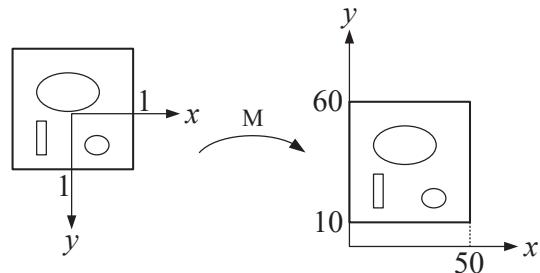


Figure 2: 2차원 윈도우 매핑 변환

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

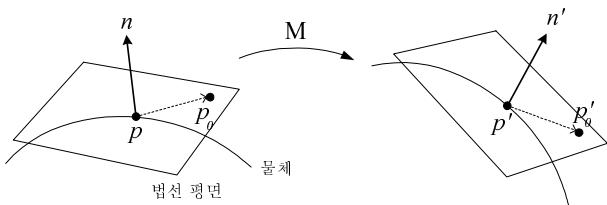


Figure 3: 법선 벡터의 변환

- 이 아핀변환을 통하여 얻은 벡터  $n' = (n'_x \ n'_y \ n'_z \ 0)^t$ 을 어떤 4행 4열 행렬  $N$ 에

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

그림 3에서와 같이 주어진 점  $p = (x \ y \ z \ 1)^t$ 에서의 법선 벡터가  $n$ 이 라고 하자. 이때 점  $p$ 에서의 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  $n$ 이 변환 행렬  $M$ 에 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  $p$ 와  $p_0$ 를 구해 위의 관계식에 대입하여 다음과 같은 식을 얻게 된다. ...

- (b) 위 문제의 아핀변환  $M$ 이 강체변환(rigid-body transformation)이라고 하자. 이때  $M$ 을 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  $M_{3 \times 3}$ 의 세 개의 열벡터를 각각  $a_1, a_2, a_3$ 라 할 때, 이 세 개의 벡터가 만족하는 수학적인 성질을 벡터의 내적을 사용하여 정확히 기술하라.

- (c) 문제 (b)의 조건 하에, 문제 (a)의 행렬  $N$ 의 왼쪽-위쪽의 3행 3열 부행렬  $N_{3 \times 3}$ 의 내용을 정확히 기술하라.

## 5. 다음은 두 프레임간의 변환에 관한 문제이다.

- (a) 그림 4의 A 소는 점  $(0, 0, -2)$ 를 원점으로 하는 자신의 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  $x_w$ 축과  $y_w$  방향과 일치하는 방식으로 세상 좌표계와 일치시켜주려 한다. 이때 필요한 4행 4열 행렬  $M_a$ 의 내용을 기술하라.
- (b) 한편 B 소는 점  $(0, 0, 2)$ 를 중심으로 하는 프레임을 기준으로 세상 좌표계에 존재하고 있는데, A 소의 각 꼭지점을 B 소의 대응되는 점으로 매핑해주는 4행 4열 행렬

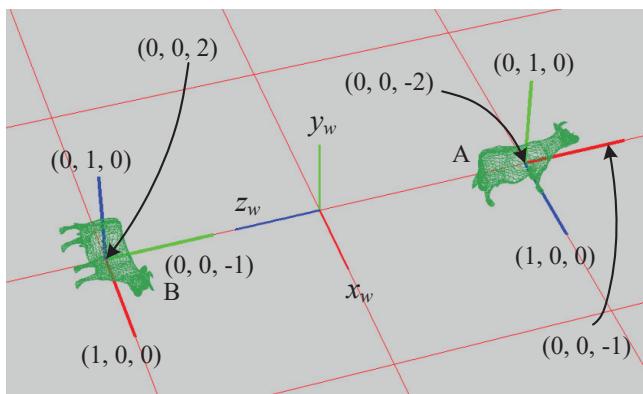


Figure 4: 두 프레임간의 변환

$M_b$ 를  $M_b = T_2 \cdot R \cdot T_1$ 와 같이 두 개의 이동변환 행렬  $T_1, T_2$ 와 한 개의 회전변환 행렬  $R$ 의 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  $\alpha$ 도만큼 회전시켜주는 회전변환에 해당한다. 이때  $n_y$ 와  $n_z$ , 그리고  $\alpha$  값을 기술하라. (참고: 그림 5에 주어진 회전변환 행렬을 참조하고, 각도는 0도와 180도 사이의 각으로 기술할 것)

## 6. 다음은 간단한 모델링 변환에 관한 문제이다.

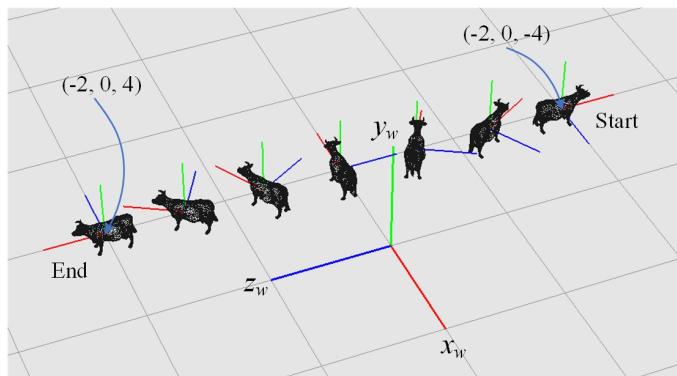


Figure 6: 간단한 모델링 변환

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  $x_m$ 축 방향을 바라보고 있고, 등은  $y_m$ 축 방향을 향하고 있는데,  $y_m$ 축 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, (alpha)*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)
}

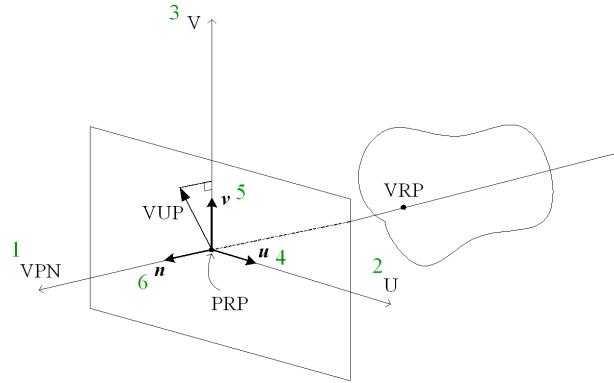
```

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  $t_x, t_y, t_z, \alpha, n_x, n_y, n_z$ 에 들어갈 값을 정확히 기술하라. (참고: 변수 ViewMatrix와 ProjectionMatrix에는 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 뷔잉변환 행렬을 계산하는 과정(이하 “이 그림”)이 주어져 있으며, 그림 9에는 `glm::lookAt()` 함수에 대한 구현 코드(이하 “이 함수”)가 주어져 있는데, 이들을 참조하여 답하라.

- (a) 이 그림에서 이 함수의 인자 중의 하나인 3차원 벡터 `eye`에 해당하는 기호의 이름을 기술하라.
- (b) 이 그림의  $u, v, n$  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)

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

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.
- 8. 다음은 원근투영 변환에 관한 문제이다.

그림 10에서와 같이 COP(Center of Projection)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 ¢er, 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

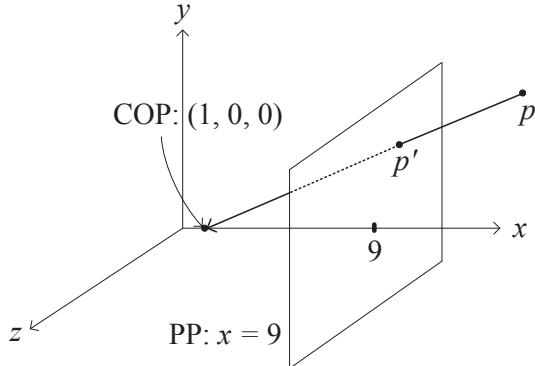


Figure 10: 원근 투영 변환

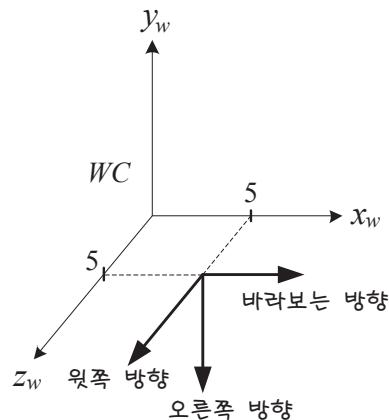


Figure 11: 카메라의 위치와 방향 설정

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

9. 그림 11에는 카메라의 위치와 방향을 설정해주는 카메라 프레임이 도시되어 있다. 이 경우 OpenGL 시스템에서 사용하는 4행 4열 뷰잉변환 행렬  $M_V$ 를 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  $z$ 축의 방향이 반대가 되도록 네 모서리를 마춰주어야 함)

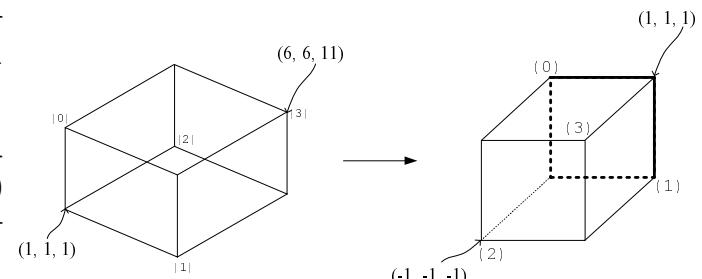


Figure 12: 직교 투영 변환

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 어떤 방식으로 텁색을 하고 있는가? 자료 구조 시간에 배운 용어를 사용할 것.
- (b) 프로그램 맥락상 13번 문장의 (A)에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라.
- (c) 프로그램 맥락상 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_RADIANS`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 번째 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  $z$ 축 각각이  $u$ ,  $v$ , 그리고  $n$ 축에 대응함을 상기할 것). 해당 부분의 코드를 볼 때 이 자동차의 앞쪽 방향은 자신의 모델링 좌표계를 기준으로 어느 축 방향을 향하고 있을까? “양의  $x$ 축” 또는 “음의  $x$ 축”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. 이

때 한 점  $q$ 가 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 변환이다?
- (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

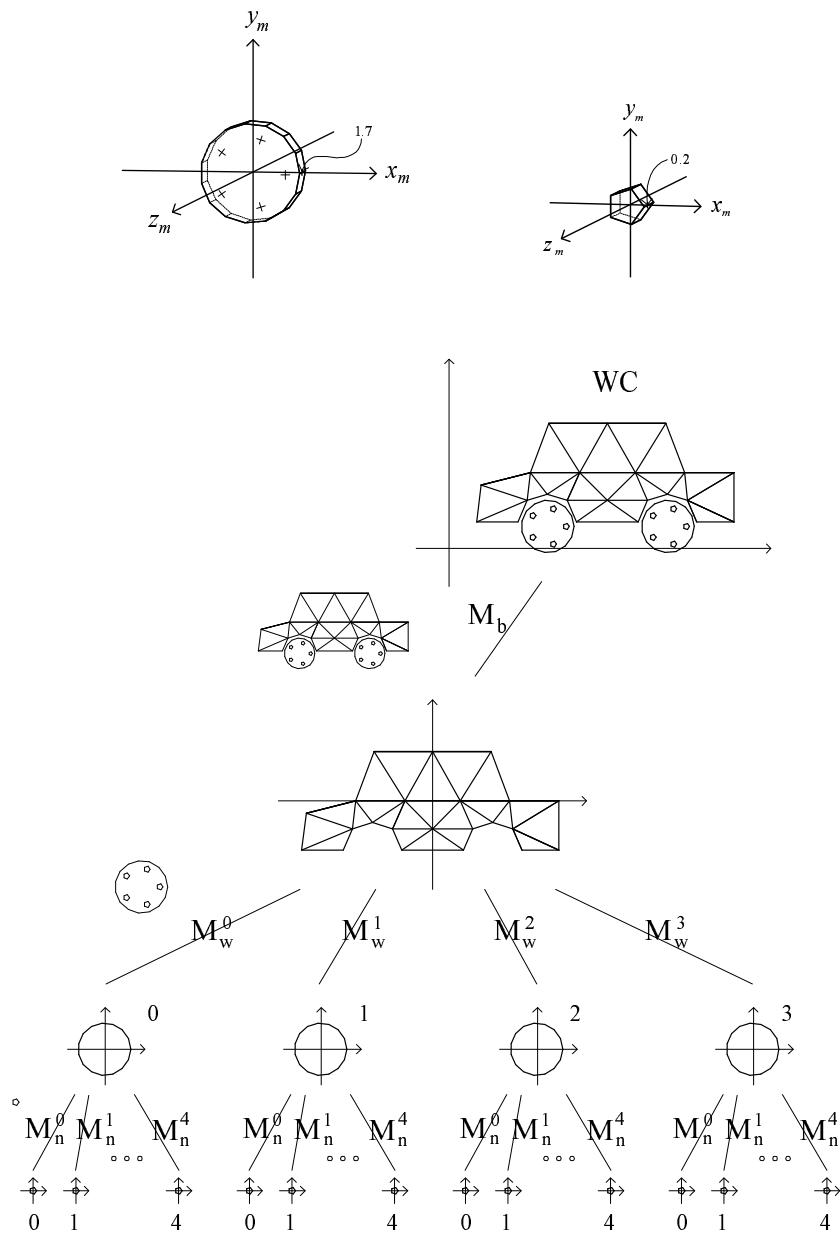


Figure 13: 자동차의 계층적 표현

직선 둘레로 반시계 방향으로  $\theta$  각도만큼 회전시켜주는 4행 4열 아핀 변환 행렬  $R$ 을 기술하라.

- (l) 다음 행렬  $M_V$ 가 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, 크기 변환, 그리고 회전 변환에 대한 4행 4열 행렬  $T(t_x, t_y, t_z)$ ,  $S(s_x, s_y, s_z)$ , 그리고  $R(\theta, n_x, n_y, n_z)$ 을 고려하자. 위 행렬  $M_V$ 를 위의 기본 변환 행렬의 곱으로 표현하라.

# 서강대학교

```

1 float rotation_angle_car = 0.0f;
2 #define TO_RADIAN 0.01745329252f
3 #define RAD 1.7f
4 #define WW 1.0f
5
6 void draw_wheel_L_and_nut(void) {
7 int i;
8
9 ModelMatrix_CAR_NUT = glm::rotate((A), (B), glm::vec3((C) , (D) , (E)));
10 ModelMatrix_CAR_NUT = glm::translate(ModelMatrix_CAR_NUT, glm::vec3(rad - 0.5f, 0.0f, WW)) ;
11
12 for (i = 0; i < 5; i++) {
13 ModelMatrix_CAR_NUT = glm::rotate((A), (B), glm::vec3((C) , (D) , (E)));
14 ModelMatrix_CAR_NUT = glm::translate(ModelMatrix_CAR_NUT, glm::vec3(rad - 0.5f, 0.0f, WW)) ;
15
16 ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_NUT;
17 glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
18
19 glUniform3f(loc_primitive_color, 0.690f, 0.769f, 0.871f); // color name: LightSteelBlue
20 draw_geom_obj(GEOM_OBJ_ID_CAR_NUT); // draw i-th nut
21 }
22
23 void draw_car_dummy(void) {
24 glUniform3f(loc_primitive_color, 0.498f, 1.000f, 0.831f); // color name: Aquamarine
25 draw_geom_obj(GEOM_OBJ_ID_CAR_BODY); // draw body
26 glLineWidth(2.0f);
27 draw_axes(); // draw MC axes of body
28 glLineWidth(1.0f);
29
30 ModelMatrix_CAR_DRIVER = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(-3.0f, 0.5f, 2.5f));
31 ModelMatrix_CAR_DRIVER = glm::rotate(ModelMatrix_CAR_DRIVER, TO_RADIAN*90.0f, glm::vec3(0.0f, 1.0f, 0.0f));
32 ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_DRIVER;
33 glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
34 glLineWidth(5.0f);
35 draw_axes(); // draw camera frame at driver seat
36 glLineWidth(1.0f);
37
38 ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(-3.0f, -3.5f, 4.5f));
39 ModelMatrix_CAR_WHEEL = glm::rotate(ModelMatrix_CAR_WHEEL, 1.0f, glm::vec3(0.0f, 0.0f, 1.0f));
40 ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
41 glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
42
43 ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(3.0f, -3.5f, 4.5f));
44 ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
45 glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
46 draw_wheel_L_and_nut(); // draw wheel 0
47
48 ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(-3.0f, -3.5f, -4.5f));
49 ModelMatrix_CAR_WHEEL = glm::scale(ModelMatrix_CAR_WHEEL, glm::vec3(1.0f, 1.0f, -1.0f));
50 ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
51 glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
52
53 ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(3.0f, -3.5f, -4.5f));
54 ModelMatrix_CAR_WHEEL = glm::scale(ModelMatrix_CAR_WHEEL, glm::vec3(1.0f, 1.0f, -1.0f));
55 ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
56 glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
57
58 draw_wheel_L_and_nut(); // draw wheel 2
59
60 void display(void) { // Display callback
61
62 gl::mat4 ModelMatrix_big_cow, ModelMatrix_small_box;
63 gl::mat4 ModelMatrix_bigr_g_box, ModelMatrix_small_box;
64
65 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
66
67 ModelViewProjectionMatrix = glm::scale(ViewProjectionMatrix, glm::vec3(5.0f, 5.0f, 5.0f));
68 glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
69 glLineWidth(2.0f);
70 draw_axes();
71 glLineWidth(1.0f);
72
73 ModelMatrix_CAR_BODY = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(20.0f, 4.89f, 0.0f));
74 ModelMatrix_CAR_BODY = glm::rotate(ModelMatrix_CAR_BODY, 90.0f*TO_RADIAN, glm::vec3(0.0f, 1.0f, 0.0f));
75 ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_BODY;
76 glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
77 draw_car_dummy();
78
79 glutSwapBuffers();
80 }
81
82 }
```

[CSE4170: 기초 컴퓨터 그래픽스]

## 중간고사

(담당교수: 임인성)

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

## 1. 다음 문제에 답하라.

- 2차원 공간에서 원점을 지나고 길이가 1인 벡터  $p$  방향을 향하는 직선을 고려하자. 이 때 한 점  $q$ 가 주어졌을 때, 이 점과 이 직선과의 최단 거리를 벡터의 내적 연산 및 벡터의 길이 연산자를 사용하여 표현하라(내적 연산자 기호는  $\circ$ , 그리고 벡터의 길이 연산자는  $\| \cdot \|$  사용).
- 3차원 공간에서 두 벡터  $(1, 2, 3)$ 과  $(4, 5, 6)$  간의 외적  $(1, 2, 3) \times (4, 5, 6)$ 를 구하라.
- RGB 모델로  $(0.3, 0.5, 1.0)$ 로 표현되는 색깔을 CMY 모델로 나타내면?
- 3차원 투영공간의 점  $(5.0, -5.0, -10.0, -2.0)$ 에 대응하는 아핀공간의 점의 좌표  $(x, y, z)$ 를 기술하라.
- 2차원 공간의 두 직선  $3x + 2y + 5 = 0$ 과  $6x + 4y + 5 = 0$ 의 교점에 대한 투영 공간에서의 동차 좌표를 상수만 사용하여 기술하라.
- 2차원 공간에서 원점을 둘레로 반시계 방향으로  $\theta$  각도만큼 회전시켜주는 3행 3열 아핀 변환 행렬  $R(\theta)$ 을 기술하라.
- 3차원 공간에서 원점을 지나고 벡터  $(0 \ 0 \ 1)^t$ 이 가리키는 방향에 의해 정의되는 직선 둘레로 반시계 방향으로  $\theta$  각도만큼 회전시켜주는 4행 4열 아핀 변환 행렬  $R(\theta, 0, 0, 1)$ 을 기술하라.
- 3차원 공간에서의 아핀 변환을 나타내는 4행 4열 행렬  $M$ 이 강체 변환이 되기 위한 조건을 이 행렬의 왼쪽 위 부분의 3행 3열 부행렬을  $M_{33}$ 의 성질을 통하여 정확히 기술하라.
- OpenGL 렌더링 시스템에서는 사진 촬영 시 피사체를 배치하는 과정을 어떤 변환을

통하여 모사하는가?

- OpenGL 렌더링 시스템에서 모델뷰 행렬 (ModelView Matrix)은 기본적으로 어떤 좌표계에서 어떤 좌표계로의 기하변환을 해주는가?

- OpenGL 렌더링 시스템에서 항상 3차원 공간의  $[-1, 1] \times [-1, 1] \times [-1, 1]$ 의 영역만 고려하는 좌표계의 이름은 무엇인가?

- 다음 행렬  $M_1$ 의 역행렬을 구하라.

$$M_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 다음 행렬  $M_2$ 의 역행렬을 구하라.

$$M_2 = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 & 1 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 만약 위의 행렬  $M_2$ 가 OpenGL 렌더링 시스템의 ‘카메라의 위치와 방향’을 설정해주는 뷰 행렬 (view matrix)라 하면, 이때 카메라는 정확히 ‘어느 지점’에서 ‘어느 방향’을 바라보고 있는 상태를 의미할까? (참고: 방향에 주의)

- 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  $M_2$ 의 역행렬  $M_2^{-1}$ 을 위의 기본 변환 행렬의 곱으로 표현하라.

- 투영 참조점이 투영 평면에서 유한 거리만큼 떨어진 투영 변환의 이름은 무엇인가?

- 아래의 변환 행렬  $M_3$ 을 통하여 점  $(x \ y \ z \ 1)^t$ 에 대하여 원근 투영을 수행할 때 진행되는

원근 나눗셈 (perspective division)에서 분모로 사용되는  $W$  좌표  $w_c$  값은 무엇인가?

$$M_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix}$$

- (r) 위 문제에서  $w_c$  값은 투영하려는 점의 어떤 기하적인 성질을 표현하는지 정확히 기술하라.

2. 3행 3열 행렬로 표현되는 2차원 이동 변환  $T(t_x, t_y)$ , 크기 변환  $S(s_x, s_y)$ , 그리고 회전 변환  $R(\theta)$ 를 고려하자. 그림 1의 위쪽의 직사각형 영역의 내용을 아래쪽의 직사각형 영역으로 매핑해주는 3행 3열의 기하 변환 행렬  $M$ 을 (i) 위의 기본 변환 행렬들의 곱으로 표현한 후, (ii) 최종 3행 3열 행렬의 내용을 기술하라.

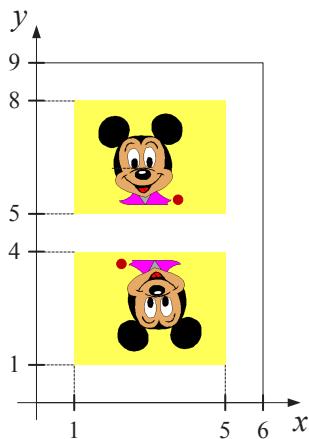


Figure 1: 2차원 윈도우 매핑

3. 다음 기하 변환에 대한 문제에 답하라.
- (a) (2번 문제의 2차원 기하 변환 행렬 기호를 고려할 때)  $R(\theta)S(1, -1) = S(1, -1)R(\alpha)$  식을 만족시켜주는  $\alpha$  값은 무엇인가?
  - (b)  $T(t_x, t_y)S(1, -1) = S(1, -1)T(\beta, \gamma)$  식을 만족시켜주는  $\beta$ 와  $\gamma$  값은 무엇인가?
  - (c)  $R(\theta)T(t_x, t_y) = T(\delta, \epsilon)R(\theta)$  식을 만족시켜주는  $\delta$ 와  $\epsilon$  값은 무엇인가?

4. 그림 2를 보면서 답하라.

- (a) 이 그림에서와 같이 2차원 공간의 점  $p$ 를 직선  $L$ 에 대하여 반사시켜  $p'$ 으로 변환시켜주는 2차원 기하 변환을 다음 OpenGL 함수들을 적절히 사용하여 C언어 문법에 맞게 정확히 기술하라. 함수 호출 순서는

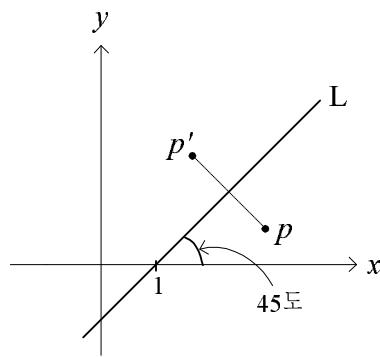


Figure 2: 직선에 대한 반사

OpenGL 프로그래밍 관례에 따르고, 이동 변환이 필요하다면  $z$  값은 0으로, 크기 변환 이 필요하다면  $z$  값은 1로 설정할 것.

- void glTranslatef(GLfloat x, GLfloat y, GLfloat z);
- void glScalef(GLfloat x, GLfloat y, GLfloat z);
- void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);

- (b) 위의 코드는 아래와 같이 세 문장으로 표현이 가능하다. 각 기하 변환 함수에 적절한 인자를 기술하라. (힌트: 3번 문제의 관계식을 활용할 것)

```
glRotatef(, , ,);
glScalef(, ,);
glTranslatef(, ,);
```

5. 다음과 같은 아핀 변환 행렬  $M$ 을 고려하자.

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

- (a)  $M$ 의 위-왼쪽 부분의 3행 3열 부행렬을  $M_{33}$  이라 하자. 지금 이 아핀 변환  $M$ 을 통하여 법선 벡터  $n = (n_x \ n_y \ n_z)^t$ 를 변환한 법선 벡터를  $n' = (n'_x \ n'_y \ n'_z)^t$ 라 할 때,  $n'$ 의 값을  $n$ 과  $M_{33}$  등을 적절히 사용하여 표현하라 (이 법선 벡터들은 모두 3차원 공간에서의 열벡터임).
  - (b) 만약  $n = (\frac{1}{\sqrt{3}} \ \frac{1}{\sqrt{3}} \ \frac{1}{\sqrt{3}})^t$ 이라면 위의 아핀 변환을 수행한 후의  $n'$ 은 무엇일 지 그 벡터를 정확히 기술하라.
6. 다음은 프레임간의 변환에 관한 문제이다.

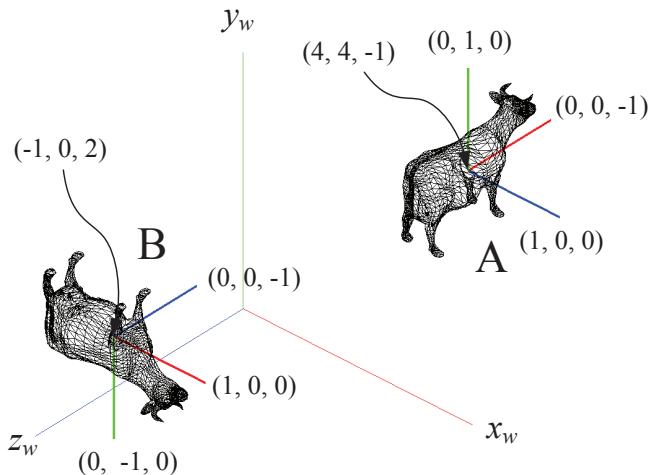


Figure 3: 프레임간의 변환

- (a) 그림 3의 A 소는 점  $(4, 4, -1)$ 을 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  $x_w$ 축과  $y_w$  방향과 일치하는 방식으로 세상 좌표계와 일치시켜주려 한다. 이때 필요한 4행 4열 행렬  $M_a$ 의 내용을 기술하라.
- (b) 한편 B 소는 점  $(-1, 0, 2)$ 을 중심으로 하는 프레임을 기준으로 세상 좌표계에 존재하고 있는데, A 소의 각 꼭지점을 B 소의 대응되는 점으로 매핑해주는 4행 4열 행렬  $M_b$ 를 네 개의 기본 변환 행렬의 곱으로 표현하라(반드시 네 개의 4행 4열 행렬의 내용을 곱해지는 순서대로 기술할 것).
7. 그림 4에서와 같이 COP(Center of Projection) 가  $(1, 0, 1)$ 이고 PP(Projection Plane)가  $z = -9$ 인 상황에서, 주어진 점  $p = (x \ y \ z \ 1)^t$ 을  $p' = (x' \ y' \ z' \ 1)^t$ 로 변환해주는 4행 4열의 원근 투영 변환 행렬  $M$ 을 구하라(반드시 유도과정이 있어야 함).
8. 다음은 간단한 모델링 변환에 관한 문제이다.

- (a) 아래의 코드는 그림 5(a)에 도시한 그림을 그려주는 OpenGL 프로그램의 일부이다.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(10.0, 10.0, 10.0,
 0.0, 0.0, -1.0, 0.0, 1.0, 0.0);
```

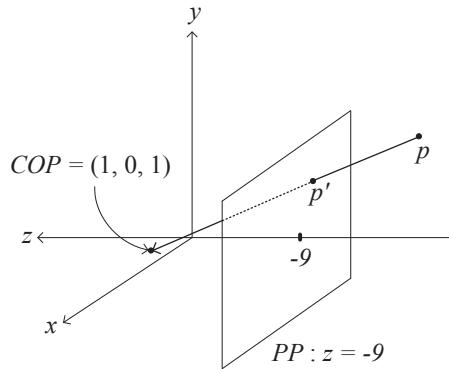
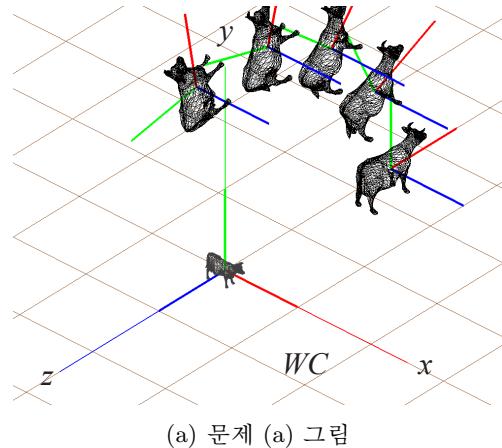
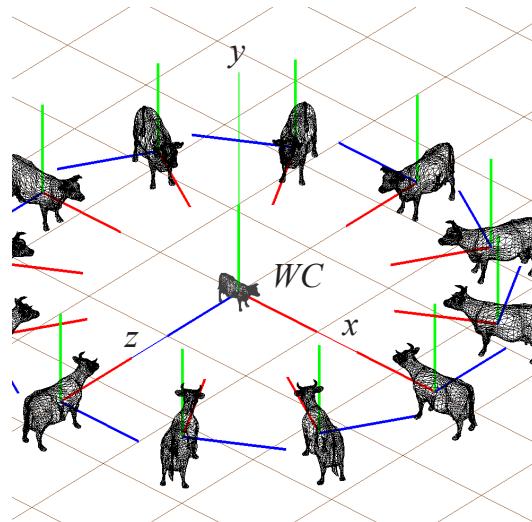


Figure 4: 원근 투영 변환 문제



(a) 문제 (a) 그림



(b) 문제 (b) 그림

Figure 5: 간단한 모델링 변환

```
draw_floor();
draw_axis(); // Draw the WC axes
draw_cow(0.25, 0.25, 0.25); // Line
(a)

for (i = 0; i < 5; i++) {
```

```

glPushMatrix();
glRotatef(, , ,);
glTranslatef(, ,);
glRotatef(, , ,);
// Draw the MC axes
draw_axis(); Line (c)
glScalef(2.0, 2.0, 2.0);
draw_cow(0.0, 0.0, 0.0); // Line
(b)
glPopMatrix();
}

```

여기서 Line (a)의 함수는 원점에 배치되어 있는 회색의 소를 그려주고 있으며, Line (b)의 함수는 소의 모델링 좌표계의 원점이 세상 좌표계의 원점을 둘레로 반경이 5인 원을 따라  $y - z$  평면상에서 25도씩 회전하고 있는 검정색 소를 그려주는 역할을 하고 있는데(이 소는 원점 주변의 소보다 2배가 크고 가장 아래쪽 소가 가장 먼저 그려짐), 이 코드에는 회전에 필요한 OpenGL API 함수들에 대한 인자가 결여되어 있다. 위 프로그램이 제대로 작동하기 위한 인자를 C/C++ 언어 문법에 맞게 정확히 기술하라.

- (b) (위의 프로그램을 제대로 완성하였다는 가정하에) 만약 Line (c)의 draw\_axis(); 문장과 그 다음의 glScalef(2.0, 2.0, 2.0); 문장의 순서를 바꾸면 어떤 변화가 있는지 정확한 수치를 사용하여 기술하라.
- (c) 아래 코드는 (a)번 문제의 프로그램에서 for 루프 부분을 대치한 것이다.

```

for (i = 0; i < 12; i++) {
 glPushMatrix();
 draw_axis();
 glScalef(2.0, 2.0, 2.0);
 draw_cow(0.0, 0.0, 0.0);
 glPopMatrix();
}

```

그림 5(b)에서는 두 배 커진 소의 모델링 좌표계의 원점이 세상 좌표계의 원점 둘레로 반경이 5인 원을 따라  $z - x$  평면상에서 30도씩 자연스럽게 회전하고 있는 모습을 도시하고 있다. 이와 같은 결과를 얻기 위하여 필요한 문장(들)을 OpenGL 및 C/C++ 언어의 문법에 맞게 기술하라. 이때 “어느 문장 바로 뒤에 어떤 문장 삽입”과 같이 정확히 기술할 것.

9. 다음은 아래의 gluLookAt(\*) 함수를 통한 뷰잉 변환에 관한 문제이다. 그림 6의 뷰잉 변환 계산 과정을 보면서 답하라.

```

gluLookAt(ex, ey, ez, cx, cy, cz, ux,
uy, uz);

```

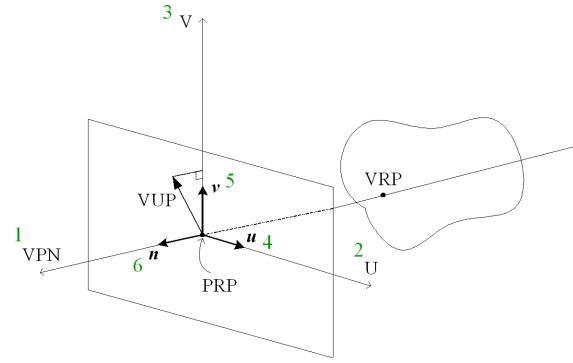


Figure 6: gluLookAt(\*) 함수를 통한 카메라의 설정

- (a) 이 그림에서 두 점 PRP와 VRP로 정의되는 3차원 공간에서의 열벡터  $\text{VPN} \equiv \text{PRP} - \text{VRP}$ 를 gluLookAt() 함수의 인자들을 사용하여 기술하라.
- (b) 한편 VUP은 위 함수의 마지막 세 인자  $ux, uy, uz$ 로 정의되는 열벡터이다. 이때 카메라의 오른쪽에 해당하는 방향 벡터를 VRP, PRP, 그리고 VUP 등 이 세 벡터에 대한 적절한 벡터 연산을 통하여 기술하라(벡터의 길이를 1로 만들 필요가 없으며, 그 벡터 연산의 이름을 기호와 함께 명확히 밝힐 것).
- (c) 위 문제에 이어 카메라의 위쪽에 해당하는 방향 벡터를 VRP, PRP, 그리고 VUP 등 이 세 벡터에 대한 적절한 벡터 연산을 통하여 기술하라(벡터의 길이를 1로 만들 필요가 없으며, 그 벡터 연산의 이름을 기호와 함께 명확히 밝힐 것).

10. 다음은 뷰잉 변환에 관한 문제이다.

- (a) 그림 7에는 카메라의 위치와 방향을 설정 해주는 카메라 프레임이 도시되어 있다. 이에 해당하는 뷰잉변환을 아래에서와 같이 OpenGL의 gluLookAt() 함수를 사용하여 구현하려 할 때, 필요한 인자 9

개를 순서대로 기술하라.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(*,*,*,*,*,*,*,*,*);
```

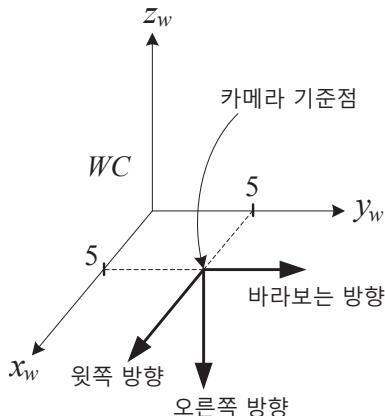


Figure 7: 카메라의 배치

(b) <문제 삭제>

(c) 이 문제의 뷰잉 변환을 다음과 같이 구현하려 한다.

```
GLfloat m[16];
:
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMultMatrixf(m);
glTranslatef((a), (b), (c));
```

이때 (a), (b), (c)의 값과 배열  $m$ 에 저장되어야 할 16개 원소 값을 순서대로 기술하라. (주의: OpenGL에서의 행렬 저장 순서를 상기할 것).

11. 다음은 직교 투영 변환과 관련한 문제이다. 그림 8에서와 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 기본 아핀 변환의 합성을 통하여 표현하라. (주의: 변환 후  $z$ 축의 방향이 반대가 되도록 네 모서리를 마취주어야 하며, 최종 행렬 내용은 기술할 필요 없음)
12. 시험지 뒤에 첨부한 프로그램은 적절한 모델링 변환을 통하여 자동차를 세상 좌표계로 그려주는 OpenGL 프로그램이다. 이 프로그램과 그림 9를 보면서 답하라.

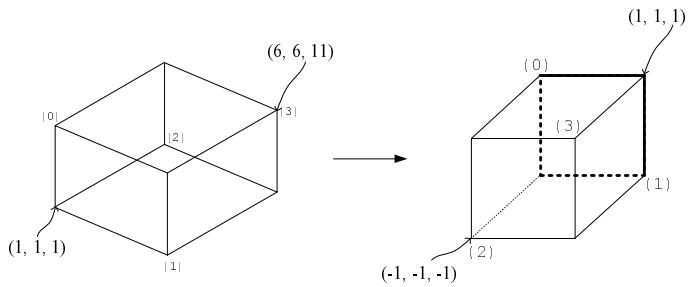


Figure 8: 직교 투영 변환 문제

- (a) 프로그램 문맥상 12번 문장에 들어갈 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라.
- (b) 프로그램 문맥상 25번 문장에 들어갈 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라.
- (c) 세상 좌표계를 기준으로 할 때 카메라가 세상을 바라보는 방향 벡터를 기술하라.
- (d) 눈 좌표계를 기준으로 할 때 카메라가 세상을 바라보는 방향 벡터를 기술하라.
- (e) 85번의 문장이 수행되기 직전의 모델뷰 행렬 스택의 내용을 1.(o)번의 행렬 기호를 사용하여 정확히 기술하라(필요하다면 이 프로그램에서 사용되고 있는 변수들도 사용할 것).
- (f) 35번 문장의 `draw_wheel_and_nut(angle)` 함수에서 이상한 부분을 지적하고 수정하라. 이유를 설명할 것.
- (g) 49번 문장의 `wheel_rot_angle_in_y(void)` 함수에서 배열  $path[i]$ 는 자동차가 움직이는 경로에 대한 정보를 저장하고 있다. 문맥상 이 함수는 자동차 바퀴의 어떤 움직임을 표현하기 위한 것인지 정확히 기술하라.
- (h) 문맥상 65번 문장의 `wheel_rot_angle_in_z(void)` 함수에서 전역 변수  $dist$ 는 매 프레임 마다 어떤 정보를 저장하고 있을지 기술하라.
- (i) 이 경우 이 함수는 자동차 바퀴의 어떤 움직임을 표현하기 위한 것인가?
- (j) (이 프로그램에 문제가 있건 없건 현재 상태에서) 그림 9에서의 행렬  $M_n^4$ 에 해당하는 행렬을 1.(o)번의 행렬 기호를 사용하여 정확히 기술하라(상수 값을 알고 있는 경우 반드시 그 상수 값을 사용할 것).

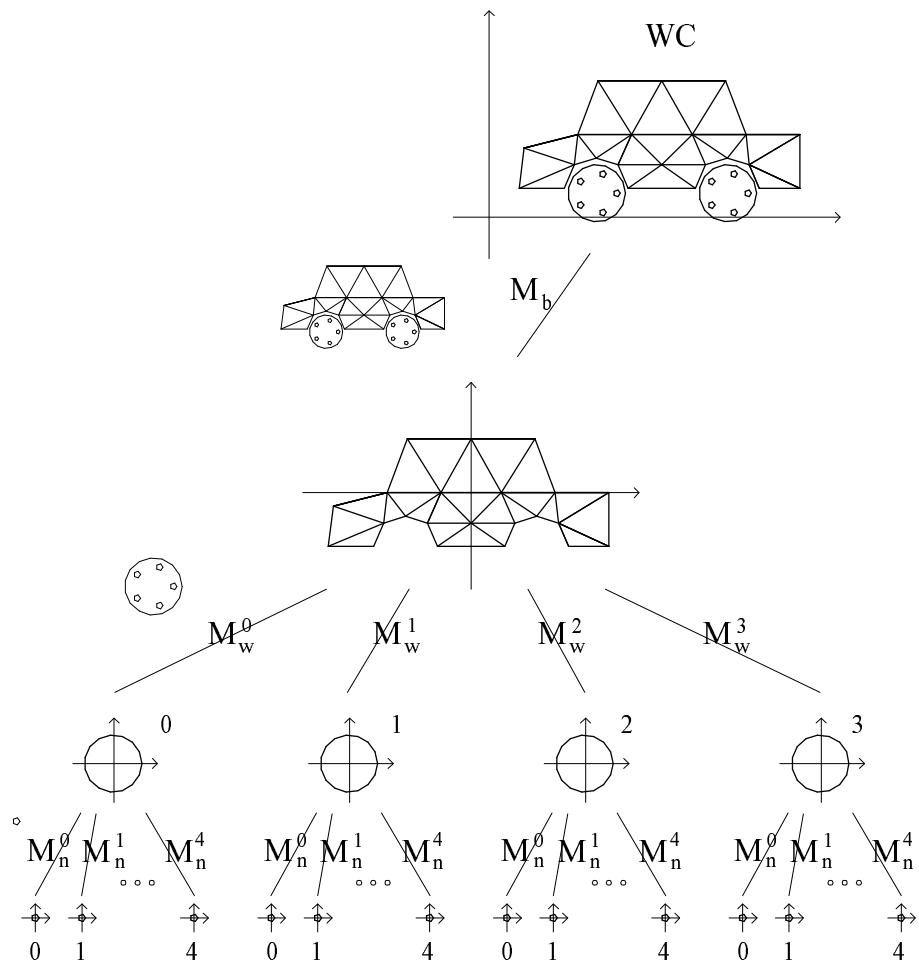


Figure 9: 자동차의 계층적 표현

```

1 #define T0_DEG 57.29579
2 void normalize_vec3(float *v) {
3 int i;
4 float tmp;
5
6 tmp = 1.0/sqrt(v[0]*v[0] + v[1]*v[1] + v[2]*v[2]);
7 // do somethig here!
8 for (i = 0; i < 3; i++) v[i] *= tmp;
9 }
10
11 float dot_prod_vec3(float *u, float *v) {
12 return(????);
13 }
14
15 float compute_length_mul_two_vec3(float *u, float *v) {
16 float tmp;
17
18 tmp = (u[0]*u[0]+u[1]*u[1]+u[2]*u[2])*(v[0]*v[0]+v[1]*v[1]+v[2]*v[2]);
19 return(sqrt(tmp));
20 }
21
22 float angle_between_two_vec3(float *u, float *v) {
23 float tmp;
24
25 tmp = ??? /compute_length_mul_two_vec3(u, v);
26 returnacos(tmp));
27 }
28
29 void cross_prod_vec3(float *u, float *v, float *n) {
30 n[0] = u[1]*v[2] - u[2]*v[1];
31 n[1] = u[2]*v[0] - u[0]*v[2];
32 n[2] = u[0]*v[1] - u[1]*v[0];
33 }
34
35 void draw_wheel_and_nut(float angle) {
36 int i;
37
38 draw_wheel(angle); // draw wheel object
39 for (i = 0; i < 5; i++) {
40 // nut i
41 glPushMatrix();
42 glTranslatef(1.2, 0.0, 1.0);
43 glRotatef(72.0*i, 0.0, 0.0, 1.0);
44 draw_nut(); // draw nut object
45 glPopMatrix();
46 }
47 }
48
49 float wheel_rot_angle_in_y(void) {
50 int i;
51 float angle, before[3], after[3], result[3];
52
53 angle = 0.0;
54 if ((cur_i < 50) || (cur_i >= npath - 50)) return(angle);
55 for (i = 0; i < 3; i++) before[i] = path[cur_i][i] - path[cur_i-50][i];
56 for (i = 0; i < 3; i++) after[i] = path[cur_i+50][i] - path[cur_i][i];
57
58 angle = 1.2*T0_DEG*angle_between_two_vec3(before, after);
59 cross_prod_vec3(before, after, result);
60
61 if (result[1] < 0) angle *= -1.0;
62 return(angle);
63 }
64
65 float wheel_rot_angle_in_z(void) {
66 static int angle = 0.0;
67
68 angle += 0.8*180.0*dist/(3.141592*rad);
69 if (angle >= 360.0) angle -= 360.0;
70
71 return(angle);

```

```

72 }
73
74 void draw_car_correct(void) {
75 float angle_y = 0.0, angle_z = 0.0;
76
77 angle_y = wheel_rot_angle_in_y();
78 angle_z = wheel_rot_angle_in_z();
79
80 draw_body(); // draw body object
81 glPushMatrix();
82 glTranslatef(-3.9, -3.5, 4.5);
83 glRotatef(angle_y, 0.0, 1.0, 0.0);
84 glRotatef(angle_z, 0.0, 0.0, 1.0);
85 draw_wheel_and_nut(angle_y); // wheel 0
86 glPopMatrix();
87
88 glPushMatrix();
89 glTranslatef(3.9, -3.5, 4.5);
90 glRotatef(angle_z, 0.0, 0.0, 1.0);
91 draw_wheel_and_nut(0.0); // wheel 1
92 glPopMatrix();
93
94 glPushMatrix();
95 glTranslatef(-3.9, -3.5, -4.5);
96 glRotatef(angle_y, 0.0, 1.0, 0.0);
97 glRotatef(angle_z, 0.0, 0.0, 1.0);
98 glScalef(1.0, 1.0, -1.0);
99 draw_wheel_and_nut(angle_y); // wheel 2
100 glPopMatrix();
101
102 glPushMatrix();
103 glTranslatef(3.9, -3.5, -4.5);
104 glRotatef(angle_z, 0.0, 0.0, 1.0);
105 glScalef(1.0, 1.0, -1.0);
106 draw_wheel_and_nut(0.0); // wheel 3
107 glPopMatrix();
108 }
109
110 void draw_world (void) {
111 GLfloat m[16], minv[16];
112
113 set_up_rot_mat(m, minv, cur_i);
114
115 glMatrixMode(GL_MODELVIEW);
116 glLoadIdentity();
117 glRotatef(90.0, 0.0, 1.0, 0.0);
118 glTranslatef(0.0, 0.0, -20.0);
119
120 draw_body();
121 draw_fences();
122 draw_ground();
123 draw_axes();
124 draw_path();
125
126 glPushMatrix();
127 glTranslatef(10.0, 5.0, 20.0);
128 glRotatef(25.0, 0.0, 1.0, 0.0);
129 draw_car_correct();
130 glPopMatrix();
131 }
```

## [CSE4170: 기초 컴퓨터 그래픽스]

### 중간고사

(담당교수: 임인성)

- 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.

- 3차원 공간의 두 벡터  $\mathbf{p} = (p_x \ p_y \ p_z)^t$  와  $\mathbf{q} = (q_x \ q_y \ q_z)^t$ 에 대하여 외적 (cross product) 연산을 통해 구한 벡터  $\mathbf{r} = \mathbf{p} \times \mathbf{q} = (r_x \ r_y \ r_z)^t$ 의  $x$  좌표  $r_x$ 의 값을 기술하라.
- 3차원 투영공간의 점  $(5.0, -5.0, -10.0, -2.0)$ 에 대응하는 아핀공간의 점의 좌표  $(x, y, z)$ 를 기술하라.
- RGB 모델로  $(0.3, 0.5, 1.0)$ 로 표현되는 색깔을 CMY 모델로 나타내면?
- 2차원 공간에서의 회전 변환에 해당하는 3행 3열 아핀 변환 행렬  $R(\theta)$ 을 기술하라.
- 주어진 4행 4열 행렬이 3차원 공간에서의 아핀 변환에 해당하는지를 알 수 있는 방법은 무엇인가?
- 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. 다음 행렬  $M$ 의 역행렬  $M^{-1}$ 을 위의 기본 변환 행렬의 곱으로 표현하라 (반드시 위의 행렬 기호를 사용할 것).

$$M = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & 5 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- OpenGL 렌더링 파이프라인에서 ‘카메라의 위치와 방향을 설정’해주는 과정에 해당하는 기하 변환의 이름은 무엇인가?
- OpenGL fixed-function 파이프라인에서 `glVertex3f()` 함수를 사용하여 설정한 꼭지점에 곱해지는 첫 번째 행렬 스택의 탑에

있는 변환 행렬은 일반적으로 어떤 좌표계에서 어떤 좌표계로의 변환을 위하여 사용하는가?

- 투영 참조점이 무한대점 (point at infinity)에 위치한 투영 변환의 이름은 무엇인가?
- 다음과 같은 `gluPerspective(fovy, asp, n, f)` 함수에 대한 변환 행렬  $M_{pers}$ 를 통하여 원근 투영을 수행할 때 진행되는 원근 나눗셈에서 분모로 사용되는  $W$  좌표  $w_c$  값은 무엇인가?

$$\begin{bmatrix} \frac{\cot(\frac{fovy}{2})}{asp} & 0 & 0 & 0 \\ 0 & \cot(\frac{fovy}{2}) & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2nf}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

- 위 문제에서  $w_c$  값의 직관적인 의미를 설명하라.

- 3행 3열 행렬로 표현되는 2차원 이동 변환  $T(t_x, t_y)$ , 크기 변환  $S(s_x, s_y)$ , 그리고 회전 변환  $R(\theta)$ 를 고려하자. 그림 1(a)의 직사각형 영역의 내용을 (b)의 직사각형 영역으로 매핑해주는 3행 3열의 기하 변환 행렬  $M$ 을 (i) 위의 기본 변환 행렬들의 곱으로 표현한 후, (ii) 최종 3행 3열 행렬의 내용을 구하라.

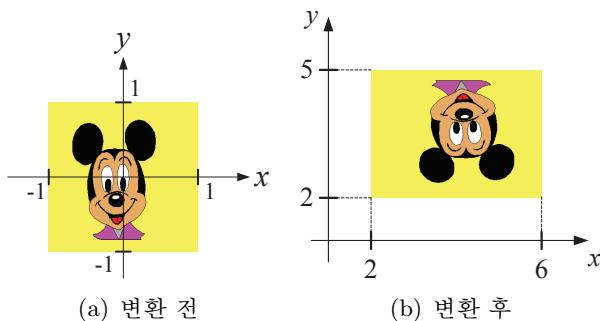


Figure 1: 2차원 윈도우 매핑

3. 아래의 코드는 그림 2에 도시한 원점 주변의 0번 비행기와 4사분면의 1번 비행기를 그려주는 OpenGL 코드이다.

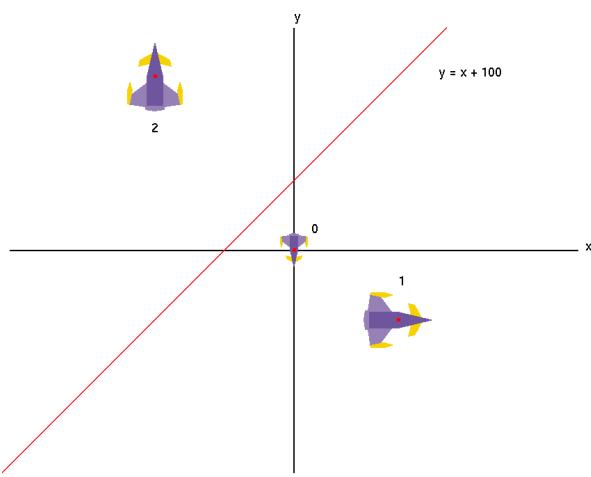


Figure 2: 2차원 기하 변환

```
draw_airplane(); // airplane 0
```

```
glPushMatrix();
glTranslatef(150.0, -100.0, 0.0);
glRotatef(90.0, 0.0, 0.0, 1.0);
glScalef(2.0, 2.0, 1.0);
draw_airplane(); // airplane 1
glPopMatrix();
```

- (a) 이제 이 그림에서처럼 1번 비행기를 직선  $y = x + 100$ 에 대하여 반사시켜 2사분면의 2번 비행기와 같이 그려주려 한다. 아래의 코드는 이를 위한 코드인데, (A)와 (B)에 들어갈 내용을 각각 한 번의 OpenGL API 함수 호출을 통하여 C/C++ 언어 문법에 맞게 기술하라.

```
glPushMatrix();
glTranslatef(-100.0, 0.0, 0.0);
(A)
glScalef(-1.0, 1.0, 1.0);
(B)
glTranslatef(250.0, -100.0, 0.0);
glRotatef(90.0, 0.0, 0.0, 1.0);
glScalef(2.0, 2.0, 1.0);
draw_airplane(); // airplane 2
glPopMatrix();
```

- (b) 3행 3열 행렬로 표현되는 2차원 이동 변환  $T(t_x, t_y)$ , 크기 변환  $S(s_x, s_y)$ , 그리고 회

전 변환  $R(\theta)$ 를 고려하자.  $R(\theta)S(-1, 1) = S(-1, 1)R(\alpha)$  식을 만족시켜주는  $\alpha$  값은 무엇인가?

- (c)  $S(-1, 1)T(t_x, t_y) = T(\beta, \gamma)S(-1, 1)$  식을 만족시켜주는  $\beta$ 와  $\gamma$  값은 무엇인가?
- (d)  $R(\theta)T(t_x, t_y) = T(\delta, \epsilon)R(\theta)$  식을 만족시켜주는  $\delta$ 와  $\epsilon$  값은 무엇인가?
- (e) 이제 2번 비행기를 그리는 작업은 아래와 같은 방식으로도 구현할 수 있다. 이때 (C)에 들어갈 내용을 한 번의 OpenGL API 함수 호출을 통하여 C/C++ 언어 문법에 맞게 기술하라.

```
glPushMatrix();
glTranslatef(-100.0, 0.0, 0.0);
glRotatef(-90.0, 0.0, 0.0, 1.0);
(C)
glTranslatef(250.0, -100.0, 0.0);
glRotatef(90.0, 0.0, 0.0, 1.0);
glScalef(2.0, 2.0, 1.0);
draw_airplane(); // airplane 2
glPopMatrix();
```

- (f) 마찬가지로 아래와 같은 방식으로 2번 비행기를 그리려 한다. 이때 공란으로 되어있는 각 OpenGL API 함수의 인자 다섯 개를 순서대로 기술하라. (힌트: 1번 문제의 답을 잘 이용할 것)

```
glPushMatrix();
glTranslatef(___, ___, 0.0);
glRotatef(___, 0.0, 0.0, 1.0);
glScalef(___, ___, 1.0);
draw_airplane(); // airplane 2
glPopMatrix();
```

4. 다음은 프레임간의 변환에 관한 문제이다.

- (a) 그림 3의 A 소는 점  $(4, 4, -1)$ 을 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  $x_w$ 축과  $y_w$  방향과 일치하는 방식으로 세상 좌표계와 일치시켜주려 한다. 이때 필요한 4행 4열 행렬  $M_1$ 의 내용을 기술하라.
- (b) 한편 B 소는 점  $(-1, 0, 2)$ 을 중심으로 하는 프레임을 기준으로 세상 좌표계에 존재하고 있는데, A 소의 각 꼭지점들을 B 소의

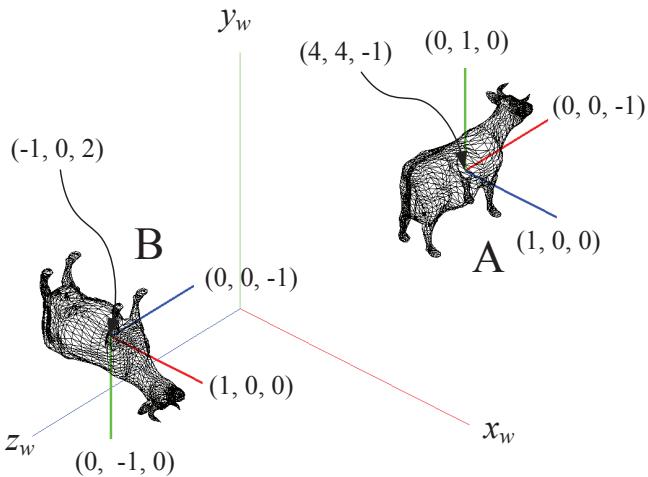


Figure 3: 프레임간의 변환

대응되는 점으로 매핑해주는 4행 4열 행렬  $M_2$ 를 네 개의 기본 변환 행렬의 곱으로 표현하라(반드시 네 개의 4행 4열 행렬의 내용을 곱해지는 순서대로 기술할 것).

5. 다음은 원근 투영 변환에 관한 문제이다.

- (a) 그림 4(a)에서와 같이 COP(Center of Projecton)가  $(1, 0, 0)$ 이고 PP(Projection Plane)이  $x = 9$ 인 상황에서, 주어진 점  $p = (x \ y \ z \ 1)^t$ 을  $p' = (x' \ y' \ z' \ 1)^t$ 로 변환해주는 4행 4열의 원근 투영 변환 행렬  $M_1$ 을 구하라(반드시 유도과정이 있어야 함).
- (b) 그림 4(b)는 임의의 점  $p$ 를 이 점과 원점을 지나는 직선과 세 점  $(1, 0, 0)$ ,  $(0, 1, 0)$ , 그리고  $(0, 0, 1)$ 을 지나는 평면과의 교점  $p'$ 로 투영해주는 모습을 도시하고 있다. 이때 이에 대한 원근 투영 변환 행렬  $M_2$ 를 구하라(반드시 유도과정이 있어야 함).

6. 다음은 간단한 모델링 변환에 관한 문제이다.

- (a) 아래의 코드는 그림 5(a)에 도시한 그림을 그려주는 OpenGL 프로그램의 일부이다.

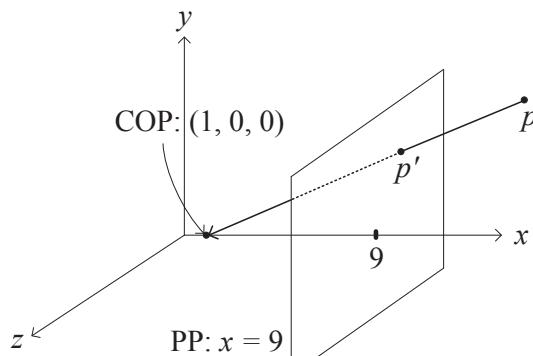
```

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(20.0, 15.0, 20.0,
 0.0, 0.0, 0.0, 1.0, 0.0);

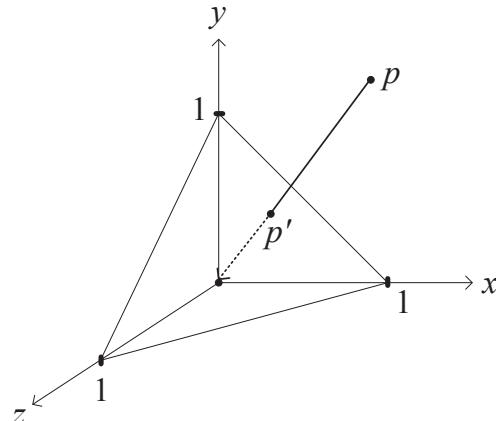
draw_axis(); draw_floor();
draw_cow(); // Line (a)

for (angle = 0.0; angle < 360.0;
 angle += 45.0) {

```



(a) 문제 (a) 그림



(b) 문제 (b) 그림

Figure 4: 원근 투영 변환

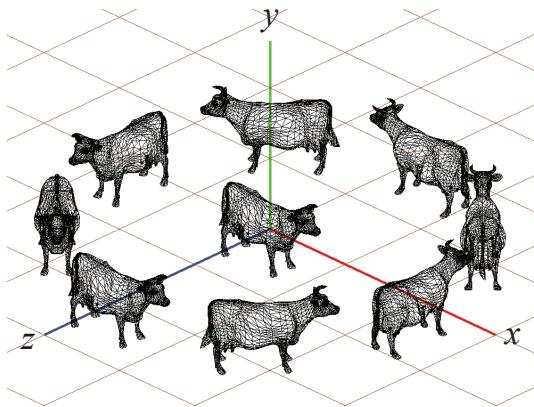
```

glPushMatrix(); // Line (c)
glTranslatef(4.0, 0.0, 0.0);
draw_cow(); // Line (b)
glPopMatrix();
}

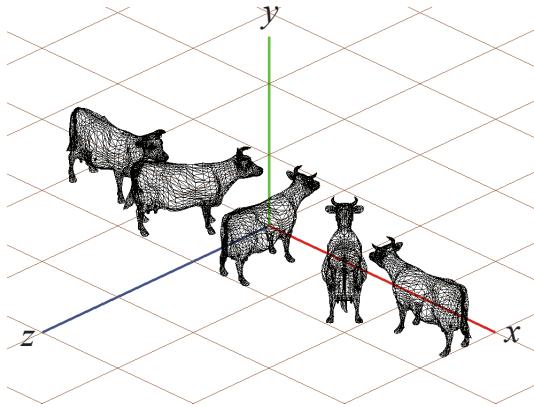
```

여기서 Line (a)의 함수는 원점에 배치되어 있는 소를 그려주고, Line (b)의 함수는 원점을 둘레로 반경 4인 원을 따라 회전하고 있는 소를 그려주는 역할을 하고 있는데, 이 코드에는 회전에 필요한 OpenGL API 함수 호출 문장이 결여되어 있다. 위 프로그램이 제대로 작동하기 위하여 어느 지점에 어떤 문장이 필요한지, 그 내용을 OpenGL 및 C/C++ 언어 문법에 맞게 정확히 기술하라. 답은 “어떤 문장 직전(또는 직후)에 다음 OpenGL 문장이 필요함”과 같이 기술하되, 두 개 이상의 OpenGL 문장이 필요할 수 있음.

- (b) 다음 프로그램에서 그림 5(b)와 같은 결과를 얻기 위하여 필요한 회전에 필요한 문장을 위 문제에서와 동일한 방식으로 답하라.



(a) 문제 (a) 그림



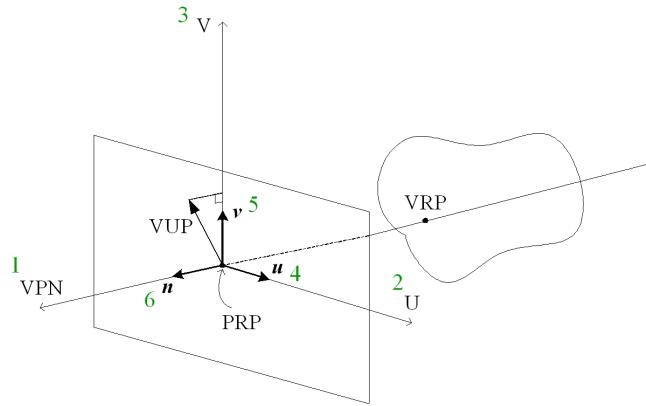
(b) 문제 (b) 그림

Figure 5: 간단한 모델링 변환

```
for (int i = 0; i < 5; i++) {
 glPushMatrix(); // Line (d)
 glTranslatef(2.0*i-4.0, 0.0, 0.0);
 draw_cow(); // Line (e)
 glPopMatrix();
}
```

7. 다음은 `gluLookAt(*)` 함수를 통한 뷰잉 변환에 관한 문제이다. 그림 6의 뷰잉 변환 계산 과정과 그림 7의 함수 구현 코드를 보면서 답하라.

- (a) 이 코드에서 이 그림의 VRP와 가장 밀접한 관련이 있는 변수(들)을 정확히 기술하라.
- (b) 이 코드에서 이 그림의  $v$ 와 가장 밀접한 관련이 있는 변수(들)을 정확히 기술하라.
- (c) 문맥 상 이 코드의 (A), (B), 그리고 (C) 부분에 들어갈 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라.
- (d) 이 코드에는 분명히 잘못된 부분이 있다. 그 부분을 명시한 후 올바르게 바로 잡아라.

Figure 6: `gluLookAt(*)` 함수를 통한 뷰잉 변환

8. 다음은 뷰잉 변환에 관한 문제이다.

- (a) 한 점  $e$ 와 서로 수직이고 길이가 1인 세 개의 벡터  $u$ (카메라 기준 오른쪽 방향),  $v$ (카메라 기준 위쪽 방향),  $n$ (카메라에서 세상을 바라보는 방향의 정반대 방향)으로 정의되는 카메라 프레임을 생각하자. 지금 점  $e$ 가 원점에, 그리고  $u$ ,  $v$ ,  $n$  벡터가 각각 세상 좌표계의  $x_w$ ,  $y_w$ ,  $z_w$  축 방향을 향하도록 카메라 프레임이 초기화되어 있다. 이 상태서 이 카메라 프레임을  $y_w$  축 둘레로 90도만큼 회전시킨 후,  $x_w$  축 방향으로 -10만큼 이동을 시켰다고 하자. 이때의 뷰잉 변환을 아래의 코드처럼 구현하려 할때, (A) 부분에 들어갈 내용을 C/C++ 언어 문법에 맞게 OpenGL API 함수 호출을 통하여 구현하라.

```
GLfloat matrix[16];
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
(A)
glGetFloatv(GL_MODELVIEW_MATRIX,
 matrix);
```

- (b) 여기서 `glGetFloatv(*)`; 문장은 이 문장 수행 당시의 현재 행렬 스택의 탑의 내용을 배열 `matrix[16]`으로 추출해주는 역할을 한다. 이때 이 배열에 저장되는 16개의 원소 값을 순서대로 기술하라(OpenGL 시스템에서 2차원 배열이 저장되는 순서를 상기할 것).

9. 지금 다음처럼 카메라 변수 `cam`을 정의한 후,

```
typedef struct _cam {
 float pos[3], uaxis[3], vaxis[3],
```

```

void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx,
 GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz) {
 GLdouble m[16]; GLdouble x[3], y[3], z[3]; GLdouble mag;

 z[0] = eyex - centerx; z[1] = eyey - centery; z[2] = eyez - centerz;
 mag = sqrt(z[0]*z[0] + z[1]*z[1] + z[2]*z[2]);
 if (mag) { z[0] /= mag; z[1] /= mag; z[2] /= mag; }
 y[0] = upx; y[1] = upy; y[2] = upz;
 x[0] = y[1]*z[2] - y[2]*z[1]; x[1] = -y[0]*z[2] + y[2]*z[0]; x[2] = y[0]*z[1] - y[1]*z[0];
 y[0] = (A); y[1] = (B); y[2] = (C);
 mag = sqrt(x[0]*x[0] + x[1]*x[1] + x[2]*x[2]);
 if (mag) { x[0] /= mag; x[1] /= mag; x[2] /= mag; }
 mag = sqrt(y[0]*y[0] + y[1]*y[1] + y[2]*y[2]);
 if (mag) { y[0] /= mag; y[1] /= mag; y[2] /= mag; }
#define M(row,col) m[col*4+row]
 M(0,0) = x[0]; M(0,1) = x[1]; M(0,2) = x[2]; M(0,3) = 0.0;
 M(1,0) = y[0]; M(1,1) = y[1]; M(1,2) = y[2]; M(1,3) = 0.0;
 M(2,0) = z[0]; M(2,1) = z[1]; M(2,2) = z[2]; M(2,3) = 0.0;
 M(3,0) = 0.0; M(3,1) = 0.0; M(3,2) = 0.0; M(3,3) = 1.0;
#undef M
 glMultMatrixd(m);
 glTranslated(eyex, eyey, eyez);
}

```

Figure 7: gluLookAt(\*) 함수의 구현

```

 naxis[3];
 GLfloat mat[16];
 GLdouble fovy, aspect, near_c,
 far_c;
} Cam;
Cam cam;

```

디스플레이 컬백 함수에서 다음과 같이 물체를 그리려 한다(여기서 sfactor는 float 타입의 전역 변수임).

```

void display (void) {
 glClear(GL_COLOR_BUFFER_BIT);
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
 // Line (a)
 glMultMatrixf(cam.mat);
 glTranslatef(-cam.pos[0],
 -cam.pos[1], -cam.pos[2]);
 // Line (b)
 glPushMatrix();
 glScalef(sfactor, sfactor,
 sfactor);
 // Line (c)
 draw_teapot();
}

```

```

 glPopMatrix();
 glFlush();
}

```

- (a) 보편적인 관점에서, Line (a)와 Line (b) 지점에서 glVertex\*(\*) 함수로 꼭지점의 좌표를 기술하면 이 꼭지점은 각각 눈 좌표계, 모델링 좌표계, 그리고 세상 좌표계 중 어느 좌표계에서의 의미를 가질까?
- (b) 다음과 같은 함수를 사용하여 세상 좌표계의 좌표축을 그려주려 한다.

```

void draw_axes(void) {
 glLineWidth(2.0);
 glBegin(GL_LINES);
 glColor3f(1.0, 0.0, 0.0); // x축
 glVertex3f(0.0, 0.0, 0.0);
 glVertex3f(150.0, 0.0, 0.0);
 :
 glEnd();
 glLineWidth(1.0);
}

```

이 함수를 Line (a), Line (b), 그리고 Line (c) 중 어느 시점에서 호출을 해야

할까?

- (c) 그림 8은 이 프로그램의 초기 렌더링 결과를 보여주고 있다. (여기서 세상 좌표계의  $x$ 축과  $y$ 축이 각각 오른쪽과 위쪽 방향으로 그려져 있고,  $z$ 축은 화면 앞으로 튀어 나오고 있음) 초기에 카메라의 위치는 다음과 같이 설정되어 있는데,

```
cam.pos[0] = 0.0, cam.pos[1] = 0.0,
cam.pos[2] = 500.0;
```

이때의 `cam.uaxis[]` 벡터의 세 원소 값을 기술하라.

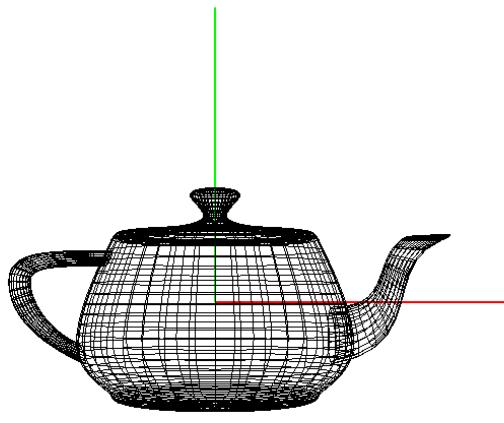


Figure 8: 물주전자 그리기

- (d) 다음은 스페셜 컬백 함수의 일부이다. 이 코드에서 문맥상 `set_rotate_mat(cam.mat);` 문장은 `cam.mat[]` 배열에 어떤 값을 어떻게 넣어주는 역할을 할지 정확히 기술하라.

```
:
case GLUT_KEY_DOWN: {
 float c, s, tx, ty;
 c = cos(3.141592*45.0/180.0);
 s = sin(3.141592*45.0/180.0);
 tx = c*cam.naxis[2] - s*cam.naxis[0];
 ty = s*cam.naxis[2] + c*cam.naxis[0];
 cam.naxis[2] = tx, cam.naxis[0] = ty;
 tx = c*cam.uaxis[2] - s*cam.uaxis[0];
 ty = s*cam.uaxis[2] + c*cam.uaxis[0];
 cam.uaxis[2] = tx, cam.uaxis[0] = ty;
 cam.pos[0] = 500.0*cam.naxis[0];
 cam.pos[1] = 500.0*cam.naxis[1];
 cam.pos[2] = 500.0*cam.naxis[2];
```

```
 set_rotate_mat(cam.mat);
}
glutPostRedisplay();
break;
:
```

- (e) 위 스페셜 컬백 함수를 사용할 경우, 아래 화살표 키를 누를 때마다 화면에서 물주전자가 어떤 식으로 움직일지를 적절한 수치를 사용하여 정확히 기술하라.

10. 그림 9는 아래에 주어진 디스플레이 컬백 함수를 통하여 세상 좌표계(WC)에서 동일한 소 모델을 사용하여 렌더링한 모습을 도시하고 있다.

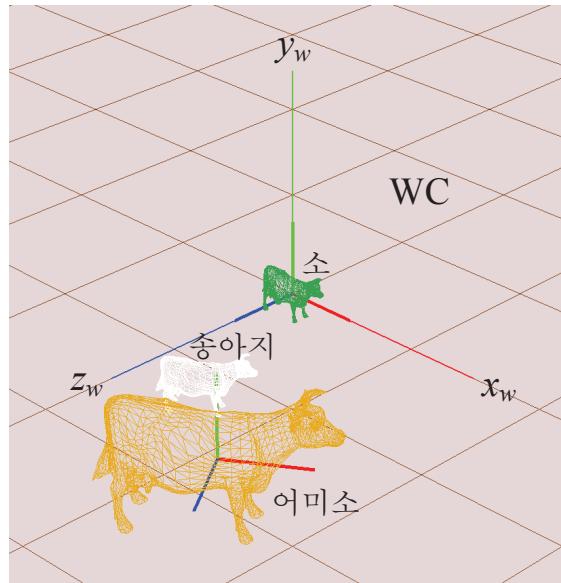


Figure 9: 간단한 계층적 모델링

```
void render(void) {
 glClear(GL_COLOR_BUFFER_BIT);

 glMatrixMode(GL_MODELVIEW);
 draw_floor(0.533, 0.271, 0.075);
 draw_axes(3.0, 1.5); // WC 좌표축
 draw_axes(1.0, 5.0);
 draw_cow(0.243, 0.627, 0.333); // 소
 glPushMatrix();
 (A)
 glScalef(1.1, 1.1, 1.1);
 draw_axes(1.0, 5.0);
 glPushMatrix();
 glTranslatef(-0.1, 0.88, 0.0);
 draw_cow(1.0, 1.0, 1.0);
```

```

glPopMatrix();
glScalef(2.5, 2.5, 2.5);
draw_cow(0.914, 0.671, 0.090);
glPopMatrix();

glutSwapBuffers();
}

```

- (a) 이 프로그램에서 세 개의 `draw_cow(*)`; 문장들은 각각 소, 송아지, 그리고 어미소를 그려주고 있다. 여기서 송아지를 그려주는 문장은 어떤 것인지 해당 함수의 세 개의 인자를 통하여 답하라.
- (b) 프로그램 문맥상 어미소는 원점 주변의 소의 몇 배 크기를 가질까?
- (c) 지금 타이머 콜백 함수에 의해 변수 `angle`의 값이 1씩 증가하면서 (0과 359 사이에서 순환) 위의 디스플레이 콜백 함수가 호출되고 있다. 어미소와 송아지가 세상 좌표계의  $y_w$ 축 둘레로 반경 4인 원을 따라 같이 회전을 하게 하게 하기 위하여 (A) 지점에 필요한 내용을 C/C++ 언어 문법에 맞게 OpenGL API 함수 호출을 통하여 구현하라 (어미소와 송아지의 얼굴이 진행 방향을 가리킴).
- (d) 위 코드에서 소, 어미소, 송아지 등 각 물체를 그려주는 문장이 수행되는 시점에는 각 물체 자신만의 좌표계가 설정되어 있다고 할 수 있다. 이때 송아지를 그려주는 문장이 수행되는 시점에서 한 꼭지점에 어떤 기하 변환을 가하면 어미소를 그려주는 문장이 수행되는 시점의 좌표계로 변환할 수 있을까?  $T(t_x, t_y, t_z)$ ,  $R(\alpha, r_x, r_y, r_z)$ ,  $S(s_x, s_y, s_z)$  행렬과 그것들의 역행렬을 적절히 사용하여 그 기하 변환을 합성하라.

#### 11. 그림 10에 주어진 GLUT API 함수를 통한 윈도우 프로그래밍 코드를 보고 답하라.

- (a) 이 프로그램을 처음 수행시키면 가로-세로 1000 픽셀 크기의 윈도우가 화면에 도시된다. 이 윈도우 안에는 두 개의 사각형이 그려지는데, 전체 윈도우 안에서의 이 두 사각형의 정확한 크기와 위치를 그리고 이 두 사각형으로 인하여 생성되는 세 영역의 초기 색깔을 명확히 기술하라 (이 세 영역에 대하여 적절히 (A), (B), 그리고 (C)로 이름을 붙이고, 이후 문제에서는 이 영역 이름을 사용하여 답할것).
- (b) 이 프로그램에서는 특정 윈도우 영역에서 메뉴를 사용할 수 있는데, (i) 그 영역이 어

느 영역인지 (자신이 붙인 이름을 사용하여) 밝히고, (ii) 어떠한 사용자 인터액션을 통해 메뉴를 화면에 도시하는지, 그리고 (iii) 각 메뉴 요소 선택을 통하여 어떤 일을 할 수 있는지 명확히 기술하라.

- (c) 어떤 특정 영역은 왼쪽 마우스 버튼 클릭에 대하여 반응을 한다. 어떤 영역인지 (자신이 붙인 이름을 사용하여) 기술하라.
- (d) 그 영역에서 왼쪽 마우스 버튼을 눌렀다가 뗄 때 어떤 영역의 내용이 어떻게 바뀌는지 정확히 기술하라. 그 어떤 작용에 대하여 필요하다면 마우스 커서의 위치를 정확히 기술할 것.
- (e) 만약 9번과 10번 문장 사이에 다음 문장을 삽입하고,

```
glutSetWindow(SubWindow0);
```

20번 문장을 다음 두 문장으로 대치 할 경우에 대하여,

```
glutSetWindow(SubWindow0);
glClearColor(0.0, 1.0, 0.0, 1.0);
```

바로 위 문제에 대하여 답하라.

<수고 많았습니다!>

```
01 :

02: int MainWindow, SubWindow0, SubWindow1;

03: void mouse(int button, int state, int x, int y) {
04: if ((button == GLUT_LEFT_BUTTON) && (state == GLUT_DOWN)) {
05: if (y <= 200) {
06: glClearColor(0.0, 0.0, 0.0, 1.0);
07: glutPostRedisplay();
08: }
09: else {
10: glClearColor(1.0, 1.0, 1.0, 1.0);
11: glutPostRedisplay();
12: }
13: }
14: else if ((button == GLUT_LEFT_BUTTON) && (state == GLUT_UP)) {
15: if (y <= 200) {
16: glClearColor(0.0, 0.0, 1.0, 1.0);
17: glutPostRedisplay();
18: }
19: else {
20: glClearColor(0.0, 0.0, 1.0, 1.0);
21: glutPostRedisplay();
22: }
23: }
24: }

25: void display(void) { glClear(GL_COLOR_BUFFER_BIT); glFlush(); }

26: void hello(int value) {
27: if (value) { glClearColor(1.0, 1.0, 0.0, 1.0); glutPostRedisplay(); }
28: else { glClearColor(1.0, 0.0, 0.0, 1.0); glutPostRedisplay(); }
29: }

30: void main (int argc, char **argv) {
31: glutInit(&argc, argv); glutInitDisplayMode(GLUT_RGBA); glutInitWindowSize(1000,1000);

32: MainWindow = glutCreateWindow("2014 CSE3170 Midterm Exam");
33: glutCreateMenu(hello);
34: glutAddMenuEntry("Boy", 0); glutAddMenuEntry("Girl", 1);
35: glutAttachMenu(GLUT_RIGHT_BUTTON);

36: glutDisplayFunc(display);
37: glClearColor(1.0, 0.0, 0.0, 1.0);

38: SubWindow0 = glutCreateSubWindow(MainWindow, 100, 100, 400, 200);
39: glutDisplayFunc(display);
40: glClearColor(0.0, 1.0, 0.0, 1.0);

41: SubWindow1 = glutCreateSubWindow(MainWindow, 600, 500, 200, 400);
42: glutDisplayFunc(display);
43: glutMouseFunc(mouse);
44: glClearColor(0.0, 0.0, 1.0, 1.0);

45: glutMainLoop();
46: }
```

Figure 10: GLUT 함수를 통한 윈도우 프로그래밍 예

## [CSE4170: 기초 컴퓨터 그래픽스]

### 중간고사

(담당교수: 임인성)

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 마지막 공간에 답이 있다고 명기한 후 기술할 것. 그 외에 연습지에 기술한 내용은 답안으로 인정 안함.

1. 2차원 아핀 변환인 이동 변환  $T(t_x, t_y)$ , 크기 변환  $S(s_x, s_y)$ , 그리고 회전 변환  $R(\theta)$ 에 해당하는 3행 3열 행렬들을 고려하자.

- (a) 3행 3열 행렬  $R(\theta)$ 를 기술하라.
- (b)  $R(\theta)S(1, -1) = S(1, -1)R(\theta^*)$ 라 할 때,  $\theta^*$ 의 값을 유도하라. (힌트: 이 문제의 등식에서  $R(\theta^*)$  행렬을 계산한 후,  $\theta^*$ 를 구할 것)
- (c)  $T(t_x, t_y)S(1, -1) = S(1, -1)T(t_x^*, t_y^*)$ 라 할 때,  $t_x^*$ 와  $t_y^*$ 의 값을 유도하라.
- (d)  $R(\theta)T(t_x, t_y) = T(t_x^*, t_y^*)R(\theta)$ 라 할 때,  $t_x^*$ 와  $t_y^*$ 의 값을 유도하라.
- (e) 임의의 점  $(x, y)$ 를  $(-y, x)$ 로 변환해주는 3행 3열의 기하 변환 행렬  $M$ 을 위의 기본 변환 행렬들의 곱으로 표현한 후, 최종 3행 3열 행렬의 내용을 기술하라. (힌트: 위의 (b), (c), (d)에서 유도한 사실을 이용할 것)
- (f) 그림 1(a)의 직사각형 영역의 내용을 (b)의 직사각형 영역으로 매핑해주는 3행 3열의 기하 변환 행렬  $M$ 을 위의 기본 변환 행렬들의 곱으로 표현한 후, 최종 3행 3열 행렬의 내용을 기술하라.

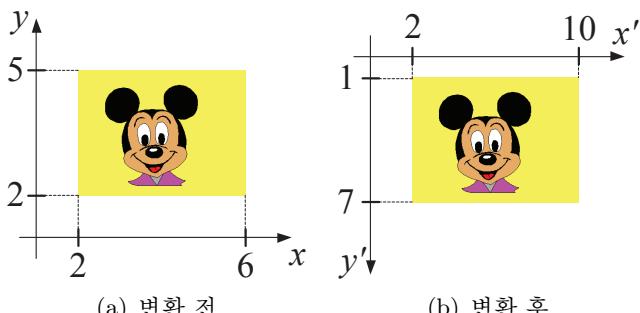


Figure 1: 2차원 윈도우 매핑

2. 아래의 코드는 그림 2에 도시한 원점 주변의 0번 비행기와 1사분면의 1번 비행기를 그려주는 OpenGL 코드이다.

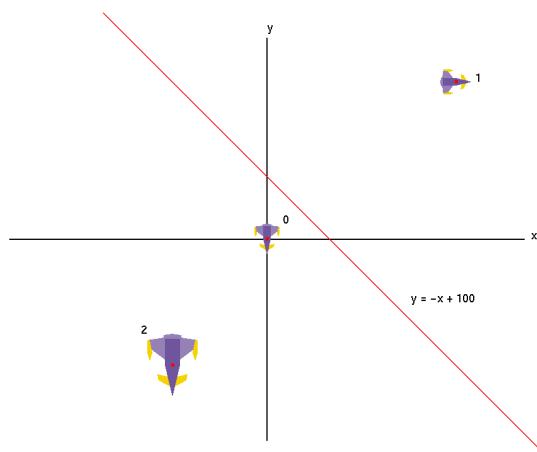


Figure 2: 2차원 기하 변환

```
draw_airplane(); // airplane 0
```

```
glPushMatrix();
glTranslatef(300, 250, 0.0);
glRotatef(90.0, 0.0, 0.0, 1.0);
draw_airplane(); // airplane 1
glPopMatrix();
```

- (a) 이제 이 그림에서처럼 1번 비행기를 2배 확대하여 직선  $y = -x + 100$ 에 대하여 반사시켜 3사분면의 2번 비행기와 같이 그려주려 한다. 아래의 코드는 이를 위한 코드인데, (A)에 들어갈 내용을 5개의 OpenGL API 함수 호출을 통하여 C/C++ 언어 문법에 맞게 기술하라.

```
glPushMatrix();
(A)
glTranslatef(300.0, 250.0, 0.0);
```

```

glRotatef(90.0, 0.0, 0.0, 1.0);
glScalef(2.0, 2.0, 1.0);
draw_airplane(); // airplane 2
glPopMatrix();

```

- (b) 위의 문제와 동일한 내용을 아래와 같은 방식으로 구현하려 한다. 이때 공란으로 되어있는 각 OpenGL API 함수의 인자를 기술하라. (힌트: 1번 문제의 답을 잘 이용할 것)

```

glPushMatrix();
glTranslatef(___, ___, 0.0);
glRotatef(___, ___, ___, ___);
glScalef(___, ___, 1.0);
draw_airplane(); // airplane 2
glPopMatrix();

```

- (c) 그림 3은 아래의 코드의 for-loop에서  $i$ 가 3일 때까지 그림을 그려준 상태이다. 이때  $i$ 가 6일 때 이 원뿔이 어느 지점에 어떻게 그려질지 그 모습을 가급적 정확히 그려라. ( $i$ 가 6일 때만의 물체의 모습을 그릴것)

```

glPushMatrix();
	glColor3f(1.0, 0.0, 0.0);
	glRotatef(-90.0, 1.0, 0.0, 0.0);
	// 원점 주변 원뿔
	glutWireCone(50.0, 100.0, 10, 10);
	glPopMatrix();

for (i = 0; i <= 6; i++) {
 glPushMatrix();
 glTranslatef(300.0, 0.0, 0.0);
 glRotatef(30.0*i, 1.0, 0.0, 0.0);
 glTranslatef(0.0, 250.0, 0.0);
 glRotatef(30.0*i, 0.0, 0.0, 1.0);
 glColor3f(0.0, 0.0, 1.0);
 glRotatef(-90.0, 1.0, 0.0, 0.0);
 glutWireCone(50.0, 100.0, 10, 10);
 glPopMatrix();
}

```

3. 그림 4에는 원점을 중심으로, 각각 세 개의 벡터  $u = (u_x \ u_y \ u_z)^t$ ,  $v = (v_x \ v_y \ v_z)^t$ ,  $n = (n_x \ n_y \ n_z)^t$  와  $u' = (u'_x \ u'_y \ u'_z)^t$ ,  $v' = (v'_x \ v'_y \ v'_z)^t$ ,  $n' = (n'_x \ n'_y \ n'_z)^t$ 에 의해 정의가 되는 두 개의 프레임이 도시되어 있다 (여기서 각 프레임의 세 벡터는 서로 수직인 단위 벡터들임). 이때 Frame 1을 Frame 2로 맞추어 주는 아핀변환에 해당하는 4

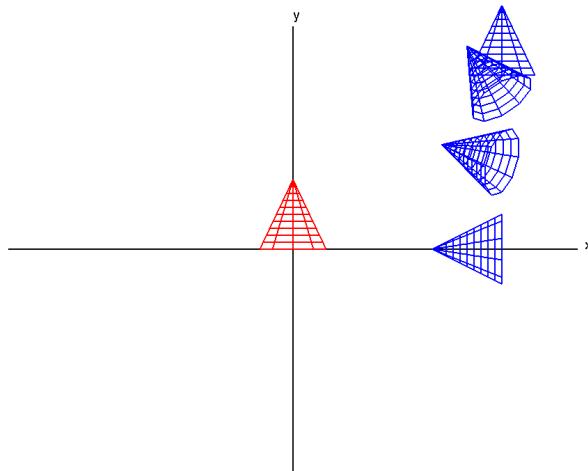


Figure 3: 3차원 기하 변환

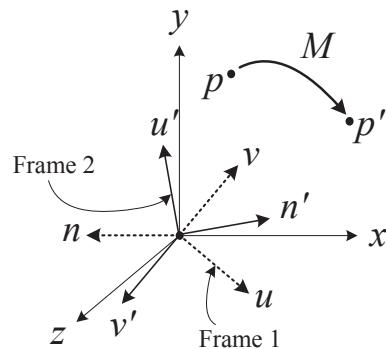


Figure 4: 프레임간의 좌표 변환

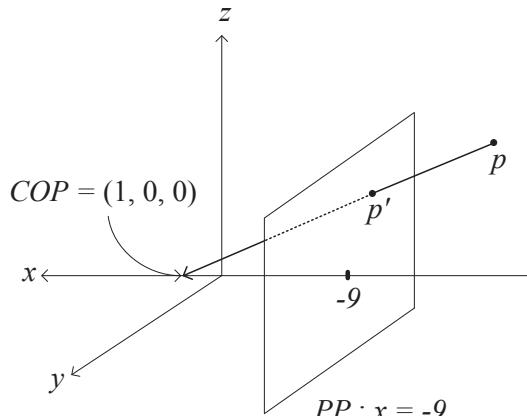


Figure 5: 원근 투영 변환

행 4열 행렬  $M$ 을 두 개의 회전 변환의 곱으로 표현하라. 최종 변환 행렬이 아니라 곱해지는 각 행렬들의 16개 원소들을 정확히 기술할 것.

4. 그림 5에서 주어진 점  $p = (x \ y \ z \ 1)^t$ 을  $p' = (x' \ y' \ z' \ 1)^t$ 로 변환해주는 4행 4열의 원근 투영 변환 행렬  $M$ 을 기술하라.

## 5. 다음은 뷰잉 변환에 관한 문제이다.

- (a) 그림 6에는 카메라의 위치와 방향을 설정해주는 프레임이 도시되어 있다. 이 경우에 해당하는 뷰잉변환을 설정해주는 OpenGL 코드를 `glTranslatef(x, y, z)`, `glScalef(x, y, z)`, 그리고 `glRotatef(a, x, y, z)` 등의 함수만을 사용하여 (가급적 적은 회수의 함수 호출을 통하여) 구현하라.

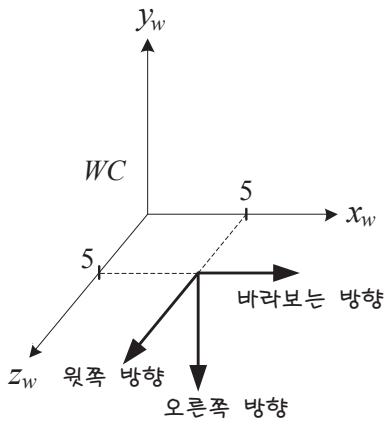


Figure 6: 카메라의 배치

- (b) 한 점  $p$ 와 서로 수직이고 길이가 1인 세 개의 벡터  $u$  (카메라 기준 오른쪽 방향),  $v$  (카메라 기준 위쪽 방향),  $n$  (카메라에서 세상을 바라보는 정반대 방향)으로 정의되는 카메라 프레임을 생각하자. 지금 비행기 물체를 자신의 모델링 좌표계 (Modeling Coordinate)에서 설계하고 있다. 카메라 프레임이 원점에서  $u$ ,  $v$ ,  $n$  벡터가 각각 모델링 좌표계의  $x$ ,  $y$ ,  $z$  축과 일치하여 있는 상태에서 모델링 좌표계의  $z$ 축 둘레로 90도 회전한 후  $x$ 축 방향으로 10만큼 이동하여 카메라 프레임을 조종석에 배치하였다고 하자. 이제 비행기 물체에 대해  $y$ 축 둘레로 90도만큼 회전시킨후,  $y$ 축으로 -10만큼, 그리고  $z$ 축으로 5만큼 이동시켜 세상 좌표계 (World Coordinate)로 배치하였다고 하자. 이때에 해당하는 뷰잉 변환을 다음과 같이 OpenGL API 함수를 사용하여 구현하려 한다.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
(B)
```

여기서 (B)에 들어갈 내용을 네 개의 OpenGL API 함수 호출을 통하여 구현하

라.

- (c) 바로 위 문제에서의 뷰잉 변환을 다음과 같이 구현하려 한다.

```
GLfloat m[16];
:
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMultMatrixf(m);
glTranslatef((a), (b), (c));
```

이때 (a), (b), (c)의 값과 배열  $m$ 에 저장되어야 할 16개 원소 값을 순서대로 기술하라.

6. 지금 아래처럼 카메라 변수 `cam`을 정의한 후,

```
typedef struct _cam {
 float pos[3], uaxis[3], vaxis[3],
 naxis[3];
 GLfloat mat[16];
 GLdouble fovy, aspect, near_c,
 far_c;
} Cam;
Cam cam;
```

디스플레이 컬백 함수에서 다음과 같이 물체를 그리려 한다(여기서 `sfactor`는 `float` 타입의 전역 변수임).

```
void display (void) {
 glClear(GL_COLOR_BUFFER_BIT);
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
 // Line (a)
 glMultMatrixf(cam.mat);
 glTranslatef(-cam.pos[0],
 -cam.pos[1], -cam.pos[2]);
 // Line (b)
 glPushMatrix();
 glScalef(sfactor, sfactor,
 sfactor);
 // Line (c)
 draw_teapot();
 glPopMatrix();
 glFlush();
}
```

- (a) 보편적인 관점에서 Line (a)와 Line (b) 지점은 각각 눈 좌표계 (Eye Coordinate), 모델

- 링 좌표계, 세상 좌표계 중 어느 좌표계에서의 의미를 가질까?
- (b) 다음과 같은 함수를 사용하여 세상 좌표계의 좌표축을 그려주려 한다.

```
void draw_axes(void) {
 glLineWidth(2.0);
 glBegin(GL_LINES);
 glColor3f(1.0, 0.0, 0.0); // x축
 glVertex3f(0.0, 0.0, 0.0);
 glVertex3f(150.0, 0.0, 0.0);
 :
 glEnd();
 glLineWidth(1.0);
}
```

이 함수를 Line (a), Line (b), Line (c) 중 어느 시점에서 호출을 해야할까?

- (c) 문맥상 `set_rotate_mat(cam.mat);` 문장은 `cam.mat[]` 배열에 어떤 값들을 어떻게 넣어주는 역할을 할지 정확히 기술하라.
- (d) 그림 7은 이 프로그램의 초기 렌더링 결과를 보여주고 있다. (여기서 세상 좌표계의  $x$ 축과  $y$ 축이 각각 오른쪽과 위쪽 방향으로 그려져 있고,  $z$ 축은 화면 앞으로 튀어 나오고 있음) 초기에 카메라의 위치는 다음과 같이 설정되어 있는데,

```
cam.pos[0] = 0.0, cam.pos[1] = 0.0,
cam.pos[2] = 500.0;
```

이때의 `cam.naxis[]` 벡터의 세 원소 값을 기술하라.

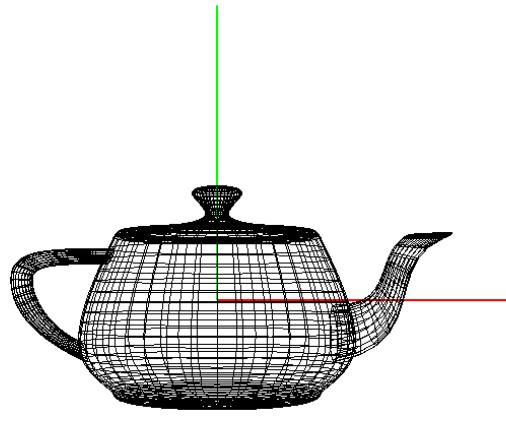


Figure 7: 물주전자 그리기

- (e) 다음은 스페셜 컬백 함수의 일부이다.

```
:
case GLUT_KEY_DOWN: {
 float c, s, tx, ty;
 c = cos(3.141592*45.0/180.0);
 s = sin(3.141592*45.0/180.0);
 tx = c*cam.naxis[2] - s*cam.naxis[0];
 ty = s*cam.naxis[2] + c*cam.naxis[0];
 cam.naxis[2] = tx, cam.naxis[0] = ty;
 tx = c*cam.uaxis[2] - s*cam.uaxis[0];
 ty = s*cam.uaxis[2] + c*cam.uaxis[0];
 cam.uaxis[2] = tx, cam.uaxis[0] = ty;
 cam.pos[0] = 500.0*cam.naxis[0];
 cam.pos[1] = 500.0*cam.naxis[1];
 cam.pos[2] = 500.0*cam.naxis[2];
 set_rotate_mat(cam.mat);
}
glutPostRedisplay();
break;
:
```

아래 화살표 키를 누를 때마다 화면에서 물주전자가 어떤 식으로 움직일지를 적절한 수치를 사용하여 정확히 기술하라.

7. 그림 8은 아래와 같은 어떤 디스플레이 컬백 함수를 통하여 렌더링한 모습을 도시하고 있다. (여기서 세상 좌표계의  $x$ 축은 오른쪽 아래로,  $y$ 축은 위쪽을 향하고 있음)

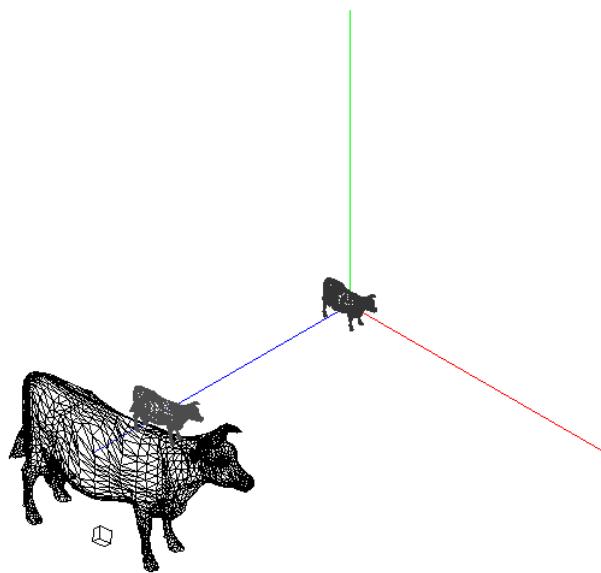


Figure 8: 계층적 모델링

```

 :
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(10.0, 10.0, 10.0, 0.0, 0.0,
 0.0, 0.0, 1.0, 0.0);
draw_cow(0.2, 0.2, 0.2); // 원점 주변 소
glPushMatrix();
// Line (a)
glScalef(4.0, 4.0, 4.0);
draw_cow(0.0, 0.0, 0.0); // 어미소
glPushMatrix();
glRotatef(10.0*angle, 1.0, 0.0, 0.0);
glTranslatef(-0.1, 0.0, 0.3);
glScalef(0.025, 0.025, 0.025);
draw_box(); // 상자
glPopMatrix();
glPushMatrix();
glTranslate(0.15, 0.3, 0.0);
glScalef(0.3, 0.3, 0.3);
draw_cow(0.0, 0.0, 1.0); // 송아지
glPopMatrix();
glPopMatrix();
:

```

- (a) 프로그램 문맥상 송아지는 원점 주변 소의 몇 배 크기를 가질까?
- (b) 지금 타이머 콜백 함수에 의해 변수 `angle`의 값이 1씩 증가하면서 (0과 359 사이 순환) 위의 디스플레이 콜백 함수가 호출되고 있다. 어미소, 송아지, 상자 물체들이 세상 좌표계의  $y$ 축 둘레로 반경 5인 원을 따라 같이 회전을 하게 하게 하기 위하여 Line (a) 지점에 필요한 OpenGL 함수를 기술하라.
- (c) 이때 상자는 어떤 물체를 기준으로 어떠한 방식으로 어떤 속도로 움직일까?
- (d) 위 프로그램에서 제거를 해도 아무런 문제가 없는 문장을 모두 정확히 기술하라 (예를 들어, “// 상자 문장 위의 `glScalef(0.025, 0.025, 0.025);` 문장”처럼 기술하고, 주어진 코드 뒤에 다른 물체를 그리는 코드가 올 수 있음).
- (e) 위 코드에서 원점 주변 소, 어미소, 송아지, 상자 등 각 문장이 수행되는 시점에는 자신만의 좌표계가 설정되어 있다고 할 수 있다. 이때 송아지 문장이 수행되는 시점에서 한 꼭지점에 어떤 기하 변환을 가하면 상자 문장이 수행되는 시점의 좌표계로 변환할 수 있을까?  $T(t_x, t_y, t_z)$ ,  $R(\alpha, r_x, r_y, r_z)$ ,

$S(s_x, s_y, s_z)$  행렬과 그것들의 역행렬을 적절히 사용하여 그 기하 변환을 합성하라.

#### 8. 다음 단답식 문제에 답하라.

- (a) 3차원 투영공간의 점  $(6.0, -6.0, -9.0, -3.0)$ 에 해당하는 아핀공간의 점의 좌표  $(x, y, z)$ 를 기술하라.
- (b) RGB 모델로 값이  $(0.3, 0.5, 1.0)$ 인 색깔을 CMY 모델로 나타내면?
- (c) 다음 행렬의 역행렬  $M^{-1}$ 의 내용을 기술하라.

$$M = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- (d) 어떤 4행 4열 행렬이 아핀 변환 행렬이 되기 위한 조건을 기술하라.
- (e) OpenGL의 뷰잉 파이프라인에서 ‘카메라의 위치와 방향을 설정’해주는 변환은 정확히 어느 좌표계에서 어느 좌표계로 보내주는 변환인가?
- (f) 3차원 공간의 두 벡터  $p = (p_x \ p_y \ p_z)^t$ 와  $q = (q_x \ q_y \ q_z)^t$ 에 대하여 외적 (cross product) 연산을 가한 벡터  $r = p \times q = (r_x \ r_y \ r_z)^t$ 의  $y$  좌표  $r_y$ 의 값을 기술하라.
- (g) `gluLookAt(-1.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.9, 0.0);` 문장 수행 시 계산이 되는 뷰잉 변환 행렬  $M_V = (m_{ij})$ , ( $i, j = 1, 2, 3, 4$ )의 세 번째 행의 원소들  $m_{31}, m_{32}, m_{33}, m_{34}$ 의 내용을 기술하라.
- (h) 3차원 아핀 변환을 나타내는 4행 4열 행렬이 변환 후에도 물체의 크기와 모양을 보존해주기 위한 조건을 정확히 기술하라.
- (i) 주어진 회전 변환  $R$ 과 크기 변환  $S$ 간에 교환 법칙이 성립하기 위한 조건은 무엇일까?
- (j) 만약 여러분이 `glTranslatef(x, y, z)` 함수를 가장 적은 회수의 산술 연산만 사용하여 구현한다고 할 때, 덧셈/뺄셈, 곱셈, 그리고 나눗셈 연산이 각각 몇 번씩 필요할까? (여기서 스택에 있는 행렬은 임의의 값을 가질 수 있다고 가정함)
- (k) 두 직선  $3x + 2y + 5 = 0$ 과  $6x + 4y + 5 = 0$ 의 교점에 대한 투영 공간에서의 동차 좌표를 상수만 사용하여 기술하라.
- (l) 사진 촬영시 피사체를 배치하는 과정을 OpenGL 렌더링에서는 어떤 변환을 통하여 구현하는가?

<수고 많았습니다!>

## [CSE4170: 기초 컴퓨터 그래픽스]

### 중간고사

(담당교수: 임인성)

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒷쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.

1. 2차원 기하 변환 중 이동 변환  $T(t_x, t_y)$ , 크기 변환  $S(s_x, s_y)$ , 그리고 회전 변환  $R(\theta)$ 에 해당하는 3행 3열 행렬들을 고려하자.

- (a) 임의의 점  $(x, y)$ 를  $(-y, -x)$ 로 변환해주는 3행 3열의 기하 변환 행렬  $M_1$ 을 위의 기본 변환 행렬들의 합성을 통하여 표현하라. 최종 행렬의 내용도 정확히 기술할 것.
- (b) 임의의 점  $(x, y)$ 를  $(y, -x)$ 로 변환해주는 3행 3열의 기하 변환 행렬  $M_2$ 를 위의 기본 변환 행렬들의 합성을 통하여 표현하라. 최종 행렬의 내용도 정확히 기술할 것.
- (c)  $T(t_x, t_y)S(1, -1) = S(1, -1)T(\alpha, \beta)$ 라 할 때,  $\alpha$ 와  $\beta$ 의 값을?
- (d)  $R(\theta)S(1, -1) = S(1, -1)R(\gamma)$ 라 할 때,  $\gamma$ 의 값을?
- (e) 그림 1(a)의 직사각형 영역의 내용을 (b)의 직사각형 영역으로 매핑해주는 3행 3열의 기하 변환 행렬  $M_3$ 를 위의 기본 변환 행렬들을 사용하여 합성하라. 최종 행렬 값을 정확히 기술할 것.

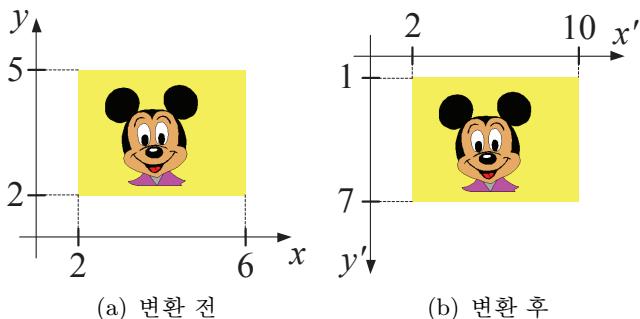


Figure 1: 2차원 윈도우 매핑

2. 그림 2에서 오른쪽에 있는 비행기는 세상 좌표계 공간에서 8자 모양의 궤적을 따라 회전을 하고 있다 (아래의 프로그램에서 Line (B)의 `draw_airplane()` 함수를 호출할 때 그려짐). 또한 원점 근처에 있는 비행기는 이 비행기가 자신의 모델링 좌표계 있는 (즉 아래의 프로그램에서 아무런 모델링 변환 없이 Line (A)의 `draw_airplane()` 함수를 호출할 때 그려지는) 모습에 해당한다. 참고로 `angle2`와 `phase` 변수는 정수 타입의 변수로 0으로 초기화되어 있고, 각 원의 반지름은 `0.2*win_height`임.

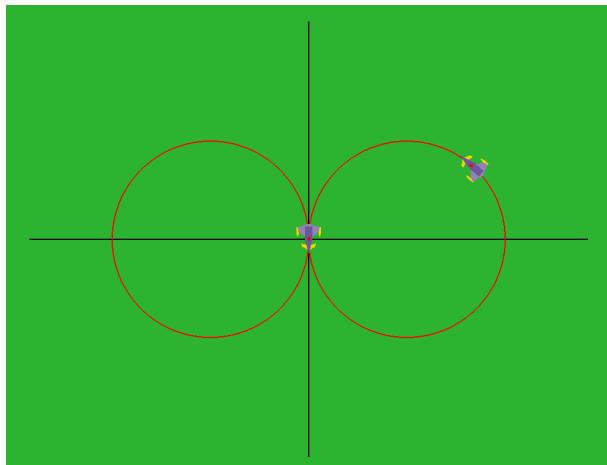


Figure 2: 비행기 애니메이션

```
#define TO_RAD 3.141592/180.0
void timer(int value) {
 glutTimerFunc(15, timer, 2);
 angle2 = (angle2 + 1) % 180;
 if (angle2 == 0) phase = (phase + 1) % 4;
 trans2_x = 0.2*win_height
 *(cos((double) TO_RAD*angle2) + 1.0);
 trans2_y = 0.2*win_height
 *sin((double) TO_RAD*angle2);
 glutPostRedisplay();
}

void display (void) {
 glClear(GL_COLOR_BUFFER_BIT);
```

```
draw_axes(); draw_path();
draw_airplane(); // Line (A)
```

```
glPushMatrix();
switch(phase) {
 case 1:
 glTranslatef(-0.4*win_height, 0.0, 0.0);
 glScalef(1.0, -1.0, 1.0);
 break;
 case 2:
 glScalef((B) , 1.0);
 break;
 case 3:
 glTranslatef(0.2*win_height, 0.0, 0.0);
 glScalef(-1.0, 1.0, 1.0);
 glTranslatef(-0.2*win_height, 0.0, 0.0);
 glScalef(1.0, -1.0, 1.0);
}
glTranslatef(trans2_x, trans2_y, 0.0);
glRotatef((A));
draw_airplane(); // Line (B)
glPopMatrix();
glutSwapBuffers();
}
```

- (a) 프로그램 매크로 상 이 그림이 그려진 순간의 변수 phase의 값은 얼마일까?  
 (b) 프로그램 매크로 상 (A)에 들어갈 내용을 C 언어 문법에 맞게 정확히 기술하라.  
 (c) 프로그램 매크로 상 (B)에 들어갈 두 개의 값을 기술하라.
3. 그림 3에는 카메라의 위치와 방향을 설정해주는 프레임이 도시되어 있다. 이 경우에 해당하는 뷰잉변환을 설정해주는 OpenGL 코드를 `glTranslatef(x, y, z), glScalef(x, y, z),` 그리고 `glRotatef(a, x, y, z)` 등의 함수만을 사용하여 (가급적 적은 회수의 함수 호출을 통하여) 구현하라.
4. 그림 4에는 점  $q = (q_x \ q_y \ q_z)^t$ 을 중심으로, 각각 세 개의 벡터  $u = (u_x \ u_y \ u_z)^t$ ,  $v = (v_x \ v_y \ v_z)^t$ ,  $n = (n_x \ n_y \ n_z)^t$ 와  $u' = (u'_x \ u'_y \ u'_z)^t$ ,  $v' = (v'_x \ v'_y \ v'_z)^t$ ,  $n' = (n'_x \ n'_y \ n'_z)^t$ 에 의해 정의가 되는 두 개의 프레임이 도시되어 있다 (여기서 각 프레임의 세 벡터는 서로 수직인 단위 벡터들임).

- (a) 지금 Frame 1을 Frame 2로 맞추어 주는 아핀변환에 해당하는 4행 4열 행렬  $M$ 을 이동변환과 회전 변환을 적절히 합성하여 구하려 한다. 이때 회전 변환을 두 번 적용할

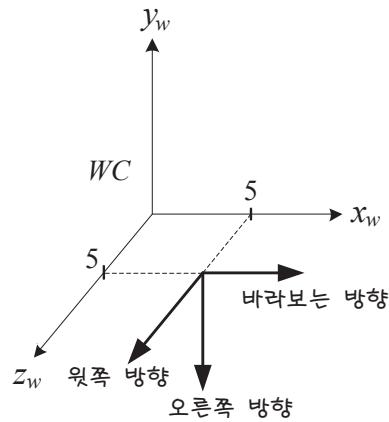


Figure 3: 간단한 뷰잉 변환

경우의  $M$ 을 4행 4열 기하 변환 행렬들의 곱으로 표현하라. 최종 변환 행렬이 아니라 곱해지는 각 요소 행렬들의 16개 원소를 정확히 기술할 것.

- (b) 위의 문제에서 회전 변환을 한 번 적용할 경우의  $M$ 을 4행 4열 행렬의 곱으로 표현하라. 마찬가지로 곱해지는 각 행렬의 16개 원소를 정확히 기술하고, 특히 회전 변환 행렬의 내용은 여섯 개의 벡터  $u, v, n, u', v', n'$ 들의 곱  $\cdot$ 을 사용하여 간결하게 표현하라 (각 벡터는 3행 1열의 벡터임을 명심하고, 예를 들어,  $u \cdot v^t$ 와 같은 방식의 곱셈 사용).

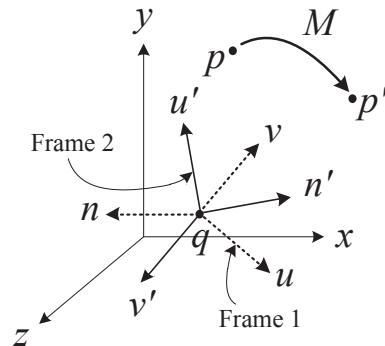


Figure 4: 프레임 변환 문제

5. 다음은 어떤 프로그램의 일부이다.
- ```
typedef struct _cam {
    float pos[3];
    float uaxis[3], vaxis[3], naxis[3];
    GLfloat mat[16];
    int move;
    GLdouble fovy, aspect, near_c, far_c;
} Cam;
Cam cam;
:
```

```

#define M(row,col) m[col*4+row]
void set_rotate_mat(GLfloat *m) {
    M(0,0) = ...; M(0,1) = ...; M(0,2) = ...;
    M(1,0) = ...; M(1,1) = ...; M(1,2) = ...;
    M(2,0) = ...; M(2,1) = ...; M(2,2) = ...;
    M(0,3) = M(1,3) = M(2,3) = 0.0;
    M(3,0) = M(3,1) = M(3,2) = 0.0;
    M(3,3) = 1.0;
}

void render(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glMultMatrixf(cam.mat);
    glTranslatef(..., ..., ...);
    draw_axis();
    draw_world(); // Modeling Transform here.
    glutSwapBuffers();
}

#define CAM_TSPEED 0.05
void renew_cam_pos_x(int del) {
    cam.pos[0] += CAM_TSPEED*del*( (A) );
    cam.pos[1] += CAM_TSPEED*del*( (B) );
    cam.pos[2] += CAM_TSPEED*del*( (C) );
}
void renew_cam_pos_y(int del) { ... }
void renew_cam_pos_z(int del) { ... }

#define TO_RADIAN 0.01745329
void get_rotation_mat(float x, float y,
                      float z, float angle, float m[3][3]) {
    :
}

void renew_cam_ori_x(int angle) {
    float m[3][3], tmpX, tmpY, tmpZ;
    get_rotation_mat(cam.uaxis[0],
                     cam.uaxis[1], cam.uaxis[2],
                     CAM_RSPEED*angle, m);
    cam.vaxis[0] = m[0][0]*(tmpX=cam.vaxis[0])
    + m[0][1]*(tmpY=cam.vaxis[1])
    + m[0][2]*(tmpZ=cam.vaxis[2]);
    cam.vaxis[1] = m[1][0]*tmpX + m[1][1]*tmpY
    + m[1][2]*tmpZ;
    cam.vaxis[2] = m[2][0]*tmpX + m[2][1]*tmpY
    + m[2][2]*tmpZ;
    cam.naxis[0] = ...;
    cam.naxis[1] = ...;
    cam.naxis[2] = ...;
    set_rotate_mat(cam.mat);
}

void renew_cam_ori_y(int angle) { ... }

```

void renew_cam_ori_z(int angle) { ... }

- 프로그램 문맥 상 투영 변환과 직접적인 관련이 있는 변수들을 모두 나열하라.
- `set_rotate_mat(*)`은 뷰잉 변환과 관련된 행렬에 대한 함수이다. 프로그램 문맥 상 $M(0,0) = \dots$, $M(1,0) = \dots$, $M(2,0) = \dots$; 문장에는 어떤 내용이 들어가야 할지 C언어 문법에 맞게 기술하라.
- 프로그램 문맥 상 `get_rotation_mat(*)` 함수는 정확히 어떤 수학적인 내용을 어떤 인자를 통하여 받아들여, 어떤 수학적인 내용을 어떤 인자로 통하여 돌려주는지를 정확히 기술하라.
- 디스플레이 컬백 함수인 `render()` 함수 내부에서 `glTranslatef(*)` 함수 호출 시 필요한 세 인자를 C언어 문법에 맞게 기술하라.
- `renew_cam_pos_x(*)` 함수는 점 `cam.pos`와 세 벡터 `cam.uaxis`, `cam.vaxis`, `cam.naxis`로 정의된 카메라 프레임을 `cam.uaxis` 방향을 따라 `del` 값이 양수일 때 오른쪽으로, 음수일 때는 왼쪽으로 이동시켜주는 작업을 한다. 이러한 목적을 달성하도록 (A), (B), 그리고 (C)에 들어갈 내용을 C언어 문법에 맞게 정확히 기술하라.
- `renew_cam_ori_x(*)` 함수도 카메라에 프레임에 대하여 어떤 조작을 가하고 있는데 그것이 무엇인지 정확히 기술하라.

- 다음은 `void gluPerspective(fovy, asp, n, f);` 함수에 관한 문제이다.

- 그림 5는 이 함수가 정의하는 뷰 볼륨을 도시하고 있다. 눈 좌표계 (EC) 상에서 투영 참조점이 원점인 상황에서 임의의 점 $(x_e \ y_e \ z_e)^t$ 인 점을 $z_e = -n$ 평면, 즉 z_e 축에 수직이고 음의 방향으로 n 만큼 떨어진 평면에 원근 투영한 점을 $(x_e^* \ y_e^* \ -n)^t$ 이라 할 때, x_e^* 와 y_e^* 값을 기술하라.
- 위 함수에 의해 정의되는 뷰 볼륨의 앞면에 해당하는 직사각형 영역 ((0)번 점과 (1)번 점을 포함하는 직사각형)의 네 개의 모서리 중 (0)번 점과 (1)번 점의 좌표를 기술하라. 당연히 z_e 좌표는 $-n$ 이며, 위 함수의 인자 `fovy`, `asp`, `n`, `f` 등과 `tan`, `cot` 등의 삼각 함수를 사용하여 표현하라.

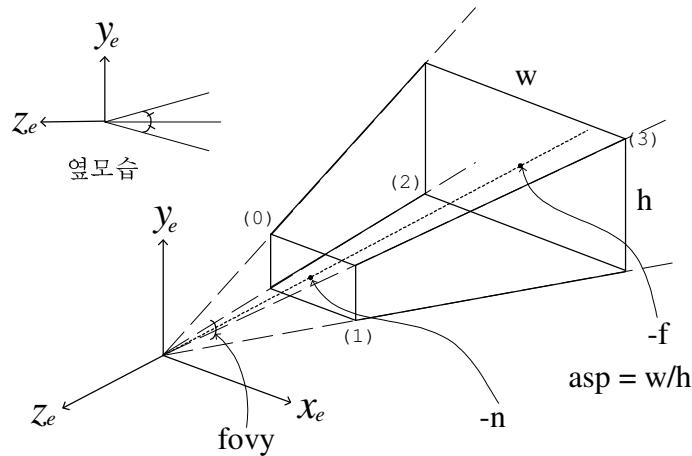


Figure 5: 투영 변환

(c) 뷰 볼륨 안에 존재하는 점을 $z_e = -n$ 평면에 투영한 점 $(x_e^* y_e^* -n)^t$ 은 바로 위 문제의 직사각형 영역 안에 들어오게 되는데, 이 영역을 x 와 y 축 각각에 대해 -1에서 1까지의 구간이 정의하는 정사각형 영역으로 매핑을 해주면, 정규 디바이스 좌표계 (NDC) 상에서의 좌표를 구할 수 있게 된다. 위 두 문제의 결과를 바탕으로 하여, EC 상의 점 $(x_e y_e z_e)^t$ 을 위 함수에 의해 NDC로 변환해준 점의 x, y 좌표 $(x_{nd} y_{nd})^t$ 를 x_e, y_e, z_e , 그리고 위 함수의 인자 $fovy, asp, n, f$ 값과 \tan, \cot 등의 삼각 함수를 사용하여 표현하라. 간단히 유도 과정을 기술할 것.

(d) z_{nd} 의 경우 다음과 같이 됨을 보일 수 있는데,

$$z_{nd} = \frac{-\frac{f+n}{f-n}z_e - \frac{2nf}{f-n}}{-ze}$$

지금까지 구한 모든 값을 사용하여 위 함수가 계산을 해주는 4행 4열 투영 변환 행렬 M_P 를 정확히 기술하라.

(e) 이제 위의 행렬 M_P 에 $(x_e y_e z_e 1)^t$ 을 곱하면 절단 좌표계 (CC) 상의 점 $(x_c y_c z_c w_c)^t$ 로 변환이 되는데 이때 w_c 값은 기하적으로 어떤 정보를 제공할까?

(f) 투영 공간 (projective space)인 CC에서 다시 유클리드 공간인 NDC로 돌아오기 위해 수학적으로 x_c, y_c, z_c 값을 w_c 로 나누어 주는데, 이때 기하적으로 어떤 현상이 발생하는가?

7. 아래 주어진 2차원 기하 변환에 관한 프로그램을 보고 답하라.

```

int iii = 0;
void display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();
    glRotatef(iii*90.0, 0.0, 0.0, 1.0);
    glBegin(GL_TRIANGLES);
    glVertex2f(1.0, 0.0);
    glVertex2f(1.0, 1.0);
    glVertex2f(0.0, 1.0);
    glEnd();
    glPopMatrix();
    glFlush();
}

void mousepress(int button, int state, int x, int y) {
    if ((button == GLUT_RIGHT_BUTTON) &&
        (state == GLUT_DOWN) &&
        (glutGetModifiers() == GLUT_ACTIVE_SHIFT))
        iii = (iii+1)%4;
    glutPostRedisplay();
}

void reshape(int width, int height) {
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glutPostRedisplay();
}

void OpenGLInitandRegisterCallback(void) {
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    glColor3f(1.0, 0.0, 0.0);
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
}

```

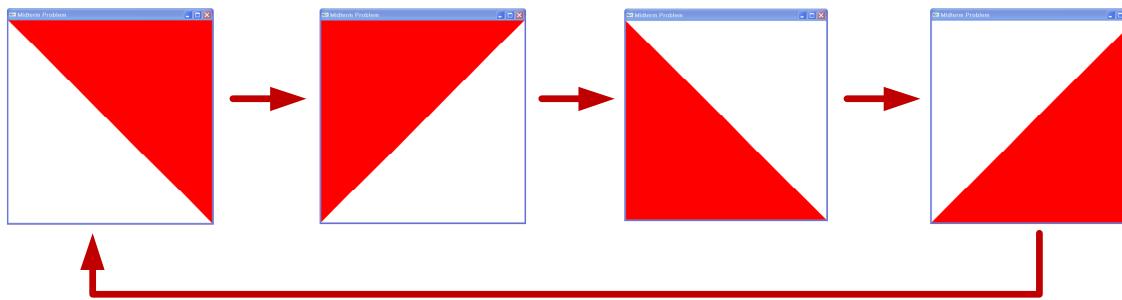


Figure 6: OpenGL을 사용한 2차원 기하 변환

```

glutMouseFunc(mousepress);
}

void main (int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Midterm Problem");
    OpenGLInitandRegisterCallback();
    glutMainLoop();
}

```

- (a) 이 프로그램은 사용자가 어떤 행동 (action)을 취할 때마다, 네 번을 주기로 화면의 내용이 반복된다. 과연 어떤 행동인지 정확히 기술하라.
- (b) 사용자가 위 문제의 행동을 취할 때 화면의 그림이 어떤 식으로 순환하는지, 그림 6과 같은 방식으로 그려라. 가장 처음 상태를 가장 왼쪽에 도시하고, 도형의 모양과 위치를 가급적 정확히 표시하며, 어떤 부분이 적색 영역인지 분명히 밝힐 것.
- (c) 원래의 프로그램 상태에서 `display()` 함수의 `glRotatef(*);` 문장 바로 다음에 다음 문장을 삽입하면 화면의 내용이 어떻게 반복될지, 그림 6과 같은 방식으로 그려라.
- ```

glTranslatef(-1.0, -1.0, 0.0);

```
- (d) 바로 위 문제의 프로그램 상태에서, 즉 이동 변환 관련 문장이 삽입된 상태에서 `reshape()` 함수의 `glutPostRedisplay();` 문장 바로 직전에 아래와 같은 문장을 삽입 할 경우 화면의 내용이 어떻게 반복될지, 그림 6과 같은 방식으로 그려라.
- ```

glOrtho(0.0, 1.0, 0.0, 1.0,
       -1.0, 1.0);

```
8. 다음 단답식 문제에 답하라. 필요할 경우 OC (Object Coordinate), CC, EC, MC, NDC,

WC, WdC (Window Coordinate) 등의 OpenGL 좌표계 이름을 적절히 사용하라.

- 3차원 투영공간의 점 $(6.0, -6.0, -9.0, -3.0)$ 에 해당하는 아핀공간의 점의 좌표 (x, y, z) 를 기술하라.
- RGB 모델로 값이 $(0.3, 0.5, 1.0)$ 인 색깔을 CMY 모델로 나타내면?
- 3차원 아핀변환을 나타내는 4행 4열 행렬 $M = (m_{ij})$, $(i, j = 1, 2, 3, 4)$ 이 변환 후에도 물체의 크기와 모양을 보존해주기 위하여 M 의 어느 부분이 어떤 성질을 가져야 하는지 행렬이나 벡터의 성질을 사용하여 기술하라.
- 다음 행렬의 역행렬 M^{-1} 의 내용을 기술하라.

$$M = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 어떤 4행 4열 행렬 $M = (m_{ij})$, $(i, j = 1, 2, 3, 4)$ 이 아핀 변환 행렬이 되기 위한 조건을 m_{ij} 의 값을 통하여 기술하라.
- 3차원 공간의 $[-1, 1] \times [-1, 1] \times [-1, 1]$ 의 영역만 고려하는 좌표계는 어느 좌표계일까?
- 카메라의 뷰 방향에 대해 꼭지점이 나열된 순서를 통하여 불필요한 삼각형 (예를 들어, 안보이는 뒷면에 해당하는)을 제거할 수 있는데, OpenGL에서는 이러한 과정이 어느 좌표계에서 수행이 될까?
- 정상적인 렌더링 상황에서 3차원 좌표점을 동차좌표 (x, y, z, w) 로 표현할 때 순간적으로 w 가 1이 아닌 값이 나타날 수 있는 OpenGL 좌표계 이름은?
- 원근 투영을 사용하여 렌더링을 할 경우, OpenGL 뷰잉 파이프라인에서 정확히 어느 좌표계에서 어느 좌표계로 넘어갈 때, 원근감이 생성이 되는가?

- (j) OpenGL의 뷔잉 파이프라인에서 ‘카메라의 위치와 방향을 설정’해주는 변환은 정확히 어느 좌표계에서 어느 좌표계로 보내주는 변환인가?
- (k) OpenGL의 뷔잉 파이프라인에서 촬영한 필름을 현상한 후 인화지에 확대/인화하는 과정은 정확히 어느 좌표계에서 어느 좌표계로 보내주는 과정에 해당하는가?
- (l) 만약 여러분이 `glScalef(x, y, z)` 함수를 가장 적은 회수의 덧셈/뺄셈, 곱셈, 그리고 나눗셈 연산을 사용하여 구현한다고 할 때 각각 몇 번씩 수행해야 할까? 여기서 스택에 있는 4행 4열 행렬은 임의의 값을 가질 수 있다고 가정하고, 답은 $+/-=$ 번, $*=$ 번, $/=$ 번과 같이 기술하라.
- (m) 정상적인 렌더링 상황에서 Projection Matrix Stack의 탑에 있는 행렬에 곱해지는 꼭지점 좌표가 존재하는 OpenGL 좌표계 이름은?
- (n) 3차원 공간의 두 벡터 $p = (p_x \ p_y \ p_z)^t$ 와 $q = (q_x \ q_y \ q_z)^t$ 에 대하여 외적 (cross product) 연산을 가한 벡터 $r = p \times q = (r_x \ r_y \ r_z)^t$ 의 y 좌표 r_y 의 값을 기술하라.
- (o) `gluLookAt(-1.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.9, 0.0)`; 문장 수행 시 계산이 되는 뷔잉 변환 행렬 $M_V = (m_{ij})$, ($i, j = 1, 2, 3, 4$)의 세 번째 행의 원소들 $m_{31}, m_{32}, m_{33}, m_{34}$ 의 내용을 기술하라.
9. 시험지 뒤에 첨부한 프로그램은 적절한 모델링 변환을 통하여 자동차를 그려주는 OpenGL 프로그램이다. 이 프로그램을 보면서 답하라.
- (a) 이 프로그램은 그림 7에 주어진 자동차에 대한 트리 구조를 어떤 방식으로 탐색을 하고 있는가? 자료 구조 시간에 배운 용어를 사용할 것.
- (b) 이 프로그램에서 원근 나눗셈과 가장 관련이 있는 문장의 번호는?
- (c) 이 프로그램에서는 사용자가 어떠한 방식으로 자동차를 앞으로 움직이게 할 수 있을까? 정확하게 기술할 것.
- (d) 세상 좌표계 (WC)를 기준으로 할 때 카메라가 세상을 바라보는 방향 벡터를 (필요 시) 이 프로그램에서 사용하는 변수들을 사용하여 표현하라.
- (e) 눈 좌표계 (EC)를 기준으로 할 때 카메라가 세상을 바라보는 방향 벡터를 (필요 시) 이 프로그램에서 사용하는 변수들을 사용하여 표현하라.
- (f) 59번의 이동 변환 관련 문장이 수행되기 직전의 모델뷰 행렬 스택의 내용을 정확하게 그려라. M_V M_P , M_{VP} 와 그림 7의 행렬 기호등을 적절히 사용하라. 현재 `draw_wheel_and_nut(angle)` 함수는 100번 문장에서 호출한 상태임.
- (g) 52번 문장의 `draw_wheel_and_nut(angle)` 함수에서 이상한 부분을 지적하고 수정하라. 이유를 설명할 것.
- (h) 이 프로그램에서 사용하는 변수들을 사용하여 정규 디바이스 좌표계 (NDC)의 좌표 (x_{nd}, y_{nd}) 를 윈도우 좌표계 (WdC)의 좌표 (x_{wd}, y_{wd}) 로 어떻게 변환 시켜주는지 뷔팅 변환(x 와 y 좌표에 대해서만)을 유도하라.
- (i) 이 프로그램에서 뷔팅과 가장 관련이 많은 문장의 번호를 기술하라.
- (j) 이 프로그램에서 자동차를 세상 좌표계로 배치해주는 모델링 변환이 강체 변환인지 아닌지 답하고 그 이유를 간략히 설명하라.

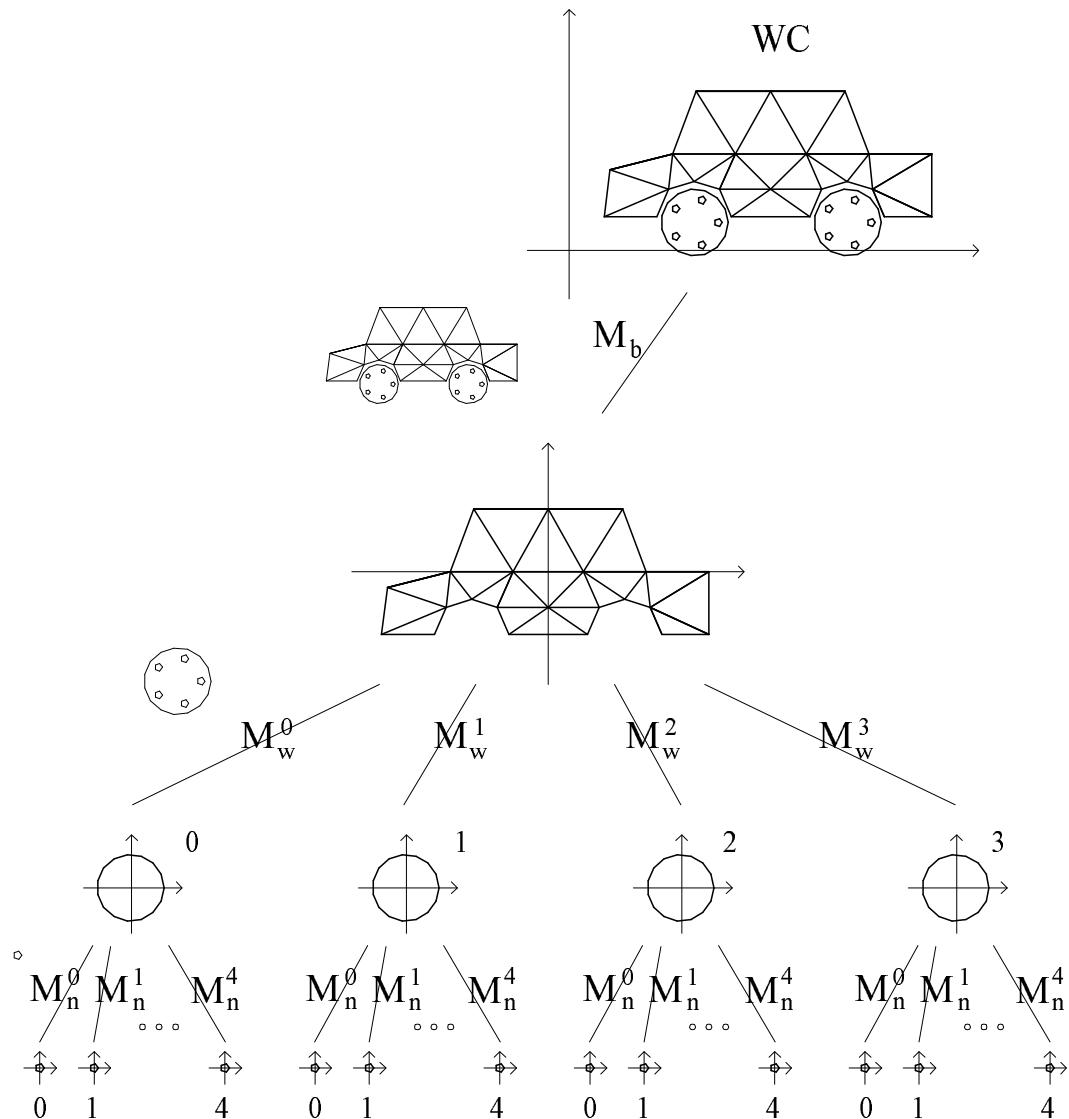


Figure 7: 자동차의 계층적 표현

filename mvcar.cpp

page 1

```

1 #include <stdio.h>
2 #include <math.h>
3 #include <GL/glut.h>
4
5 #define MAX_POLY 200
6 #define MAX_VERT 20
7 #define MAX_PATH 1000
8
9 #define DRAW_CAR_DUMMY 2001
10 #define DRAW_CAR_CORRECT 2002
11
12 typedef struct {
13     int nvertex;
14     float poly[MAX_VERTEX];
15 } mypolygon;
16
17 mypolygon body[MAX_POLY], wheel[MAX_POLY], nut[MAX_POLY];
18 int npolyb, npolyw, npolyn;
19
20 float path[MAX_PATH][3];
21 int npath, path_exist, drawing_state = DRAW_CAR_CORRECT;
22 double dist;
23
24 int prev_i, cur_i = 0;
25 int rightbuttonpressed = 0;
26
27 void read_object(char *file, mypolygon *object, int *npoly);
28 void read_path(char *file, mypolygon *object, int *npoly);
29 void read_ground(char *file);
30 void read_objects(void);
31 void draw_axes(void);
32 void draw_path(void);
33 void draw_ground(void);
34 void draw_body(void);
35 void draw_body(void) {
36     // Draw the body.
37     ...
38 }
39
40 void draw_wheel(float angle) {
41     // Draw the wheel.
42     ...
43 }
44
45 void draw_nut(void) {
46     // Draw the nut.
47     ...
48 }
49
50 #define rad 1.7
51 #define ww 1.0
52 void draw_wheel_and_nut(float angle) {
53     int i;
54     draw_wheel(angle); // draw wheel object
55     for (i = 0; i < 5; i++) {
56         // nut i
57         glPushMatrix();
58         glTranslatef(rad-0.5, 0, ww); // rad = 1.7, ww = 1.0
59         glRotatef(72.0*i, 0.0, 0.0, 1.0);
60         draw_nut(); // draw nut object
61         glPopMatrix();
62     }
63 }
```

- 2012년 4월 26일(목) 오후 7:00 (AS 414) -

© 2012 서강대학교 공과대학 컴퓨터공학과 이인성

page 2

page 2

filename mvcar.cpp

```

127 }
128 void draw_fences(void) {
129 ...
130 }
131 }
132 void draw_world (void) {
133 ...
134 GLfloat m[16];
135 glMatrixMode(GL_MODELVIEW); // Modeling Transformation
136 glLoadIdentity();
137 gluLookAt(-15.0, 20.0, 40.0, path[cur_i][0], 4.89, path[cur_i][2], 0.0, 1.0, 0.0);
138
139 draw_ground();
140 draw_fences();
141 draw_axes();
142 if (path_exist) draw_path();
143
144 set_up_rot_mat(m, cur_i);
145 glPushMatrix();
146 glTranslatef(path[cur_i][0], 4.89, path[cur_i][2]);
147 glMultMatrixf(m);
148 draw_car_dummy();
149 glPopMatrix();
150
151 draw_car_dummy();
152 glPopMatrix();
153 }
154
155 void render(void) {
156 ...
157 glClear(GL_COLOR_BUFFER_BIT);
158 draw_world();
159 glutSwapBuffers();
160 }
161 void keyboard (unsigned char key, int x, int y) {
162 ...
163 }
164 int prevx_mouse;
165 void mousepress(int button, int state, int x, int y) { GLUT_DOWN)
166 if ((button == GLUT_RIGHT_BUTTON) && (state == GLUT_DOWN)) {
167 prevx_mouse = x;
168 rightbuttonpressed = 1;
169 }
170 else if ((button == GLUT_RIGHT_BUTTON) && (state == GLUT_UP))
171 rightbuttonpressed = 0;
172 }
173 }
174 void mousemove(int x, int y) {
175 double deltax;
176 if (rightbuttonpressed) {
177 ...
178 deltax = x - prevx_mouse;
179 prevx_mouse = x;
180 if ((cur_i + deltax > 0) && (cur_i + deltax < npath-1)) {
181 prev_i = cur_i; cur_i += deltax;
182 dist = sqrt((path[cur_i][0]-path[prev_i][0])*(path[cur_i][1]-path[prev_i][1])
183 ...
184 ...
185 ...
186 if (deltax < 0.0) dist *= -1.0;
187 glutPostRedisplay();
188 }
189 }
190 }
191 void reshape(int width, int height) {
192 glViewport(0, 0, width, height);
193 glMatrixMode(GL_PROJECTION);
194 glLoadIdentity();
195 gluPerspective(30.0, width/(double) height, 1.0, 150.0);
196
197 }
198 }
199 void init_OpenGL(void) {
200 ...
201 ...
202 }
203 void init_windows(void) {
204 ...
205 glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
206 glutInitWindowSize(1280, 1024);
207 glutCreateWindow("Car in Hierarchy 2");
208 glutDisplayFunc(render);
209 glutKeyboardFunc(keyboard);
210 glutMouseFunc(mousepress);
211 glutMotionFunc(mousemove);
212 glutReshapeFunc(reshape);
213 }
214 void main(int argc, char **argv) {
215 ...
216 readObjects();
217 glutInit(&argc, argv);
218 init_windows();
219 init_OpenGL();
220 glutMainLoop();
221 }
222 }
```

- 2012년 4월 26일(목) 오후 7:00 (AS 414)

© 2012 서강대학교 공과대학 컴퓨터공학과 임인상

[CSE4170: 기초 컴퓨터 그래픽스]

중간고사

(담당교수: 임인성)

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.

1. 다음은 2차원 기하 변환에 관한 문제이다.

- 점 $(4, 5)$ 를 중심으로 물체의 크기를 두 배 확대해주는 3행 3열의 기하 변환 행렬 M 을 이동 변환 $T(t_x, t_y)$, 크기 변환 $S(s_x, s_y)$, 회전 변환 $R(\theta)$ 등의 기본 변환 행렬을 사용하여 합성하여 작성하라.
 - M 을 세 개의 기본 변환 행렬의 곱으로 표현하라.
 - M 을 두 개의 기본 변환 행렬의 곱으로 표현하라.
- 임의의 점 (x, y) 를 (y, x) 로 변환해주는 3행 3열의 기하 변환 행렬 M 을 위 문제에 주어진 기본 변환 행렬을 사용하여 표현하라.
- $T(t_x, t_y)S(1, -1) = S(1, -1)T(\alpha, \beta)$ 라 할 때, α 와 β 의 값은?
- $R(\theta)S(1, -1) = S(1, -1)R(\gamma)$ 라 할 때, γ 의 값은?
- 임의의 각도 θ 에 대해 $c = \cos(\theta)$ 와 $s = \sin(\theta)$ 라 하자. $R(\theta)T(t_x, t_y) = T(\delta, \epsilon)R(\theta)$ 이라 할 때, δ 와 ϵ 의 값을 t_x, t_y, c, s 등을 사용하여 표현하라.

2. 그림 1과 아래 코드를 고려하자.

```
void display (void) {
    :
    glClear(GL_COLOR_BUFFER_BIT);
    draw_axes_text_line();
    draw_airplane(); // Draw A

    glPushMatrix();
    // Transforms for B
    draw_airplane(); // Draw B
```

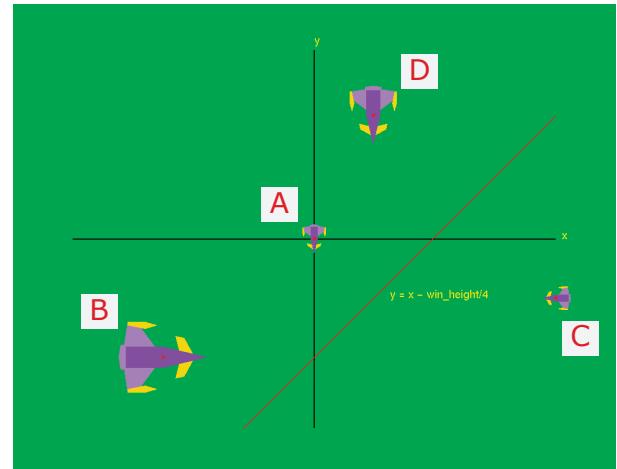


Figure 1: 2차원 기하변환

```
glPopMatrix();

glPushMatrix();
:
draw_airplane(); // Draw C
glPopMatrix();

glPushMatrix();
glTranslatef(H/4.0, 0.0, 0.0);
glRotatef(45.0, 0.0, 0.0, 1.0);
glScalef(sx, sy, 1.0);
glRotatef(-45.0, 0.0, 0.0, 1.0);
glTranslatef(tx, ty, 0.0);
glRotatef(270.0, 0.0, 0.0, 1.0);
glScalef(2.0, 2.0, 1.0);
draw_airplane(); // Draw D
glPopMatrix();
:
```

아래 문제에서 요구하는 2차원 기하 변환을 다음 OpenGL 함수들을 적절히 사용하여 C언어 문법에 맞게 정확히 기술하라. 함수 호출 순서는 OpenGL 프로그래밍 관례에 따르고, 이동

변환이 필요하다면 z 값은 0으로, 크기 변환이 필요하다면 z 값은 1로 설정할 것.

- void glTranslatef(GLfloat x, GLfloat y, GLfloat z);
- void glScalef(GLfloat x, GLfloat y, GLfloat z);
- void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);

- (a) // Draw 'X' 라인의 draw_airplane(); 문장은 각각 이 그림에서 'X'로 표시된 비행기를 그려주고 있다. B 비행기는 A 비행기를 세 배 확대하여 원점 둘레로 시계 방향으로 270도 회전한 후, 그 중심을 $(-W/4.0, -H/4.0)$ 지점으로 이동시켜 그런 상태를 보여주고 있다. 이러한 그림이 그려지도록 // Transforms for B 부분의 내용을 OpenGL 함수를 사용하여 기술하라.
- (b) 비행기 C는 비행기 A를 시계 방향으로 90도 회전한 후, 그 중심을 $(W/2.5, -H/8.0)$ 지점으로 이동시켜 그런 상태를 보여주고 있다. 지금 그러한 비행기 C를 두 배 확대한 후 $y = x - \frac{H}{4.0}$ 둘레로 반사시켜 비행기 D를 그려주려 한다. 이때 필요한 sx, sy 와 tx, ty의 내용을 정확히 기술하라.
- (c) 비행기 D를 그려주는 코드는 아래와 같이 단순화시킬 수 있다.

```
glPushMatrix();
glTranslatef(tx, ty,
0.0);
glRotatef(ra, rx, ry, rz);
glScalef(ssx, ssy, 1.0);
draw_airplane(); // Draw D
glPopMatrix();
```

이때 필요한 tx, ty, ra, rx, ry, rz, ssx, ssy의 내용을 정확히 기술하라.

3. 3차원 공간에 주어진 법선 벡터 (normal vector) $n = (nx, ny, nz)^t$ 에 대하여 회전 변환을 가하는 문제를 고려하자.

- (a) 회전 변환 행렬 M 을 아래와 같이 정의한다면,

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

이때 v_1, v_2, v_3 의 값은 어떤 값을 가질까? 축을 나타냄)

- (b) 이 행렬의 왼쪽-위쪽의 3행 3열 부행렬의 세 개의 열벡터를 각각 r_1, r_2, r_3 라 할 때, 이 세 개의 벡터가 만족하는 수학적인 성질을 정확히 기술하라.

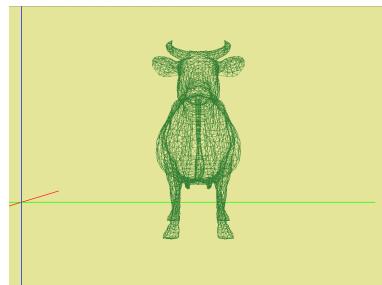
- (c) n 에 회전 변환을 가하려면 $(M^{-1})^t$ 에 n 을 곱하면 된다. 이때 $(M^{-1})^t$ 의 내용을 정확히 기술하라.

4. DOP (Direction of Projection) 가 $(-2, -2, -\sqrt{2})$ 이고 PP (Projection Plane) 가 $2x + 2y + \sqrt{2}z = 1$ 인 평행 투영의 변환 행렬이 다음과 같다고 하자.

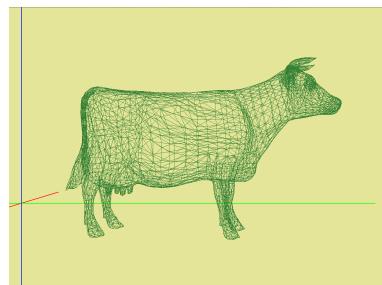
$$M = \begin{bmatrix} e & f & g & h \\ i & j & k & l \\ m & n & o & p \\ q & r & s & t \end{bmatrix}$$

- (a) 네 번째 행의 q, r, s, t 값을 기술하라.
(b) 첫 번째 행의 e, f, g, h 값을 유도하라 (유도 과정을 기술할 것).

5. 다음은 OpenGL을 사용한 3차원 뷰잉에 관련된 문제이다. 아래 그림과 코드를 보고 답하라.



(a) 프로그램 초기 수행 모습 (소가 여러분을 향해 있음)



(b) 'r' 키를 16번 누른 후의 모습 (즉 90도 회전한 모습)

Figure 2: 간단한 3차원 뷰잉 (빨강, 초록, 파랑 색깔의 세 직선은 각각 세상 좌표계에서의 x_w, y_w, z_w 축을 나타냄)

```

void display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(..., ..., ..., ..., ..., ..., ..., ..., ...);
    // ModelView matrix
    draw_axes();
    glColor3f(0.2, 0.5, 0.2);
    glPushMatrix();
    // We need a modeling transform here.
    draw_cow();
    glPopMatrix();
    glFlush();
}

```

- (a) 위의 코드에서 `draw_cow()` 함수는 자신의 모델링 좌표계 (Modeling Coordinate, MC)에서 정의된 소를 구성하는 삼각형을 그려주는 함수이다. 그림 2(a)는 이 소를 각 축 방향으로 5배 확대하여, x 축 둘레로 90도 회전한 후, $(5.0, 5.0, 1.5)$ 만큼 이동시켜 그려준 상태를 보여주고 있다. 이 때 `// We need a modeling transform here.`에 들어갈 내용을 OpenGL 함수를 적절히 사용하여 기술하라 (2번 문제에 기술된 OpenGL 함수 정의를 참고할 것).
- (b) ‘r’ 키를 누를 때마다 `int rotangle = 0;` 과 같이 정의된 전역 변수 (global variable) `rotangle`에 대해 다음과 같은 내용의 코드가 수행된다고 가정하자.

```

case 'r':
    rotangle = (rotangle+5)%360;
    glutPostRedisplay();
    break;

```

이에 반응하여 ‘r’ 키를 한 번씩 누를 때마다, 위의 문제에서와 같이 세상 좌표계 (World Coordinate, WC)에 배치된 소가 이 좌표계에서 $(5.0, 5.0, 1.5)$ 점을 지나고 z_w 축 (그림에서 수직 방향의 축)에 평행한 직선 둘레로 5도씩 회전하도록 하려 한다 (그림 2(b) 참조). 위 문제에서 기술한 여러분의 코드를 어떻게 확장하면 될지 정확히 기술하라.

- (c) 이 문제에서는 카메라가 세상 좌표계의 $(15.0, 5.0, 1.5)$ 지점에서 $(5.0, 5.0, 1.5)$ 지점을 바라보고 있으며, 양의 z_w 방향이 위쪽 방향으로 설정되어 있다. 이에 필요한 `gluLookAt()` 함수의 9개 인자를 정확히 기술하라.

- (d) 이 프로그램이 정상적으로 수행될 때 // `ModelView` matrix 시점에서 모델뷰 행렬 스택의 탑에 있는 4행 4열의 뷰잉 변환 행렬 M_V 의 내용을 정확히 기술하라.
- (e) 이 뷰잉 변환 행렬 M_V 는 세상 좌표계의 벡터 $(5.0, 5.0, 1.5)^t$ 를 눈 좌표계 (Eye Coordinate, EC)의 어떤 점으로 변환해 줄까?
- (f) M_V 는 $M_V = RT$ 와 같이 회전 변환과 이동 변환을 나타내는 두 개의 4행 4열 행렬 R 과 T 의 곱으로 표현이 가능한데, 이 때 R 과 T 의 내용을 정확히 기술하라.
- (g) M_V 의 역행렬을 $M_V^{-1} = T_1 R_1$ 과 같이 이동 변환 행렬 T_1 과 회전 변환 행렬 R_1 의 곱으로 표현할 때 4행 4열 행렬인 R_1 의 내용을 기술하라.
- (h) M_V 의 역행렬을 $M_V^{-1} = R_2 T_2$ 와 같이 회전 변환 행렬 R_2 와 이동 변환 행렬 T_2 의 곱으로 표현할 때 4행 4열 행렬인 T_2 의 내용을 기술하라.

6. OpenGL에서는 그림 3에 도시한 바와 같이 눈 좌표계 (EC)상에서 `glOrtho(*)` 함수를 사용하여 직교 투영을 위한 뷰잉 볼륨을 설정하면, 이 안의 내용이 정규 디바이스 좌표계 (NDC)의 정규화된 정육면체 영역으로 뷰 매핑된다.

- (a) 이때 절단 좌표계 (Clip Coordinate, CC) 상에서의 절단 (clipping) 과정 후 정규 디바이스 좌표계 (Normalized Device Coordinate, NDC)로 투영된 꼭지점의 x_{nd} 와 y_{nd} 의 좌표값은 각각 어떤 범위의 값을 가지 수 있을까?

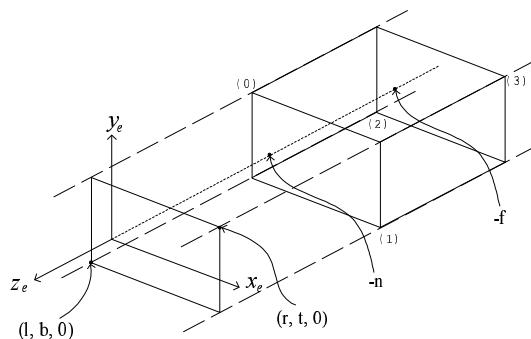


Figure 3: `glOrtho(l, r, b, t, n, f)` 함수

- (b) 이 함수가 호출될 때 생성되는 투영 변환 행렬 M_{Ortho} 를 기본 변환 행렬 $T(x, y, z)$ (이동 변환), $S(x, y, z)$ (크기 변환), $R(angle, x, y, z)$ (회전 변환) 등의 곱

으로 표현하라. 뷰 매핑 과정에서 투영 방향이 반대로 바뀌게 되는 것을 유념할 것.

- (c) M_{ortho} 는 다음과 같이 하나의 행렬로 나타낼 수 있는데, 이 때 α, β, γ 값이 무엇인지 기술하라.

$$M_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \alpha & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \beta & \gamma \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7. 다음은 색깔 혼합에 관한 문제이다. ($c_S \alpha_S$)와 ($c_D \alpha_D$)를 각각 S와 D 이미지의 대응되는 화소의 미리 곱한 색깔(pre-multiplied color)이라 할 때, 두 색깔의 합성을 통하여 생성한 결과 색깔 ($c_O \alpha_O$)는 다음과 같이 표현할 수 있다.

$$\begin{pmatrix} c_O \\ \alpha_O \end{pmatrix} = F_S \begin{pmatrix} c_S \\ \alpha_S \end{pmatrix} + F_D \begin{pmatrix} c_D \\ \alpha_D \end{pmatrix}$$

- (a) 만약 값이 (0.3, 0.3, 0.3, 0.3)인 미리 곱한 색깔로 어떤 화소를 칠한다고 할 때, 이는 그 화소를 어떻게 칠한다는 것인지 정확히 기술하라.
 (b) 그림 4에 도시하는 합성 연산의 경우 F_S 와 F_D 의 값은 각각 얼마인가?

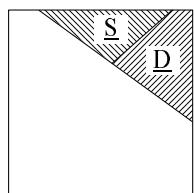


Figure 4: 합성 예 1

- (c) 만약 S over D 연산을 적용한다면, 합성 후 α_O 는 어떤 값을 가질지 그 식을 S와 D의 관련 값을 사용하여 정확히 기술하라.
 (d) 그림 5와 같은 상황을 생각해 보자. 지금 관찰자가 세 개의 물체 M_0, M_1, M_2 를 바라보고 있는데, M_0 는 실제 색깔이 C_0 인 유리로서 뒤에서 들어오는 빛을 $1 - \alpha_0$ 의 비율로 통과시킨다. 한편 M_1 은 색깔이 C_1 이고 빛을 $1 - \alpha_1$ 의 비율 만큼만 통과시키고, 제일 오른쪽에 있는 M_2 는 완전히 불투명한 벽으로서 C_2 의 색깔을 가진다. 이 때 M_0 와 M_1 을 앞에서 뒤로 가면서 (front-to-back order) 합성한다고 할 때의 불투명도 α_{01} 값이 무엇일지 정확히 기술하라.

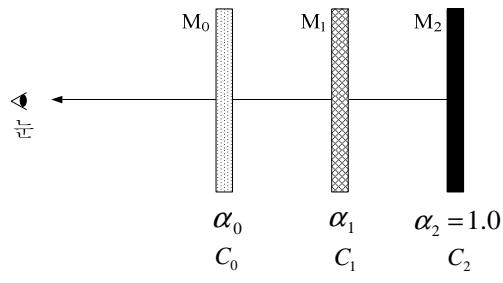


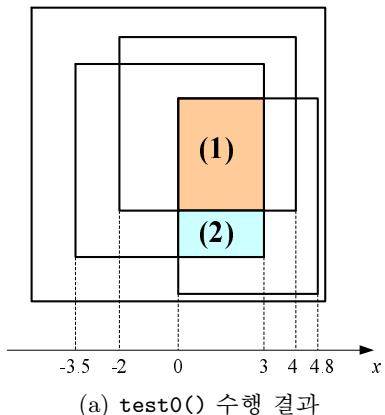
Figure 5: 합성 예 2

- (e) 바로 위 문제에서와 같이 M_0 와 M_1 을 앞에서 뒤로 가면서 합성한다고 할 때의 결과 색깔 C_{01} 값이 무엇일지, 위 문제의 인자를 사용하여 정확히 기술하라. 여기서 C_0, C_1 , 그리고 C_{01} 은 미리 곱한 색깔이 아닌 원래의 RGB 색깔을 의미함.
 (f) 위 문제에서 모든 합성이 끝난 후 결과적으로 눈에 보이는 색깔 C_{012} 는 무엇일지 위 문제의 인자를 사용하여 정확히 기술하라.
 (g) 아래의 프로그램에서 함수 `test0()`을 수행시킬 경우 그림 6(a)와 같이 서로 다른 색깔로 칠해지는 여러 영역을 볼 수 있는데, 이때 (1)번 영역과 (2)번 영역의 RGB 색깔을 정확히 기술하라. (주의: RGB 각 채널 값은 0과 1 사이의 값을 가짐)

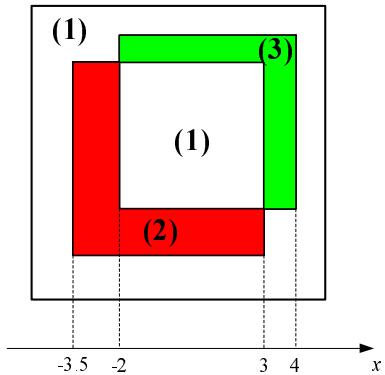
```
void rect(GLfloat l, GLfloat r,
          GLfloat b, GLfloat t, GLfloat R,
          GLfloat G, GLfloat B, GLfloat A) {
    glColor4f(R, G, B, A);
    glBegin(GL_QUADS);
    glVertex2f(l, b); glVertex2f(r, b);
    glVertex2f(r, t); glVertex2f(l, t);
    glEnd();
}

void test0(void) {
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glEnable(GL_BLEND);
    glBlendFunc(GL_ONE, GL_ZERO);
    rect(-2.0, 4.0, -2.0, 4.0, 0.0, 1.0,
         0.0, 1.0);
    glBlendFunc(GL_SRC_ALPHA, GL_DST_ALPHA);
    rect(-3.5, 3.0, -3.5, 3.0, 1.0, 0.0,
         0.0, 1.0);
    rect(0.0, 4.8, -4.8, 2.0, 0.0, 0.0, 1.0,
         1.0);
    glDisable(GL_BLEND);
}

void test1(void) {
```



(a) test0() 수행 결과



(b) test1() 수행 결과

Figure 6: 색깔의 혼합

```

glClearColor(0.0, 0.0, 0.0, 0.0);
glClear(GL_COLOR_BUFFER_BIT);
 glEnable(GL_BLEND);
 glBlendFunc(GL_ZERO, GL_ZERO);
 rect(-2.0, 4.0, -2.0, 4.0, 0.0, 1.0,
 0.0, 1.0);
 glBlendFunc((A), (B));
 rect(-3.5, 3.0, -3.5, 3.0, 1.0, 0.0,
 0.0, 1.0);
 glDisable(GL_BLEND);
}

```

- (h) 다음 함수 `test1()`을 수행시켰을 때 그림 6(b)와 같은 결과를 얻으려면, (A)와 (B)에 어떤 인자값이 설정되어야 할까? (여기서 (1), (2), (3)번 영역의 색깔은 각각 (0, 0, 0), (1, 0, 0), (0, 1, 0)임) 다음 값을 중 적절한 인자를 선택하라.

GL_ZERO, GL_ONE,
GL_SRC_ALPHA,
GL_DST_ALPHA,
GL_ONE_MINUS_SRC_ALPHA,
GL_ONE_MINUS_DST_ALPHA

8. 다음은 계층적 모델링을 통하여 자동차를 그려주는 OpenGL 프로그램으로서, 사용자가 마우스를 사용하여 자동차를 경로를 따라 대화식으로 (interactively) 움직일 때, 네 바퀴가 적절한 속도로 진행 방향으로 회전하며, 특히 두 앞 바퀴는 자동차의 진행 방향에 따라 적절히 좌우로도 회전을 하도록 코딩이 되어 있다 (그림 7 참조). 이 프로그램과 그림 8을 보면서 답하라.

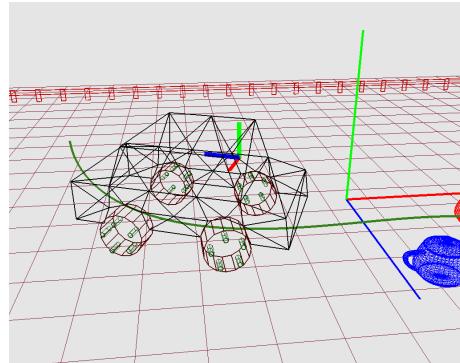


Figure 7: 자동차 그리기

```

void draw_wheel_and_nut(void) {
    int i;
    draw_wheel(); // draw wheel object
    for (i = 0; i < 5; i++) {
        glPushMatrix();
        glRotatef(72.0*i, 0.0, 0.0, 1.0);
        glTranslatef(1.2, 0, 1.0);
        draw_nut(); // draw nut object
        glPopMatrix();
    }
}

void draw_car_correct(void) {
    float angle_1, angle_2;
    angle_1 = wheel_rot_angle_in_1();
    angle_2 = wheel_rot_angle_in_2();
    draw_body(); // draw body object
    glPushMatrix();
    glTranslatef(-3.9, -3.5, 4.5);
    glRotatef(angle_1, 0.0, 1.0, 0.0);
    glRotatef(angle_2, 0.0, 0.0, 1.0);
    draw_wheel_and_nut(); // wheel 0
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-3.9, -3.5, -4.5);
    glRotatef(angle_1, 0.0, 1.0, 0.0);
    glRotatef(angle_2, 0.0, 0.0, 1.0);
    draw_wheel_and_nut(); // wheel 1
    glPopMatrix();
}

```

```

glScalef(1.0, 1.0, -1.0);
draw_wheel_and_nut(); // wheel 1
glPopMatrix();

glPushMatrix();
glTranslatef(3.9, -3.5, 4.5);
// Need a transform
draw_wheel_and_nut(); // wheel 2
glPopMatrix();

glPushMatrix();
glTranslatef(3.9, -3.5, -4.5);
// Need a transform
glScalef(1.0, 1.0, -1.0);
draw_wheel_and_nut(); // wheel 3
glPopMatrix();
}

void draw_world(void) {
    GLfloat m[16];
    :
    set_up_rot_mat(m, cur_i);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(-15.0, 20.0, 40.0,
        path[cur_i][0]+3, 4.89,
        path[cur_i][2]+3, 0.0, 1.0, 0.0);

    glTranslatef(path[cur_i][0], 4.89,
        path[cur_i][2]);
    glMultMatrixf(m);
    draw_car_correct();
}

```

- (a) 이 프로그램에서 자동차를 세상 좌표계로 배치해주는 모델링 변환이 강체 변환인지 아닌지 ‘예/아니오’로 답하고 그 이유를 간략히 설명하라.
- (b) (문제 삭제)
- (c) 눈 좌표계를 기준으로 할 때 카메라가 세상을 바라보는 방향을 벡터로 표현하라.
- (d) 문맥 상 이 코드에서 그림 8의 M_b 행렬의 내용을 결정하는 기하 변환 관련 문장을 모두 정확히 기술하라.
- (e) 문맥 상 이 코드에서 그림 8의 M_n^4 행렬의 내용을 결정하는 기하 변환 관련 문장을 모두 정확히 기술하라.
- (f) 이 프로그램에서는 // wheel 0과 // wheel 1 문장에서 각각 앞 바퀴를 그려주고 있다. 문맥 상 이 문장들의 바로 앞 glRotatef(angle_1, 0.0, 1.0,

0.0); 문장은 어떤 기하 변환을 목적으로 하는지 정확히 기술하라 (그림 8을 잘 보고 답할 것).

- (g) (앞 문제에 이어) 문맥 상 이 두 문장의 바로 앞 glRotatef(angle_2, 0.0, 0.0, 1.0); 문장은 어떤 기하 변환을 목적으로 하는지 정확히 기술하라.
 - (h) 위의 두 문제를 생각할 때, 문맥 상 // wheel 2과 // wheel 3 문장 앞의 // Need a transform 부분에 공통적으로 들어갈 내용을 기술하라.
9. 다음은 OpenGL 시스템의 fixed-function 파이프라인에 관한 단답식 문제이다. 그림 9를 보면서, OC (Object Coordinate), CC, EC, MC, NDC, WC, WdC (Window Coordinate) 등의 좌표계 이름을 참고하여 적절히 답하라.
- (a) 원근 투영을 사용하여 렌더링을 할 경우, 원근감이 생성되는 시점과 가장 관련이 많은 box의 기호를 기술하라.
 - (b) OpenGL에서는 꼭지점이 나열된 순서 정보를 통하여 불필요한 삼각형을 제거할 수 있는데, 이와 가장 관련이 많은 box의 기호를 기술하라.
 - (c) (G) box와 (H) box 사이의 지점에 해당하는 OpenGL 좌표계의 이름은?
 - (d) 정상적인 렌더링 상황에서 좌표점을 동차좌표 (x, y, z, w) 로 표현할 때 순간적으로 w 가 1이 아닌 값이 나타날 수 있는데, 이러한 좌표값을 기반으로 수행되는 어떤 제거 계산과 가장 관련이 높은 box의 기호를 기술하라.
 - (e) 정상적인 기하 변환을 할 경우 전체 파이프라인 과정에서 항상 강체 변환이 사용되는 부분은 어느 좌표계에서 어느 좌표계로의 변환에 해당하는가?
 - (f) OpenGL 렌더링 파이프라인에서 “정규화된 필름”에 해당하는 좌표계는 이 그림에서 어느 box와 어느 box 사이일까?
 - (g) 촬영한 사진을 현상한 후, 인화지에 확대하는 과정과 가장 밀접한 관련이 있는 box의 기호를 기술하라.
 - (h) 정상적인 기하 변환을 할 경우 주어진 물체의 꼭지점에 대해 실제로 모델링 변환이 적용되는 box의 기호를 기술하라.

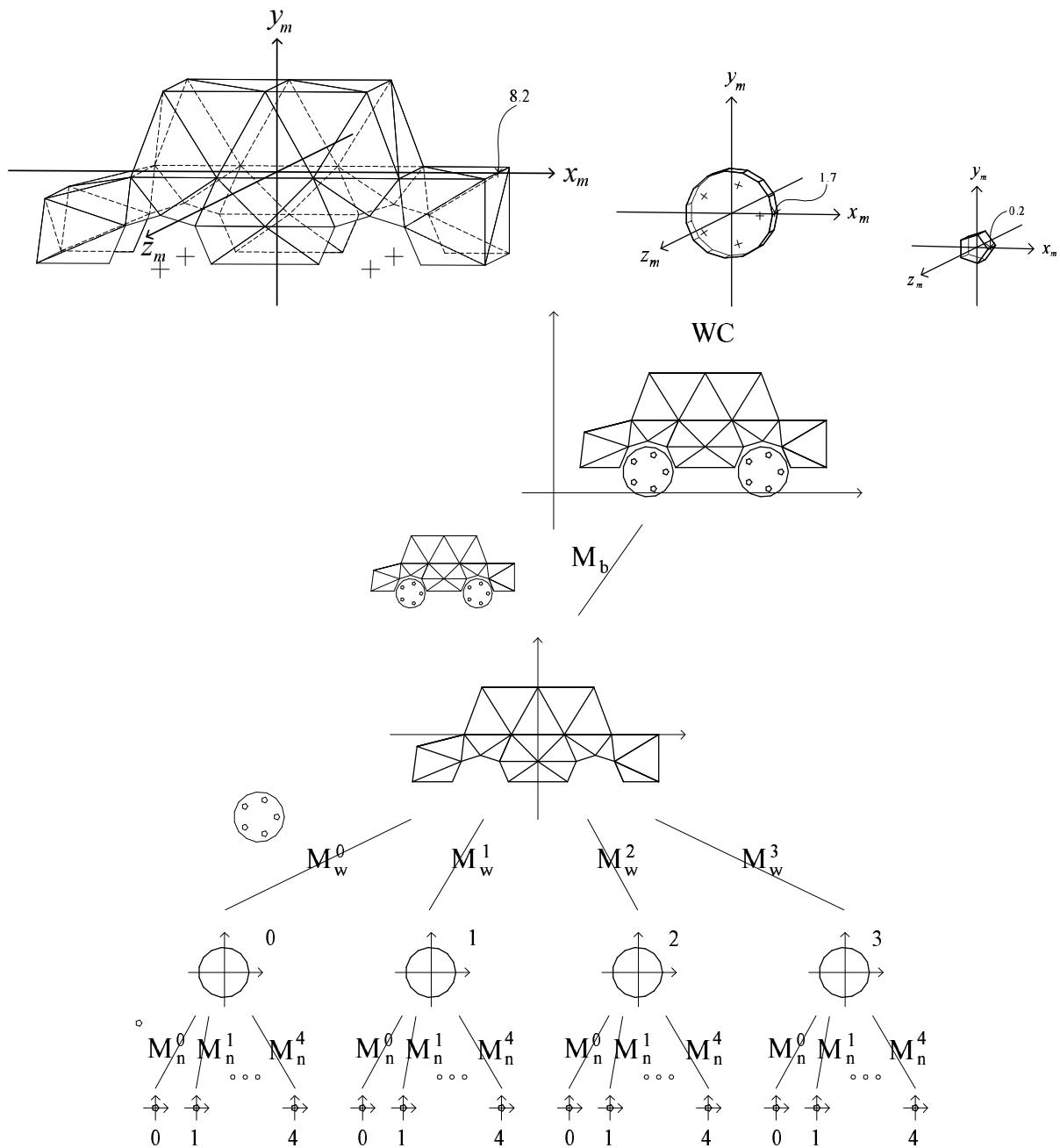


Figure 8: 자동차의 계층적 표현

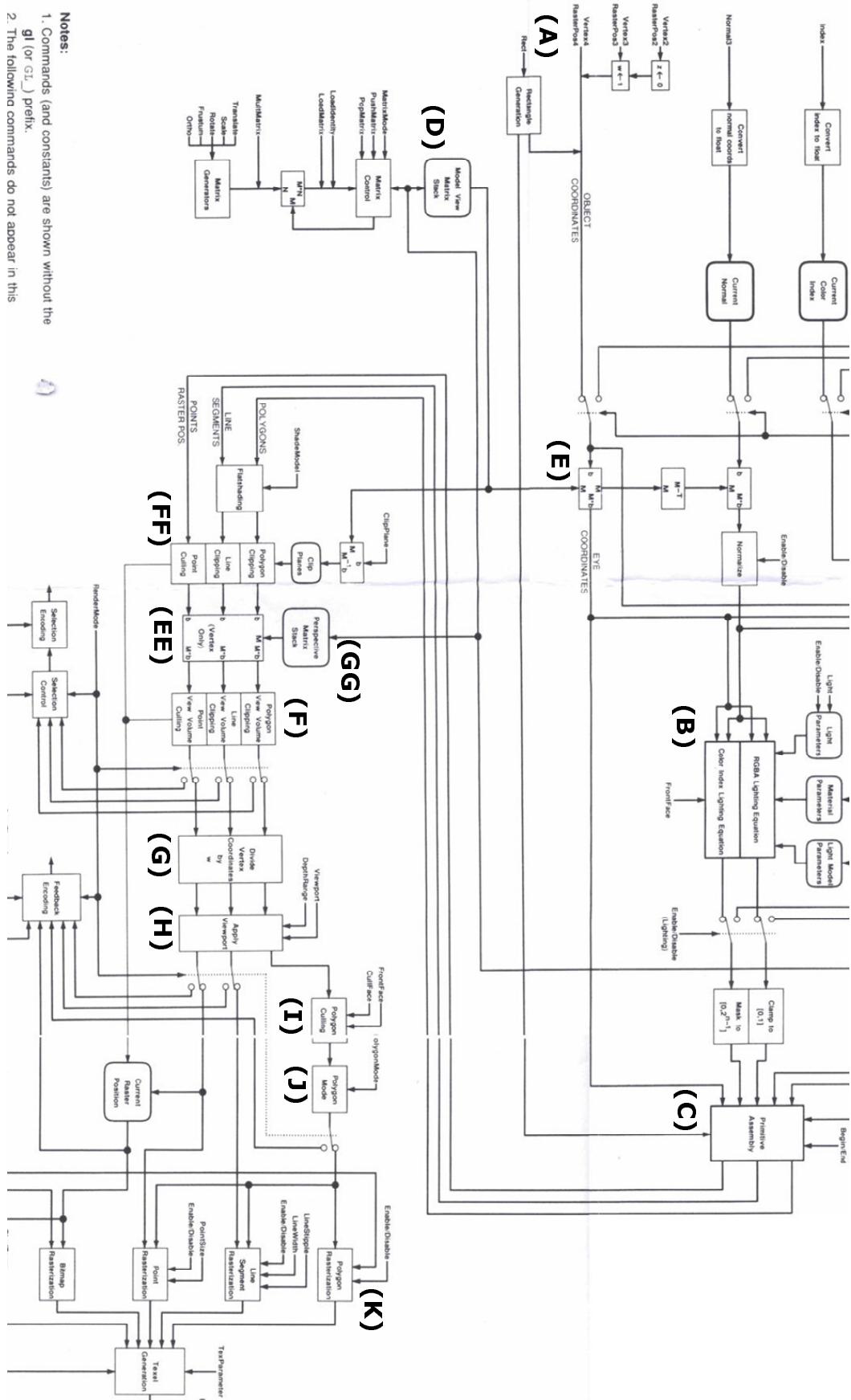


Figure 9: OpenGL fixed-function 렌더링 파이프라인

[CSE4170: 기초 컴퓨터 그래픽스]

중간고사

(담당교수: 임인성)

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.

1. 다음 물음에 답하라.

- 동차 좌표 점 $(-2.0, 0.0, 8.0, -0.5)$ 에 해당하는 유클리드 공간에서의 점의 좌표는?
- RGB 색깔 모델로 표현된 색깔 $(0.7, 0.6, 0.1)$ 을 CMY 색깔 모델을 사용하여 표현했을 때의 값은?
- 3차원 기하 변환을 나타내는 4행 4열 행렬이 아핀 변환인지 아닌지를 어떻게 구별할 수 있을까?
- 투영 참조점이 무한대점 (point at infinity)에 위치한 투영 변환에 관련된 OpenGL 함수의 이름은?
- gluPerspective(fovy, asp, n, f) 함수에 대한 변환 행렬 M_{pers} 는 다음과 같다. 이 행렬을 이용하여 원근 투영 시 EC의 점 (x_e, y_e, z_e) 가 NDC의 점 (x_{nd}, y_{nd}, z_{nd}) 로 변환되는 과정에서 원근감이 생성되는 수학적인 과정을 설명하라.

$$\begin{bmatrix} \frac{\cot(\frac{fovy}{2})}{asp} & 0 & 0 & 0 \\ 0 & \cot(\frac{fovy}{2}) & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2nf}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

2. 다음 행렬 M_2 를 보고 답하라.

$$M_2 = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 3차원 공간에서 동일한 직선상에 존재하는 세 점 p_0, p_1, p_2 에 대하여 $\overline{p_0p_2} : \overline{p_0p_1} = 3 : 1$ 이라 하자. p'_0, p'_1, p'_2 를 각각 위의 세 점을 M_2 를 사용하여 변환한 점이라 할 경우 비율 $\overline{p'_0p'_2} : \overline{p'_0p'_1}$ 은 얼마가 될까?
- M_2 는 강체 변환에 해당하는 행렬일까? ‘예/아니오’로 답하고 그 이유를 수학적으로 밝혀라.
- M_2 를 두 개의 기본 변환 행렬의 곱으로 표현하라.
- M_2 의 역행렬 M_2^{-1} 을 하나의 4행 4열 행렬로 표현하라. 어떤 식으로 계산하였는지 밝힐 것.
- 그림 1에서와 같이 2차원 공간의 점 p 를 직선 L 에 대하여 반사시켜 p' 으로 변환시켜주는 2차원 기하 변환을 다음 OpenGL 함수들을 적절히 사용하여 C언어 문법에 맞게 정확히 기술하라. 함수 호출 순서는 OpenGL 프로그래밍 관례에 따르고, 이동 변환이 필요하다면 z 값은 0으로, 크기 변환이 필요하다면 z 값은 1로 설정할 것.

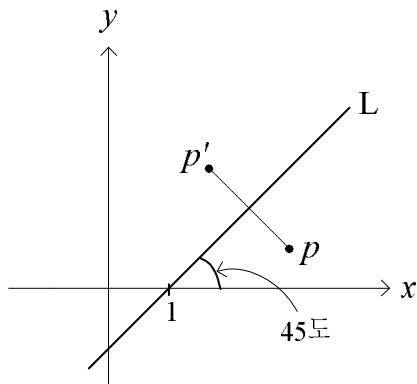


그림 1: 직선에 대한 반사

- void glTranslatef(GLfloat x, GLfloat y, GLfloat z);
 - void glScalef(GLfloat x, GLfloat y, GLfloat z);
 - void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);
4. 그림 2에서와 같이 윈도우의 윈도우의 내용을 오른쪽 윈도우로 매핑해주는 2차원 변환에 대한 3행 3열 행렬 M_4 를 기본 변환 행렬 $T(x, y)$ (이동 변환), $S(x, y)$ (크기 변환), $R(angle)$ (회전 변환) 등의 합성을 통하여 표현하고, 전체 곱 행렬을 구하라.

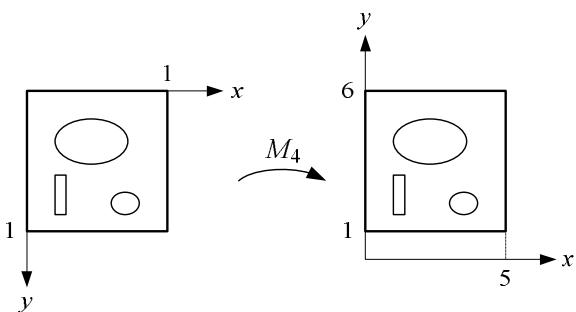


그림 2: 윈도우 매핑

5. OpenGL에서는 그림 3에 도시한 바와 같이 눈좌표계(EC)상에서 glOrtho(*) 함수를 사용하여 직교 투영을 위한 뷔퍼 볼륨을 설정하면, 이 안의 내용이 정규 디바이스 좌표계(NDC)의 정규화된 정육면체 영역으로 뷔

매핑이 된다.

- (a) 이때 절단 좌표계(CC) 상에서의 절단(clipping) 과정 후 NDC로 투영된 꼭지점의 x_{nd} 와 y_{nd} 의 좌표값은 각각 어떤 범위의 값을 가질 수 있을까?

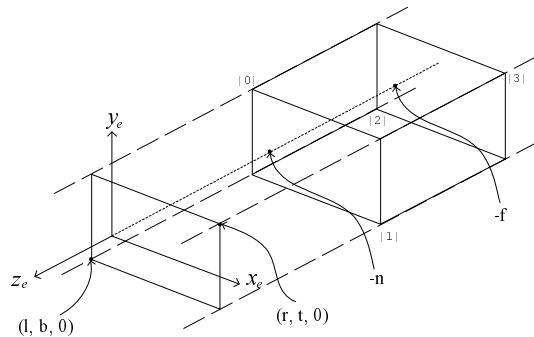


그림 3: glOrtho(l, r, b, t, n, f) 함수

- (b) 이 함수가 호출될 때 생성되는 투영변환 행렬 M_{ortho} 를 기본 변환 행렬 $T(x, y, z)$ (이동 변환), $S(x, y, z)$ (크기 변환), $R(angle, x, y, z)$ (회전 변환) 등의 곱으로 표현하라. 뷔 매핑 과정에서 투영 방향이 반대로 바뀌게 되는 것을 상기할 것.
- (c) M_{ortho} 는 다음과 같이 하나의 행렬로 나타낼 수 있는데, 이 때 α , β , 그리고 γ 값이 무엇인지 기술하라.

$$M_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \alpha & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \beta & \gamma \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6. 다음의 2차원 기하 변환에 관한 프로그램을 보고 답하라.

```
int iii = 0;
void display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glPushMatrix();
    glRotatef(iii*90.0, 0.0, 0.0, 1.0);
```

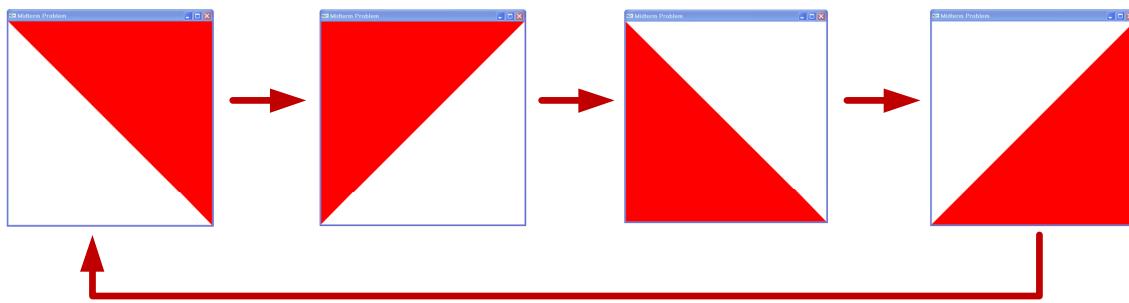


그림 4: OpenGL을 사용한 2차원 기하 변환

```

glTranslatef(-250.0, -250.0, 0.0);
glBegin(GL_TRIANGLES);
    glVertex2f(500.0, 0.0);
    glVertex2f(500.0, 500.0);
    glVertex2f(0.0, 500.0);
glEnd();
glPopMatrix();
glFlush();
}

```

```

void sogang(int button, int state, int x,
int y) {
    if ((button == GLUT_LEFT_BUTTON) &&
        (state == GLUT_DOWN))
        iii = (iii+1)%4;
    glutPostRedisplay();
}

```

```

void university(int width, int height) {
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, (double) width, 0.0,
            (double) height, -1.0, 1.0);
    glutPostRedisplay();
}

```

```

void OpenGLInitandRegisterCallback(void) {
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    glColor3f(1.0, 0.0, 0.0);
    glutDisplayFunc(display);
}

```

```

glutReshapeFunc(university);
glutMouseFunc(sogang);
}

```

```

void main (int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Midterm Problem");
    OpenGLInitandRegisterCallback();
    glutMainLoop();
}

```

- (a) 이 프로그램은 사용자가 어떤 행동 (action)을 취할 때마다, 네 번을 주기로 화면의 내용이 반복된다. 과연 어떤 행동인지 정확히 기술하라.
- (b) 사용자가 위 문제의 행동을 취할 때 화면의 그림이 어떤 식으로 순환하는지, 그림 4와 같은 방식으로 그려라. 도형의 모양과 위치를 가급적 정확히 표시하고 어떤 부분이 적색 영역인지 분명히 밝힐 것.
- (c) 만약 그림 4에 도시한 방식으로 화면이 바뀌도록 하려면 이 프로그램을 어떻게 수정해야하는가? “어느 함수의 어떤 문장과 어떤 문장 사이에 어떤 문장(들)을 삽입함”과 같이 표시하고, 삽입할 문장(들)을 OpenGL 함수들을 적절히 사용하여 C언어 문법에 맞게 정확히 기술하라.

(d) 원래의 프로그램 상태에서 university()

함수의 `glOrtho()` 함수 문장을 아래와 같이 수정할 경우 화면의 내용이 어떻게 반복될지, 그림 4와 같은 방식으로 그려라.

```
glOrtho(0.0, width/2.0, 0.0, height/2.0,
        -1.0, 1.0);
```

(e) 원래의 프로그램 상태에서 university()

함수의 `glOrtho()` 함수 문장 직전에 아래와 같은 문장을 삽입할 경우 화면의 내용이 어떻게 반복될지, 그림 4와 같은 방식으로 그려라.

```
glScalef(0.5, 0.5, 1.0);
```

(힌트: 앞 문제의 직교 투영 변환을 잘 생각해보고, 이 크기 변환이 직교 투영에 어떤 영향을 미치는지를 생각해볼 것)

7. 그림 5에서 주어진 점 $p = (x \ y \ z \ 1)^t$ 을 $p' = (x' \ y' \ z' \ 1)^t$ 로 변환해주는 4행 4열의 원근 투영 변환 행렬 M_7 을 기술하라.

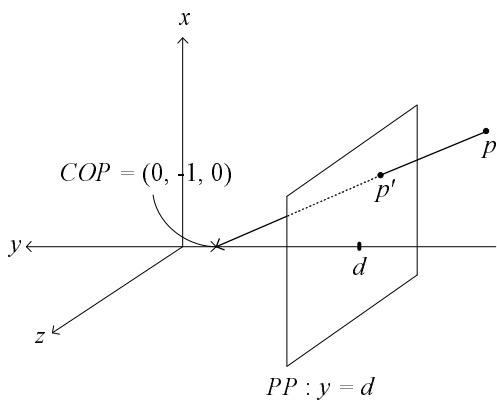


그림 5: 원근 투영 변환 예

8. 다음은 주어진 경로 $p[c_i][0], 0.0, p[c_i][2])$ 를 따라 움직이는 자동차에서 바라본 세상을 렌더링해주는 코드의 일부이다 (그림 6 참조).

```
void set_up_rot_mat(float *m, float
                     *minv, int i) {
    GLfloat u[3], v[3], n[3];
```

```
v[0] = ...; // Line (A)
```

```
if (i == 0) {
    u[0] = p[0][0] - p[1][0];
    u[1] = p[0][1] - p[1][1];
    u[2] = p[0][2] - p[1][2];
}
else {
    u[0] = p[i-1][0] - p[i][0];
    u[1] = p[i-1][1] - p[i][1];
    u[2] = p[i-1][2] - p[i][2];
}
normalize_vec3(u);
cross_prod_vec3(u, v, n);
m[0] = ...; m[15] = 1.0;
minv[0] = ...; minv[15] = 1.0;
```

```
}
```

```
void dispaly(void) {
    GLfloat m[16], minv[16];
    glClear(GL_COLOR_BUFFER_BIT);
    set_up_rot_mat(m, minv, cur_i);
```

(B)

```
draw_axes(); draw_path();
```

```
glPushMatrix();
glTranslatef(p[c_i][0], 4.89,
             p[c_i][2]);
glMultMatrixf(m);
draw_car();
glPopMatrix();
glutSwapBuffers();
```

(a) 지금 `glMultMatrixf(m);`에 해당하는 기하 변환 후, `glTranslatef(p[c_i][0], 4.89, p[c_i][2]);`에 해당하는 기하 변환을 통하여 자동차의 body를 세상 좌표계로 보내주고 있다. 맨 마지막에 `glutSwapBuffers();`를 넣어야 한다.

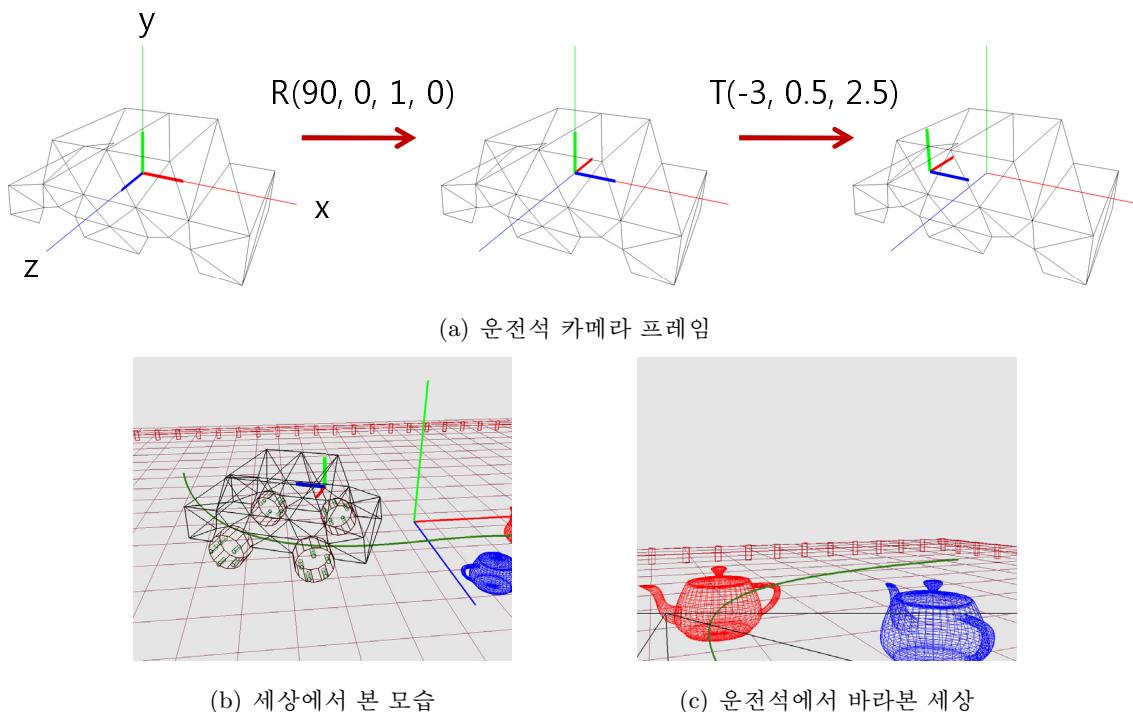


그림 6: 자동차 그리기

타내는 기하 변환은 이동 변환, 크기 변환, 회전 변환 중 어떤 변환에 해당할까?

(b) Line (A)에서는 v 벡터의 값 $v[0]$, $v[1]$, $v[2]$ 의 값을 설정하고 있는데, 각 원소 같은 문맥상 어떤 값으로 설정되어야 할까? 세상 좌표계와 body의 모델링 좌표계 모두 위쪽이 y 축 방향에 해당함.

(c) 문맥 상 `set_up_rot_mat()` 함수 안에서 배열 m 과 $minv$ 의 값을 설정하고 있는데, $m[4]$ 와 $minv[4]$ 에 저장되어야 하는 각 값을 C 언어 문법에 맞게 기술하라. OpenGL에서의 기하 변환 행렬 원소의 저장 순서를 고려할 것.

(d) 다음은 `set_up_rot_mat()` 함수 안에서 호출하는 `cross_prod_vec3()` 함수의 정의이다. $n[1]$ 변수에 설정되는 값이 무엇인지 그 식을 C 언어 문법에 맞게 기술하라.

```
void cross_prod_vec3(float *u, float
*v, float *n) {
    n[0] = ...; n[1] = ...; n[2] = ...;
```

}

(e) 그림 6(a)는 body의 모델링 좌표계 상에서 운전석에 카메라를 배치하는 과정을 보여주고 있다. 만약 자동차를 세상에 배치한 후, 이 카메라에서 세상을 바라보는 장면을 렌더링할 때의 뷰잉 변환 행렬 M_V 를 기본 변환 행렬 $T(x, y, z)$ (이동 변환), $S(x, y, z)$ (크기 변환), $R(angle, x, y, z)$ (회전 변환)이나 이 프로그램의 변수 m 과 $minv$ 가 나타내는 M_m 과 M_{minv} 등의 곱으로 표현하라.

(f) 지금 운전석에 배치한 카메라를 사용하여 렌더링 한다고 가정할 때, `display()` 함수의 (B)에 들어갈 뷰잉 변환 코드를 OpenGL 함수들을 적절히 사용하여 C언어 문법에 맞게 정확히 기술하라.

9. 다음은 OpenGL 프로그램에서 뷰잉 변환을 설정하고 있는 예를 보여주고 있다.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

```
gluLookAt(0.0, 0.0, 10.0,
           10.0, 0.0, 10.0, 0.0, 1.0, 0.0);
```

- (a) 이 코드를 수행할 때 적용되는 뷰잉 변환 행렬 MV 의 내용을 하나의 4행 4열 행렬로 표현하라.

- (b) 원래의 코드에서 `gluLookAt(*)`; 문장을 아래와 같은 두 문장으로 대치하려하는데, 같은 내용의 그림이 그려지도록 하기 위해 서는 각 문장의 인자가 무엇이 되어야 하는지 정확히 기술하라.

```
glRotatef(*, *, *, *);
glTranslatef(*, *, *);
```

10. 다음 OpenGL 코드를 보고 `draw_object()` 함수가 그려주는 네 개의 물체 `OBJ1`, `OBJ2`, `OBJ3`, 그리고 `OBJ4` 간에 존재하는 계층성을 이진 트리로 표현하라.

```
glPushMatrix();
glRotatef(angle, 0.0, 1.0, 0.0);
glTranslatef(0.0, 0.0, 5.0);
glScalef(6.5, 6.5, 6.5);
draw_object(OBJ1);
glPushMatrix();
glRotatef(10.0*angle, 1.0, 0.0, 0.0);
glTranslatef(-0.1, 0.0, 0.3);
glScalef(0.025, 0.025, 0.025);
draw_object(OBJ2);
glRotatef(40.0*angle, 1.0, 0.0, 0.0);
glTranslatef(0.0, 4.0, 0.0);
glScalef(0.4, 0.4, 0.4);
draw_object(OBJ3);
glPopMatrix();
glTranslatef(0.15, 0.3, 0.0);
glScalef(0.3, 0.3, 0.3);
draw_object(OBJ4);
glPopMatrix();
```

11. 다음은 OpenGL 시스템의 fixed-function 파이프라인에 관한 단답식 문제이다. 그림 7을 보고, OC, CC, EC, MC, NDC, WC, WdC 등의 좌표계 이름을 참고하여 적절히 답하라.

- (a) 원근 투영을 사용하여 렌더링을 할 경우, 원근감이 생성되는 시점과 가장 관련이 많은 box의 기호를 기술하라.
- (b) OpenGL에서는 꼭지점이 나열된 순서 정보를 통하여 불필요한 삼각형을 제거할 수 있는데, 이와 가장 관련이 많은 box의 기호를 기술하라.
- (c) (G) box와 (H) box 사이의 지점에 해당하는 OpenGL 좌표계의 이름은?
- (d) 정상적인 렌더링 상황에서 좌표점을 동차 좌표 (x, y, z, w) 로 표현할 때 순간적으로 w 가 1이 아닌 값이 나타날 수 있는데, 이러한 좌표값을 기반으로 수행되는 어떤 제거 계산과 가장 관련이 높은 box의 기호를 기술하라.
- (e) 정상적인 기하 변환을 할 경우 전체 파이프라인 과정에서 항상 강체 변환이 사용되는 부분은 어느 좌표계에서 어느 좌표계로의 변환에 해당하는가?
- (f) OpenGL 렌더링 파이프라인에서 “정규화된 필름”에 해당하는 좌표계는 이 그림에서 어느 box와 어느 box 사이일까?
- (g) 촬영한 사진을 현상한 후, 인화지에 확대하는 과정과 가장 밀접한 관련이 있는 box의 기호를 기술하라.
- (h) (EE) box에서 수행되는 기하 변환은 항상 아핀 변환이라고 할 수 있는가? 예 또는 아니오로 답하고 그 이유를 기술하라.
- (i) 정상적인 기하 변환을 할 경우 주어진 물체의 꼭지점에 대해 실제로 모델링 변환이 적용되는 box의 기호를 기술하라.

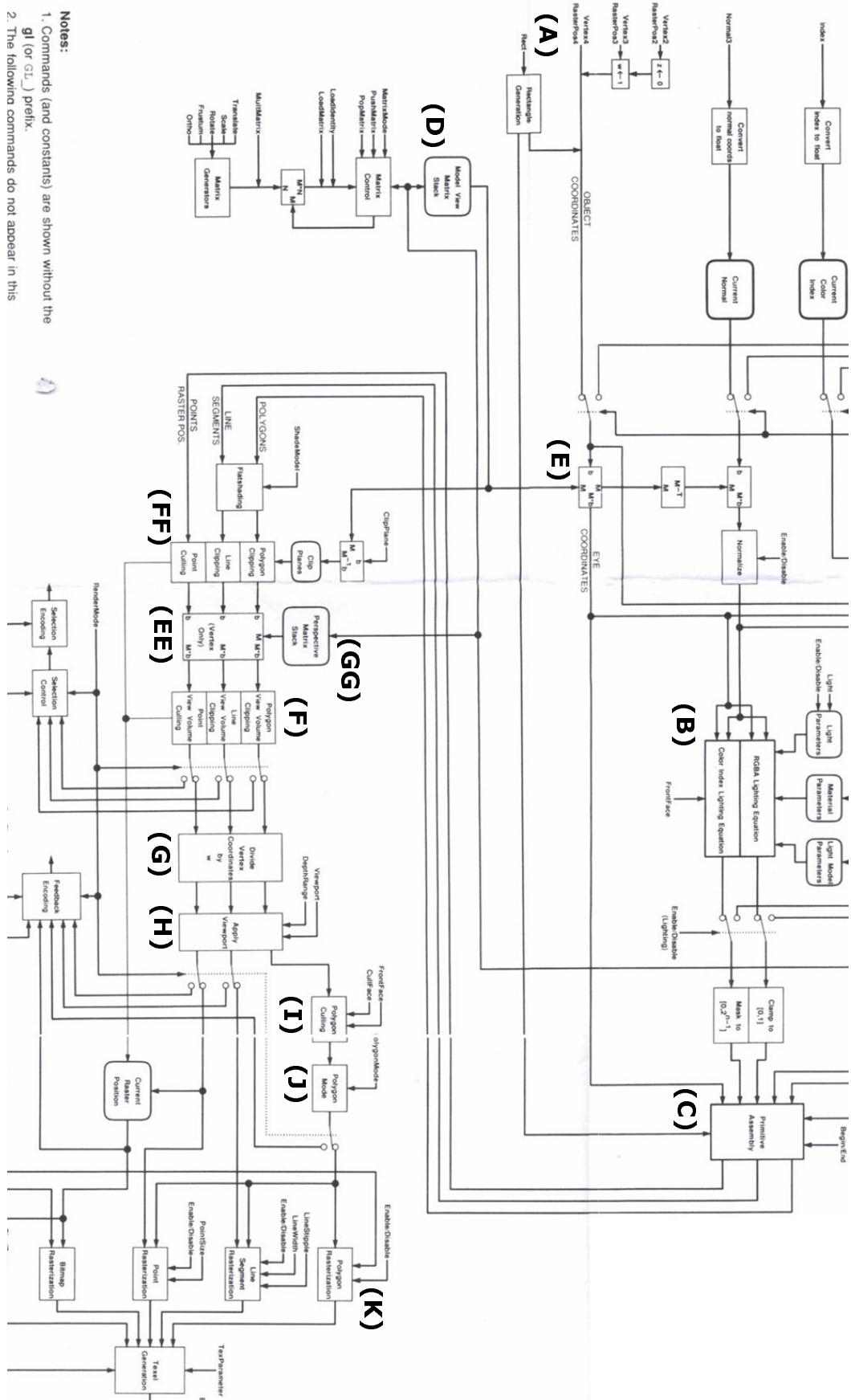


그림 7: OpenGL fixed-function 렌더링 파이프라인

[CSE4170: 기초 컴퓨터 그래픽스]

중간고사 문제

(담당교수: 임인성)

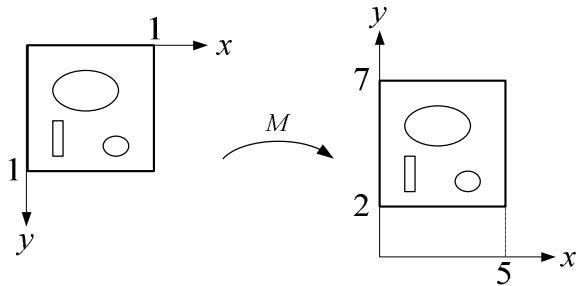


그림 1: 2차원 윈도우 매핑 변환

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.

- 그림 1에서와 같이 윈쪽의 윈도우의 내용을 오른쪽 윈도우 안으로 매핑을 해주는 2차원 아핀 변환에 대한 3행 3열 행렬 M 을 기본 아핀 변환의 합성을 통하여 구하라 (M 을 기본 변환 행렬의 곱으로 표현한 후, 최종 결과 행렬을 반드시 기술할 것).
- 다음은 OpenGL에서 사용하는 기하 파이프라인에 대한 단답식 문제이다. CC, EC, MC, NDC, WC, WdC 등의 좌표계 이름과 그림 2에 표시된 기호를 사용하거나, 또는 “어떤 어떤 box의 윈쪽”과 같은 방식으로 적절히 그리고 정확히 답하라.
 - 그림 2의 (A)로 표시된 지점에 해당하는 OpenGL 좌표계의 이름은?

- 그림 2의 (B)로 표시된 지점에 해당하는 OpenGL 좌표계의 이름은?
- 그림 2의 (C)로 표시된 지점에 해당하는 OpenGL 좌표계의 이름은?
- 기하 변환 과정 중 다각형의 꼭지점이 나열된 순서 정보를 통하여 안 보이는 뒷면에 해당하는 다각형을 제거할 수 있는데, OpenGL에서는 정확히 어떤 지점에서 이 계산이 수행되는가?
- 정상적인 기하 변환을 할 경우 전체 파이프라인 과정에서 항상 강체 변환이 사용되는 부분은 어느 좌표계에서 어느 좌표계로의 변환에 해당하는가?
- OpenGL 렌더링 파이프라인에서 “정규화된 필름”에 해당하는 좌표계가 존재하는 위치를 그림 2에서 찾는다면 정확히 어디인가?
- 정상적으로 렌더링 할 경우, 카메라를 배치하기 위하여 적절한 OpenGL 함수를 호출할 경우, 가장 직접적으로 영향을 미치는 box는 어떤 것일까?
- 정상적으로 렌더링을 할 경우, `glOrtho()` 함수가 가장 직접적으로 영향을 미치는 위치는?
- 촬영한 사진을 현상 한 후, 인화지에 확대하는 과정과 가장 밀접한 관련이 있는 곳은?

3. 다음에 주어진 간단한 3D 뷰잉 관련 OpenGL 코드를 보고 답하라.

```

void display(void) {
    glClearColor(0, 0, 0, 1);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    glPolygonMode(GL_FRONT_AND_BACK,
                  GL_FILL);

// (LINE A)
 glBegin(GL_TRIANGLES);
    glVertex2f(-2.0, -2.0);
    glVertex2f(0.0, -2.0);
    glVertex2f(0.0, 0.0);
    glVertex2f(-2.0, 0.0); glEnd();
    glFlush();
}

void reshape(int W, int H) {
    glViewport(0, 0, W, H);
}

void RegisterCallback(void){
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
}

void main (int argc,
           char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Sogang CSE");
    RegisterCallback();
    glutMainLoop();
}

```

- (a) 위 코드를 수행할 경우 500×500 크기의 윈도우에 어떤 그림이 그려질지, 가급적 정확한 척도 (화면에 눈금을 표시하거나 하는 등의)를 사용하여 윈도우의 내용을 도시하라.

(b) 만약 이 코드의 (LINE A) 부분에 아래의 두 문장을 삽입한 후, 프로그램을 수행 시키면 윈도우에 어떤 그림이 그려질까? 위 문제와 마찬가지로 가급적 정확한 척도를 사용하여 그 내용을 도시하라.

```

glMatrixMode(GL_PROJECTION);
glOrtho(-2, 2, -2, 2, -1, 1);

```

(c) 바로 위 문제에서와 같이 코드의 내용을 수행할 경우, 어떤 심각한 문제가 발생하는데 그것이 무엇인지 기술하라.

(d) 위 두 문장이 삽입된 상태에서 그 문제를 해결하려면 코드를 어떻게 수정해야 할까?

(e) 이번에는 원래의 코드의 (LINE A) 부분에 아래의 네 문장을 삽입한 후, 프로그램을 수행시키면 윈도우에 어떤 그림이 그려질까? 위 문제와 마찬가지로 가급적 정확한 척도를 사용하여 그 내용을 도시하라.

```

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(1, 1, 0);
glScalef(0.5, 0.5, 0.5);

```

4. 다음은 OpenGL의 뷰잉 변환에 관한 문제이다. 아래에 주어진 어떤 display callback 함수의 내용을 보고 답하라. 여기서 glutWireTeapot(2.0); 문장은 원점을 중심으로 적절한 크기의 주전자를 그려주는 역할을 한다.

```

void display(void) {
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(34.0, 1.0, 0.5, 1000.0);
}

```

```

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0.0, 0.0, 10.0,
           10.0, 0.0, 10.0, 0.0, 1.0, 0.0);

glTranslatef(10.0, 0.0, 10.0);
glutWireTeapot(2.0);
glFlush();
}

```

(a) 위 코드를 수행할 때 적용되는 뷰잉 변환 행렬 M_V 의 내용을 기술하라.

(b) 원래의 코드에서 `gluLookAt(*)`; 문장을 아래와 같은 두 문장으로 대치하려하는데, 같은 내용의 그림이 그려지도록 하기 위해서는 각 문장의 인자가 무엇이 되어야 하는지 정확히 기술하라.

```

glRotatef(*, *, *);
glTranslatef(*, *, *);

```

(c) 뷰잉 변환은 세상 좌표계에서 카메라의 프레임을 위치시키는 것에 대응된다. 만약 바로 위의 문제에서와 같은 상태에서 카메라 프레임을 하나의 물체라 가정한 후, 세상 좌표계에서 이 물체를 x_w 축 방향으로 3.0만큼 이동을 시켰다면, 이때의 뷰잉 변환은 `glTranslatef(3.0, 0.0, 0.0);` 또는 `glTranslatef(-3.0, 0.0, 0.0);` 문장 중 하나를 바로 위 문제의 두 문장의 적절한 위치에 삽입하면 된다. 과연 어떻게 하면 되는가? (이 문제는 위 문제의 정답을 기술한 사람만 풀 것).

(d) 원래의 코드에서 `gluLookAt(*)`; 문장을 아래와 같은 두 문장으로 대치하려는데, 같은 내용의 그림이 그려지도록 하기 위해서는 각 문장의 인자가 무엇이 되어야 할지 정확히 기술하라.

```

glTranslatef(*, *, *);
glRotatef(*, *, *);

```

(e) 원래의 코드를 수행시키면 그림 3과 같은 그림이 그려진다. 이 코드의 `gluLookAt(*);` 문장의 마지막 세 인자 $0.0, 1.0, 0.0$ 을 $0.0, 0.0, 1.0$ 로 대치하면 어떻게 그림이 그려질까?

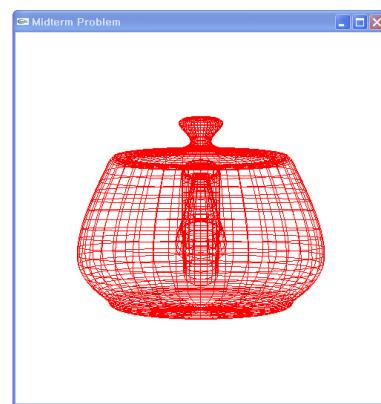


그림 3: 주전자 도시

5. 그림 4에는 모델링 좌표계 (MC)에 설계되어 있는 자동차 모델을 세상 좌표계 (WC)로 배치해주는 과정이 도시되어 있다 (이 그림에서 z_m 과 z_w 축은 각각 오른손 좌표계 방향으로 되어 있음).

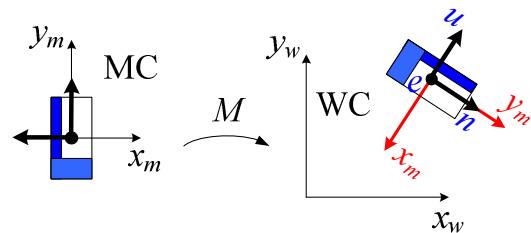


그림 4: 모델링 변환과 뷰잉 변환

(a) 자동차의 위치와 방향에 관련된 정보가 세상 좌표계에서 한 점 $e = (e_x, e_y, e_z)$ 과 길이가 1이고 서로 수직인 세 벡터 $u = (u_x, u_y, u_z)$, $v = (0, 0, 1)$, $n = (n_x, n_y, n_z)$ 로 주어져 있다. 이때 적용되는 모델링 변환에 해당하는 4행 4열 행렬 M_M 을 이동 변환, 크기 변환, 회전 변환 등 의 기본 변환 행렬의 곱으로 적절히 표현

- 하라 (예를 들어, $M_M = T_1 \cdot R_1 \cdot T_2 \cdot S_1$ 와 같은 방식으로 표현하고, 각 기본 변환 행렬의 내용을 정확히 기술하되, 전체 곱 행렬은 계산할 필요가 없음).
- (b) $C = (e, (u, v, n))$ 정보를 사용하여 카메라를 배치하려 할 때 적용되는 뷰잉 변환 M_V 를 위의 문제에서 사용한 기본 변환의 행렬 또는 그의 역행렬을 가급적 많이 사용하여 표현하라.
6. 그림 5는 다음 문제와 관련한 투영 변환에 대한 그림이다.
- (a) OpenGL에서 `gluPerspective(60.0, 1.0, 5.0, 10.0);`과 같은 문장을 수행 할 경우 계산되는 투영 변환 행렬 M_P 는 OpenGL에서 규정한 방식대로 눈 좌표계 (EC)의 점 (x_e, y_e, z_e) 를 정규디바이스 좌표계 (NDC) 점 (x_{nd}, y_{nd}, z_{nd}) 로 변환을 해준다. z_{nd} 의 경우 $z_{nd} = \alpha + \frac{\beta}{z_e}$ 와 같은 형태의 변환을 사용하여 유도할 수 있는데, 이때 α 와 β 값이 무엇인지 상세히 유도하라 (위 함수의 인자 값을 사용하여 문제를 풀 것).
- (b) y_{nd} 를 x_e, y_e, z_e 로 표현하라 ($\tan 45 = 1, \tan 60 = \sqrt{3} = \frac{1}{\tan 30}$).
- (c) M_P 행렬의 두 번째 행과 네 번째 행의 내용을 정확히 기술하라
7. 시험지 뒤에 첨부한 프로그램은 적절한 모델링 변환을 통하여 자동차를 그려주는 OpenGL 프로그램이다. 이 프로그램을 보면서 답하라.
- (a) 이 프로그램은 그림 6에 주어진 자동차에 대한 트리 구조를 어떤 방식으로 탐색을 하고 있는가? 자료 구조 시간에 배운 용어를 사용할 것.
- (b) 이 프로그램에서 원근 나눗셈(perspective division)과 가장 관련이 있는 문장의 번호는?
- (c) 이 프로그램에서는 사용자가 어떠한 방식으로 자동차를 앞으로 움직이게 할수 있을까? 정확하게 기술할 것.
- (d) 세상 좌표계 (WC)를 기준으로 할 때 카메라가 세상을 바라보는 방향을 벡터로 표현하라.
- (e) 눈 좌표계 (EC)를 기준으로 할 때 카메라가 세상을 바라보는 방향을 벡터로 표현하라.
- (f) 59번의 이동 변환 관련 문장이 수행되기 직전의 모델뷰 행렬 스택의 내용을 정확하게 그려라. $M_V M_P, M_{VP}$ 와 그림 6의 행렬 기호등을 적절히 사용하라. 현재 `draw_wheel_and_nut(angle)` 함수는 100번 문장에서 호출한 상태임.
- (g) 52번 문장의 `draw_wheel_and_nut(angle)` 함수에서 이상한 부분을 지적하고 수정하라. 이유를 설명할 것.
- (h) 83번 문장의 `void cross_prod_vec3(float *u, float *v, float *n)` 함수는 3차원 벡터에 대한 외적, 즉 $n = u \times v$ 을 계산하기 위한 함수이다. 내용을 메꾸어라.
- (i) 이 프로그램에서 사용하는 인자들을 사용하여 정규 디바이스 좌표계 (NDC)의 좌표 (x_{nd}, y_{nd}) 를 윈도우 좌표계 (WdC)의 좌표 (x_{wd}, y_{wd}) 로 어떻게 변환 시켜주는지 뷰폿 변환(x 와 y 좌표에 대해서만)을 유도하라.
- (j) 이 프로그램에서 뷰 매핑과 가장 관련이 많은 문장의 번호를 기술하라.
- (k) 이 프로그램에서 자동차를 세상 좌표계로 배치해주는 모델링 변환이 강체 변환인지 아닌지 답하고 그 이유를 간략히 설명하라.

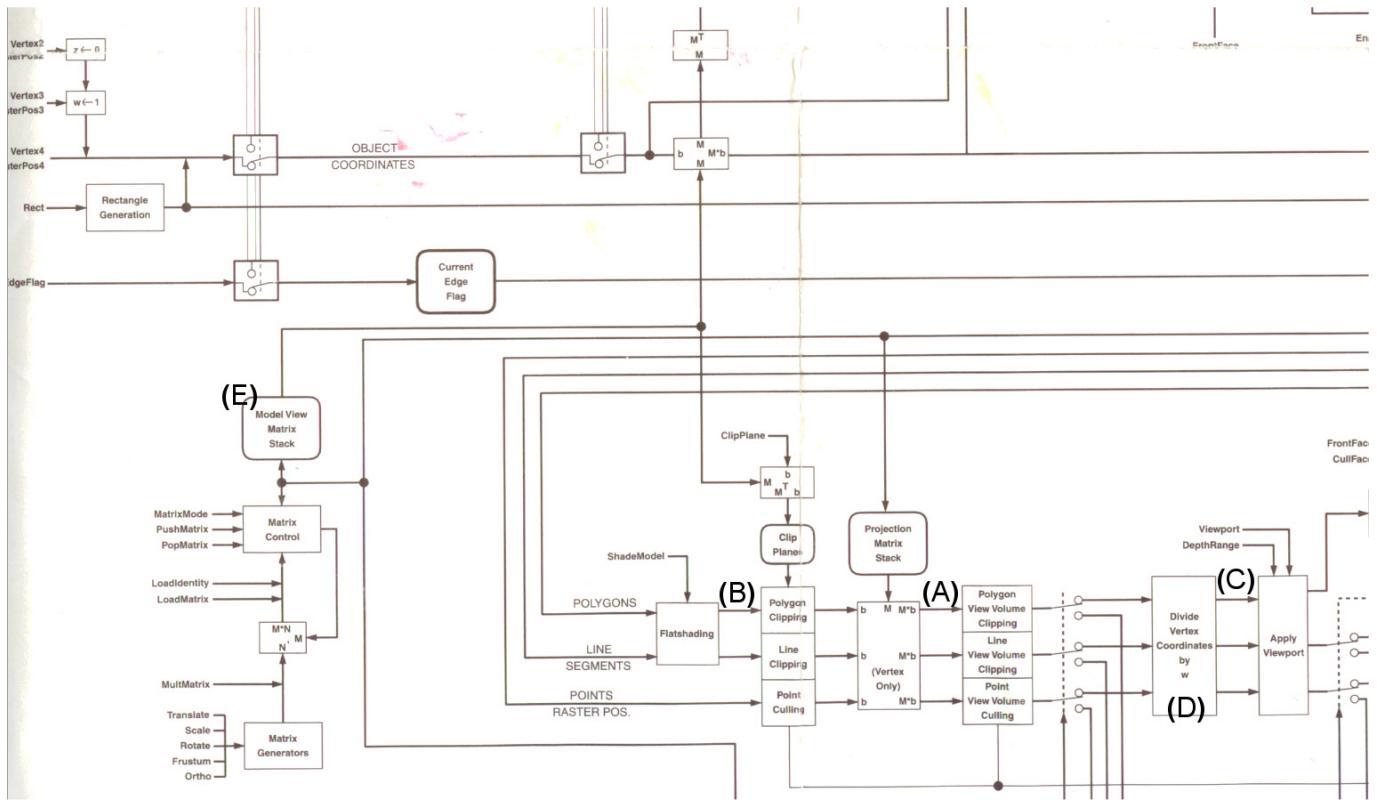


그림 2: OpenGL 파이프라인

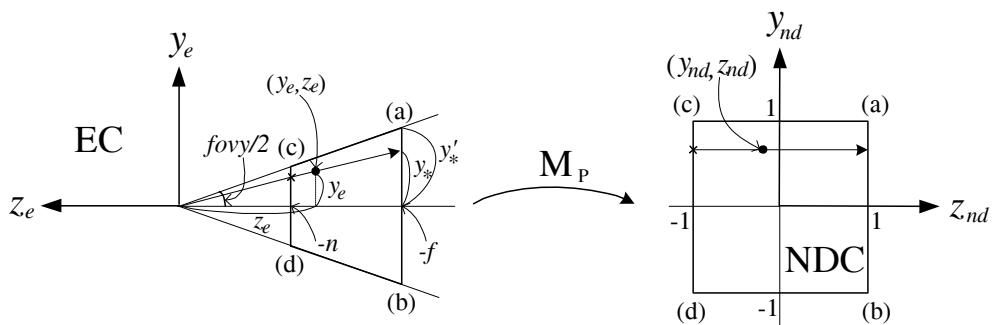


그림 5: 투영 변환

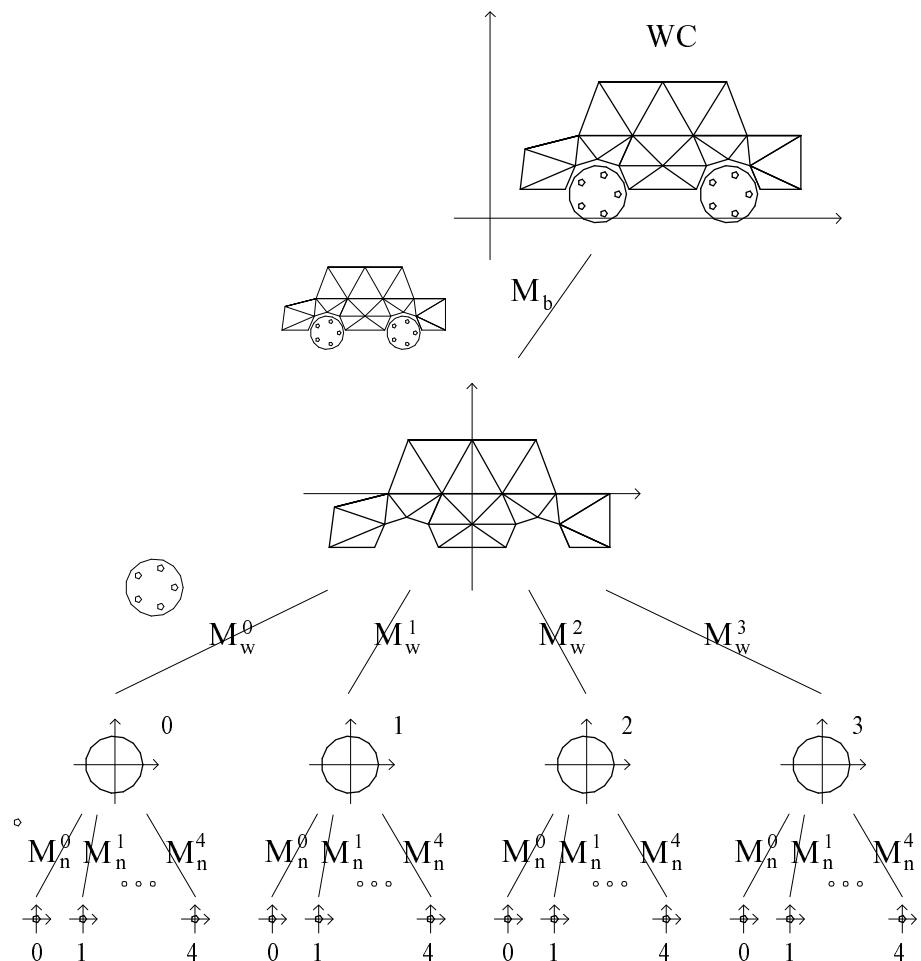


그림 6: 자동차의 계층적 표현

```

1 #include <stdio.h>
2 #include <math.h>
3 #include <GL/glut.h>
4
5 #define MAX_POLY 200
6 #define MAX_VERT 20
7 #define MAX_PATH 1000
8
9 #define DRAW_CAR_DUMMY 2001
10 #define DRAW_CAR_CORRECT 2002
11
12 typedef struct {
13     int invertex;
14     float poly[MAX_VERT][3];
15 } mypolygon;
16
17 mypolygon body[MAX_POLY], wheel[MAX_POLY], nut[MAX_POLY];
18 int npolyb, npolyw, npolyn;
19
20 float path[MAX_PATH][3];
21 int npath, path_exist, drawing_state = DRAW_CAR_CORRECT;
22
23 double dist;
24
25 int prev_i, cur_i = 0;
26 int rightbutbpressed = 0;
27
28 void read_object(char *file, mypolygon *object, int *npoly);
29 void read_path(char *file);
30 void read_objects(void);
31 void draw_axes(void);
32 void draw_path(void);
33 void draw_ground(void);
34
35 void draw_body(void) {
36     // Draw the body.
37     ...
38 }
39
40 void draw_wheel(float angle) {
41     // Draw the wheel.
42     ...
43 }
44
45 void draw_nut(void) {
46     // Draw the nut.
47     ...
48 }
49
50 #define rad 1.7
51 #define vw 1.0
52 void draw_wheel_and_nut(float angle) {
53     int i;
54     draw_wheel(angle); // draw wheel object
55     for (i = 0; i < 5; i++) {
56         // nut i
57         glPushMatrix();
58         glTranslate(rad-0.5, 0, vw); // rad = 1.7, vw = 1.0
59         glRotatef(72.0*i, 0.0, 0.0, 1.0);
60         draw_nut(); // draw nut object
61         glPopMatrix();
62     }
63 }
64
65 #define TO_DEG 57.29579
66 void normalize_vec3(float *v) {
67
68     ...
69 }
70
71 float dot_prod_vec3(float *u, float *v) {
72 }
73
74 float compute_length_mul_two_vec3(float *u, float *v) {
75
76     ...
77 }
78
79 float angle_between_two_vec3(float *u, float *v) {
80     ...
81 }
82
83 void cross_prod_vec3(float *u, float *v, float *n) {
84     // ???
85 }
86
87 float wheel_rot_angle_in_y(void) {
88     ...
89 }
90
91 float wheel_rot_angle_in_z(void) {
92     ...
93 }
94
95 void draw_car_dummy(void) {
96     draw_body(); // draw body object
97     glPushMatrix();
98     glTranslate(-3.9, -3.5, 4.5);
99     draw_wheel_and_nut(0.0); // wheel 0
100    glPopMatrix();
101
102    glPushMatrix();
103    glTranslate(-3.9, -3.5, 4.5);
104    draw_wheel_and_nut(0.0); // wheel 1
105    glPopMatrix();
106
107    glPushMatrix();
108    glTranslate(-3.9, -3.5, -4.5);
109    glScalef(1.0, 1.0, -1.0);
110    draw_wheel_and_nut(0.0); // wheel 2
111
112    glPopMatrix();
113
114    glPushMatrix();
115    glTranslate(3.9, -3.5, -4.5);
116    glScalef(1.0, 1.0, -1.0);
117    draw_wheel_and_nut(0.0); // wheel 3
118
119 }
120
121 void set_up_rot_mat(float *m, int i) {
122
123 }
124
125 void draw_fence(GLfloat r, GLfloat g, GLfloat b) {
126     ...
127 }

```

```

127 }
128 void draw_fences(void) {
129     ...
130 }
131 }
132 void draw_world(void) {
133     GLfloat m[16];
134     glLoadMatrix(m);
135     glMatrixMode(GL_MODELVIEW); // Modeling Transformation
136     glLoadIdentity();
137     gluLookAt(-15.0, 20.0, 40.0, path[cur_i][0], 4.89, path[cur_i][2], 0.0, 1.0, 0.0);
138 }
139 void draw_ground() {
140     draw_fences();
141     draw_axes();
142     if (path_exist) draw_path();
143     if (path_exist) draw_path();
144     set_up_rot_mat(m, cur_i);
145     glPushMatrix();
146     glTranslatef(path[cur_i][0], 4.89, path[cur_i][2]);
147     glPushMatrix();
148     gITranslatef(path[cur_i][0], 4.89, path[cur_i][2]);
149     gMultMatrixf(m);
150     draw_car_dummy();
151     glPopMatrix();
152     glPopMatrix();
153 }
154 void render(void) {
155     glClear(GL_COLOR_BUFFER_BIT);
156     drawWorld();
157     glutSwapBuffers();
158 }
159 }
160 void keyboard (unsigned char key, int x, int y) {
161     ...
162 }
163 }
164 int prevx_mouse;
165 void mousepress(int button, int state, int x, int y) {
166     if ((button == GLUT_RIGHT_BUTTON) && (state == GLUT_DOWN)) {
167         if (prevx.mouse == x) {
168             rightbuttonpressed = 1;
169         }
170         else if ((button == GLUT_RIGHT_BUTTON) && (state == GLUT_UP))
171             rightbuttonpressed = 0;
172     }
173 }
174 void mousemove(int x, int y) {
175     double deltax;
176     if (rightbuttonpressed) {
177         if (deltax = x - prevx.mouse;
178             prevx.mouse = x;
179             if ((cur_i + deltax > 0) && (cur_i + deltax < npath-1)) {
180                 prev_i = cur_i; cur_i += deltax;
181                 dist = sqrt((path[cur_i][0]-path[prev_i][0])*(path[cur_i][0]-path[prev_i][1])
182                             +(path[cur_i][1]-path[prev_i][1])*(path[cur_i][1]-path[prev_i][2])
183                             +(path[cur_i][2]-path[prev_i][2])*(path[cur_i][2]-path[prev_i][2]));
184             }
185             if (deltax < 0.0) dist *= -1.0;
186         }
187     }
188 }
189 }
190 }
191 void reshape(int width, int height) {
192     glViewport(0, 0, width, height);
193 }
194 glMatrixMode(GL_PROJECTION);
195 gluLoadIdentity();
196 gluPerspective(30.0, width/(double)height, 1.0, 150.0);
197 }
198 }
199 void init_OpenGL(void) {
200     ...
201 }
202 }
203 void init_windows(void) {
204     glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
205     glutInitWindowSize(1280, 1024);
206     glutCreateWindow("Car in Hierarchy 2");
207     glutDisplayFunc(render);
208     glutKeyboardFunc(keyboard);
209     glutMouseFunc(mousepress);
210     glutMotionFunc(mousemove);
211     glutReshapeFunc(reshape);
212 }
213 }
214 void main(int argc, char **argv) {
215     read_objects();
216     glutInit(&argc, argv);
217     init_windows();
218     init_OpenGL();
219     glutMainLoop();
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
599 }
```

[CSE4170: 기초 컴퓨터 그래픽스]

중간고사 문제

(담당교수: 임인성)

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.

- 다음 단답식 문제에 답하라. 필요한 경우 OC, CC, EC, MC, NDC, WC, WdC 등의 OpenGL 좌표계 이름을 참고하여 답하라.
 - 좌표가 $(3.0, 6.0, -9.0, -2.0)$ 인 3차원 투영 공간의 점에 해당하는 아핀 공간의 점의 좌표 (x, y, z) 는?
 - 3차원 아핀 변환을 나타내는 4행 4열 행렬 M 이 변환 후에도 물체의 크기와 모양을 보존해주기 위한 조건을 정확히 기술하라.
 - 주어진 회전 변환 R 과 크기 변환 S 간에 교환 법칙이 성립하기 위한 조건은 무엇인가?
 - 원근 투영에 대한 계산을 할 경우 바로 이 계산이 이뤄질 때 원근감이 생기게 되는데, 이 계산의 이름을 무엇이라 할까?
 - OpenGL의 뷔잉 파이프라인에서 `glVertex3f()` 함수를 사용하여 설정한 꼭지점에 곱해지는 두 번째 행렬 스택의 탑에 있는 변환 행렬은 일반적으로 어떤 좌표계에서 어떤 좌표계로의 변환을 위하여 사용하는가?
 - OpenGL 렌더링 파이프라인에서 일단 NDC로 옮겨 온 후 어떤 형태의 투영 변환만 일어나는지 정확히 기술하라.

- 그림 1에서와 같이 왼쪽의 윈도우의 내용을 오른쪽 윈도우 안으로 매핑을 해주는 2차원 아핀 변환에 대한 3행 3열 행렬 M 을 기본 아핀 변환의 합성을 통하여 구하라 (여기서 왼쪽의 윈도우는 각 변의 길이가 2이고 중심이 원점이 사각형인데, 합성 과정을 반드시 기술한 후, 최종 결과 행렬을 기술할 것).

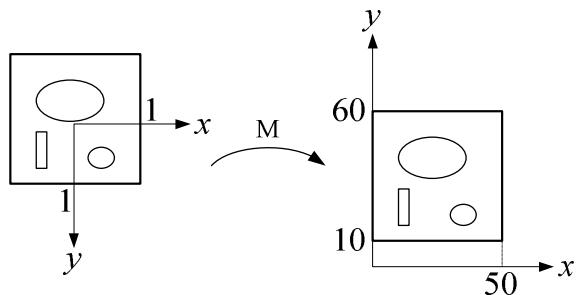


그림 1: 2차원 윈도우 매핑 변환

- 그림 2는 OpenGL의 직교 투영 함수인 `glOrtho(l, r, b, t, n, f)` 함수에 관한 내용을 나타내고 있는데, 이 함수를 호출할 경우, 그림에 도시된 바와 같은 뷔잉 볼륨이 $[-1, 1] \times [-1, 1] \times [-1, 1]$ 구간의 정육면체로 매핑되는 투영 변환 행렬 M_{ortho} 가 계산된다 (OpenGL에서는 이 과정에서 투영 방향이 반대로 바뀌게 되는 것을 상기할 것). 이때 α, β, γ 에 해당하는 값을 기술하라 (그러한 값을 얻게 된 이유를 상세히 기술할 것).

$$M_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & \alpha \\ 0 & \frac{2}{t-b} & 0 & \beta \\ 0 & 0 & \frac{-2}{f-n} & \gamma \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

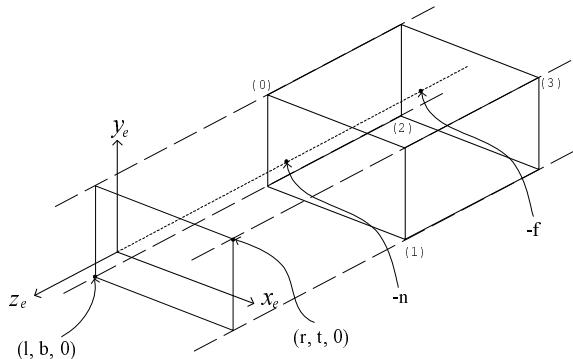


그림 2: `glOrtho()` 함수에 대한 뷔잉 볼륨

4. 다음은 주어진 법선 벡터 (*normal vector*) n 에 대하여 4행 4열 행렬 M 에 해당하는 아핀 변환을 가하는 것에 관한 문제이다. 그림 3을 보고 답하라.

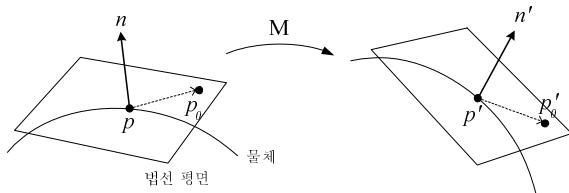


그림 3: 법선 벡터의 변환

- (a) 아래 글을 읽고 계속하여 $n' = (M^{-1})^t \cdot n$ 임을 수식을 통하여 정확히 증명하라.

그림 3에서와 같이 주어진 점 $p = (x \ y \ z \ 1)^t$ 에서의 법선 벡터가 $n = (n_x \ n_y \ n_z \ 0)^t$ 이라고 하자. 이때 점 p 에서의 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 n 이 변환 행렬 M 에 의해 각각 $p' = (x' \ y' \ z' \ 1)^t$, $p'_0 = (x'_0 \ y'_0 \ z'_0 \ 1)^t$, $n' = (n'_x \ n'_y \ n'_z \ 0)^t$ 으로 변환이 된다고 하면, $p' = M \cdot p$ 와 $p'_0 = M \cdot p_0$ 로부터 p 와 p_0 를 구해 위의 관계식에 대입하여 다음과 같은 식을 얻게 된다.

- (b) 법선 벡터 n 에 대하여 회전 변환을 가하는 문제를 생각해보자. 이때, M 을 아래와 같이 정의한다면,

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

이때 v_1, v_2, v_3 의 값은 어떤 값을 가질까?

- (c) 이 행렬의 왼쪽-위쪽의 3행 3열 부행렬의 세 개의 열벡터를 각각 a_1, a_2, a_3 라 할 때, 이 세 개의 벡터가 만족하는 수학적인 성질을 정확히 기술하라.
 (d) 문제 (b)의 조건하에, 회전 행렬 M 에 대하여 행렬 $(M^{-1})^t$ 의 내용을 정확히 기술하라. 문제 (b)의 답의 내용을 반영할 것.

5. 다음은 투영 변환에 관련된 문제이다.

- (a) 그림 4는 원근 투영을 위한 OpenGL 함수 `gluPerspective(fovy, aspect, n, f)`를 호출할 경우 좌표가 (x_e, y_e, z_e) 인 EC의 점이 NDC의 (x_{nd}, y_{nd}, z_{nd}) 로 변환되는 과정을 보여주고 있다. z_{nd} 의 경우 $z_{nd} = \alpha + \frac{\beta}{z_e}$ 와 같은 형태의 변환을 사용하여 유도할 수 있는데, 이때 α 와 β 값이 무엇인지 기술하라 (이 두 값을 구하는 과정을 기술할 것).

- (b) x 와 y 에 대해서는 $x_{nd} = \frac{\cot(\frac{fovy}{2}) \cdot x_e}{\frac{asp}{-z_e}}$ 와 $y_{nd} = \frac{\cot(\frac{fovy}{2}) \cdot y_e}{-z_e}$ 와 같은 관계를 유도할 수 있는데, 이때 위의 함수 호출 시 계산되는 원근 투영 변환 행렬 M_{pers} 를 정확히 기술하라.

6. 다음은 수업 시간에 설명한 움직이는 자동차에 관한 문제이다. 그림 5(a)에는 자동차 몸체를 자신의 모델링 좌표계에서 도시한 내용이 주어져 있다 (가는 선분의 좌표축의 빨강색, 녹색, 파랑색 선분은 각각 모델링 좌표계에서의 x, y, z 축을 나타냄). 이 그림에는 몸체 외에 자동차 운전석에서 세상을 바라보는 모습을 렌더링하기 위하여 카메라 프레임을 자동차 몸체의 모델링 좌표계와 일치된 상태에서 y 축 둘레로 90도 회전한 후, x, y, z 축 방향으로 각각 -3.0, 0.5, 그리고 2.5만큼 이동시켜 배치한 모습도 도시하고 있는데, 굵은 선분의 좌표축이 눈좌표계의 좌표축을 의미하게 된다 (색깔의 의미는 모델링 좌표계에서의 의미와 같다). 다음 그림 5(b)는 자동차 몸체에 대해 배열 m 이 저장하고 있는 회전 변환을 가한 다음 x, y, z 축 방향으로 각각 $x0, y0, z0$ 만큼 이동시켜 세상에 배치한 모습을 도시하고 있다. 아래에는 이 상황에서 그림 5(c)에 도시된 바와 같이 자동차의 카메라 프레임에서 세상을 바라본 모습을 렌더링하기 위한 뷰잉 변환을 OpenGL API를 사용하여 프로그래밍한 예가 주어져 있다. 이때 (A)와 (B)에 들어갈 내용을 C 언어와 OpenGL API 문법에 맞게 기술하라 ($minv$ 는 m 행렬의 역행렬을 저장하고 있는 배열임).

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
(A)
glMultMatrixf(minv);
(B)
```

7. 다음은 3차원 뷰잉에 관한 문제이다. 아래 프로그램과 그림 6을 보고 답하라.

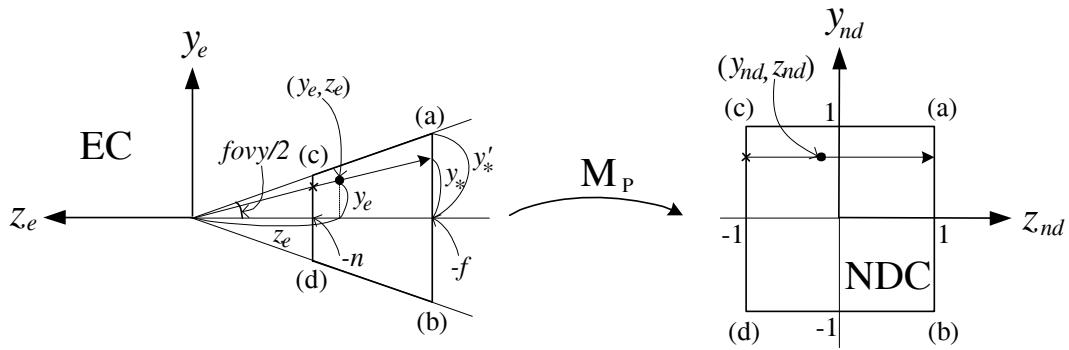


그림 4: 투영 변환의 유도

- (a) 서로 수직이고 길이가 1인 세 벡터 $\mathbf{u} = (u_x \ u_y \ u_z)^t$, $\mathbf{v} = (v_x \ v_y \ v_z)^t$, $\mathbf{n} = (n_x \ n_y \ n_z)^t$ 가 주어진 점 $p = (e_x \ e_y \ e_z)^t$ 을 원점으로 하여 세상 좌표계에 떠 있다. 이 점이 눈좌표계의 원점, 그리고 세 벡터의 방향이 눈 좌표계에서 각각 x , y , z 축의 방향이 되도록 해주는 뷰잉 변환에 대한 4행 4열 행렬 M_V 를 회전 변환 행렬 R 과 이동 변환 행렬 T 의 곱 $R \cdot T$ 의 형태로 표현하라.
- (b) 그림 (a), (b), 그리고 (c)는 각각 자신의 모델링 좌표계에서의 LEGS, ARM, 그리고 CAR 물체를 도시하고 있다. 그림 (d)는 $c_angle = -45.0$ 과 $r_angle = 0.0$ 인 상태에서 렌더링 한 모습을 보여주고 있다 (각 축의 색깔의 의미는 위의 문제에서와 동일하고, 16개의 ARM이 있음을 알 수가 있음). 이러한 그림이 그려지도록 하려할 때, 아래 프로그램의 (A) 부분에 들어갈 내용을 C 언어 문법에 맞게 기술하라.
- (c) 이 프로그램에서 r_angle 변수의 역할은 무엇인가?
- (d) 이 문제의 렌더링 계산 시 사용되는 뷰잉 변환 행렬 M_V 를 두 개의 변환 행렬의 곱으로 표현하라.
- (e) 아래 프로그램에서 동일한 그림이 그려지도록 뷰잉 변환에 해당하는 두 문장을 $gluLookAt()$ 함수를 사용하여 대치하라.
- (f) 그림 (e)는 그림 (d)의 렌더링 상황에서 놀이 기구의 일부 내용을 확대한 모습을 도시하고 있다. 만약 아래 프로그램의 (B)와 (C) 사이의 두 문장을 제거한다면, 렌더링 결과가 어떻게 변할지 그림 (e)의 내용에 대해 그 결과를 가급적 정확히 스케치하라.

```

void draw_world() {
    int i;

    draw_axes();
    glPushMatrix();
    draw_floor();
    draw_LEGS();
    // r_angle = 0.0
    glRotatef(r_angle, 0, 0, 1);
    for(i = 0; i < 16; i++) {
        glPushMatrix();
        glRotatef((A), 0.0, 0.0, 1.0);
        draw_ARM();
        glPushMatrix();
        glTranslatef(0.0, 6.0, 0.0);
        // (B)
        glRotatef(r_angle, 0.0, 0.0, 1.0);
        glRotatef(-22.5*i, 0.0, 0.0, 1.0);
        // (C)
        glScalef(0.12, 0.12, 0.12);
        draw_CAR();
        glPopMatrix();
        glPopMatrix();
    }
    glPopMatrix();
}

void render3(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    // c_angle = -45.0
    glRotatef(c_angle, 0.0, 1.0, 0.0);
    glTranslatef(-25.0, 1.0, -25.0);
    draw_world();
    glutSwapBuffers();
}

```

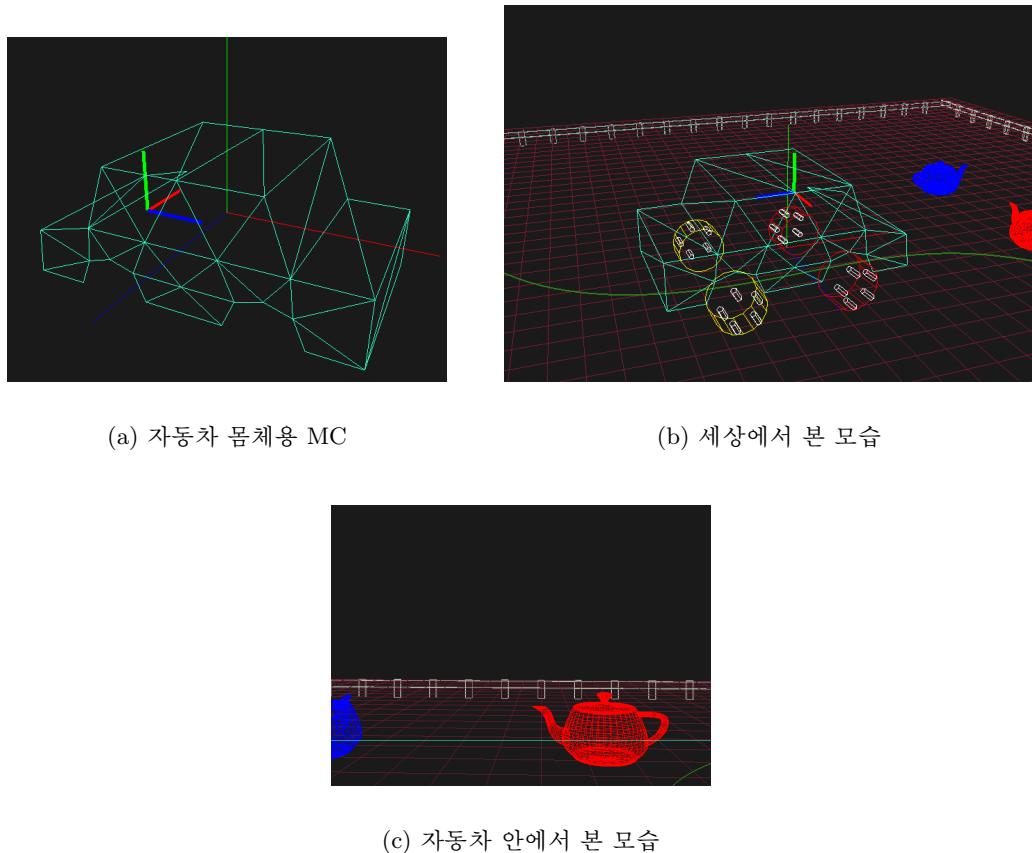


그림 5: 올바른 카메라 배치를 위한 뷰잉 변환

8. 다음은 OpenGL 시스템에서 사용하는 렌더링 파이프라인에 관한 단답식 문제이다. 그림 7을 보고, OC, CC, EC, MC, NDC, WC, WdC 등 의 좌표계 이름을 참고하여 적절히 답하라.
 - (a) 원근 투영을 사용하여 렌더링을 할 경우, OpenGL 렌더링 파이프라인의 어느 부분에서 원근감이 생성이 되는지 가장 연관성이 높은 box의 기호를 기술하라.
 - (b) 정상적으로 렌더링 할 경우, 카메라를 배치하기 위하여 적절한 OpenGL 함수를 호출할 경우, 가장 직접적으로 영향을 미치는 box의 기호를 기술하라.
 - (c) OpenGL 상수 GL_CCW와 가장 연관성이 높은 box의 기호를 기술하라.
 - (d) (G) box와 (H) box 사이의 지점에 해당하는 OpenGL 좌표계의 이름은?
 - (e) OpenGL 렌더링 파이프라인에서 NDC까지 변환되어 온 점 p 의 y_{nd} 좌표가 가질 수 있는 좌표값의 범위는?

- (f) OpenGL 렌더링 파이프라인을 따라 흘러가는 그래픽스 데이터는 어느 특정 지점에서 그 형태가 완전히 변하게 되는데, 이 지점과 가장 연관성이 높은 box의 기호를 기술하라.
- (g) 정상적으로 렌더링을 할 경우, glOrtho() 함수를 호출할 경우, 가장 직접적으로 영향을 미치는 box의 기호를 기술하라.
- (h) 정상적인 렌더링 상황에서 좌표점을 투영 좌표계 (x, y, z, w) 로 표현할 경우 w 가 1이 아닌 값이 나타날 수 있는 box의 기호를 기술하라.

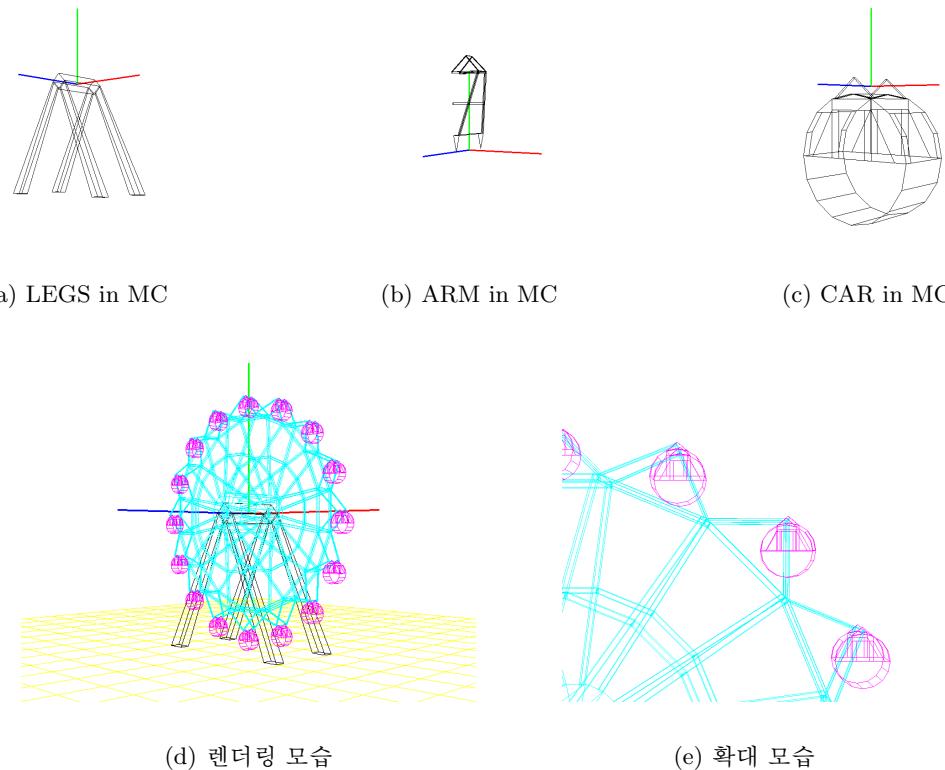


그림 6: 놀이 기구 렌더링 관련 그림

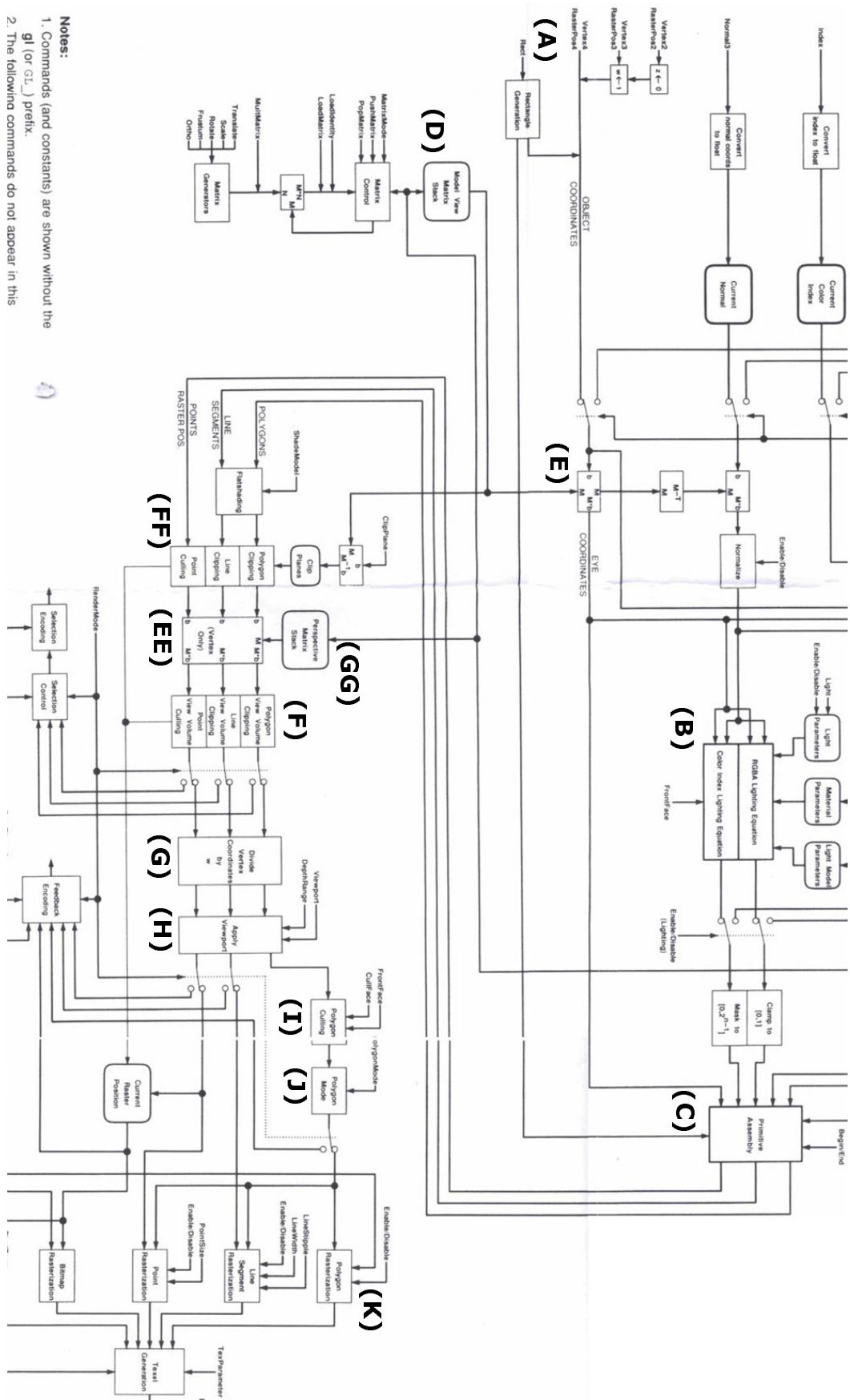


그림 7: OpenGL 렌더링 파이프라인

[43-170: 기초 컴퓨터 그래픽스] – 중간고사

(담당교수: 임인성)

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.

- 다음은 주어진 법선 벡터 (*normal vector*) n 에 대하여 4행 4열 행렬 M 이 기술해주는 아핀 변환을 가하는 것에 관한 문제이다. 그림 1을 보고 답하라.

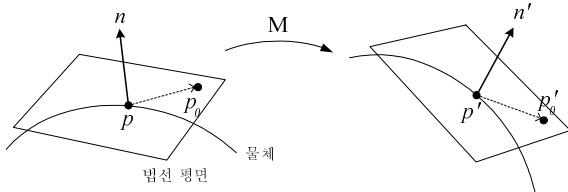


그림 1: 법선 벡터의 변환

- (a) 아래 글을 읽고 계속하여 $n' = (M^{-1})^t \cdot n$ 임을 수식을 통하여 정확히 증명하라.

그림 1에서와 같이 주어진 점 $p = (x \ y \ z \ 1)^t$ 에서의 법선 벡터가 $n = (n_x \ n_y \ n_z \ 0)^t$ 이라고 하자. 이때 점 p 에서의 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 n 이 변환 행렬 M 에 의해 각각 $p' = (x' \ y' \ z' \ 1)^t$, $p'_0 = (x'_0 \ y'_0 \ z'_0 \ 1)^t$, $n' = (n'_x \ n'_y \ n'_z \ 0)^t$ 으로 변환이 된다고 하면, $p' = M \cdot p$ 와 $p'_0 = M \cdot p_0$ 로부터 p 와 p_0 를 구해 위의 관계식에 대입하여 다음과 같은 식을 얻게 된다.

- (b) 법선 벡터 n 에 대하여 회전 변환을 가하는 문제를 생각해보자. 이때, M 을 아래와

같이 정의한다면,

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

이때 v_1, v_2, v_3 의 값은 어떤 값을 가질까?

- (c) 이 행렬의 왼쪽-위쪽의 3행 3열 부행렬의 세 개의 열벡터를 각각 a_1, a_2, a_3 라 할 때, 이 세 개의 벡터가 만족하는 수학적인 성질을 정확히 기술하라.
 (d) 문제 (b)의 조건 하에, 회전 행렬 M 에 대하여 행렬 $(M^{-1})^t$ 의 내용을 정확히 기술하라. 문제 (b)의 답의 내용을 반영할 것.

- 다음은 투영 변환에 관한 문제이다. 그림 2를 보고 답하라.

- (a) OpenGL API의 `gluPerspective(fovy, asp, n, f)` 함수를 호출할 경우 좌표가 (x_e, y_e, z_e) 인 EC의 점이 NDC의 (x_{nd}, y_{nd}, z_{nd}) 로 변환이 된다고 할 때, y_{nd} 를 x_e, y_e, z_e 를 사용하여 표현하라. 반드시 유도 과정을 기술할 것.
 (b) z_{nd} 의 경우 $z_{nd} = \alpha + \frac{\beta}{z_e}$ 와 같은 형태의 변환을 사용하여 변환을 한다. 이때 α 와 β 값을 유도하라.

- 다음은 OpenGL API를 사용한 2차원 뷰잉 (2D viewing)의 구현에 관한 문제이다. 그림 3(a)에 도시되어 있는 바와 같이 현재 세상 좌표계의 한 점 $p = (50, 50)$ 을 중심으로 길이가 각각 50인 두 선분이 수직으로 배치되어 있고, 한변의 길이가 10인 정사각형 다섯개가 p 를 중심으로 대각선으로 배치되어 있다 (모델링 좌표계와 세상 좌표계 모두 y 축이 위쪽 방향을 향함).

- (a) 그림 4에 주어진 프로그램에서 위의 내용처럼 선분과 정사각형들이 배치될 수 있

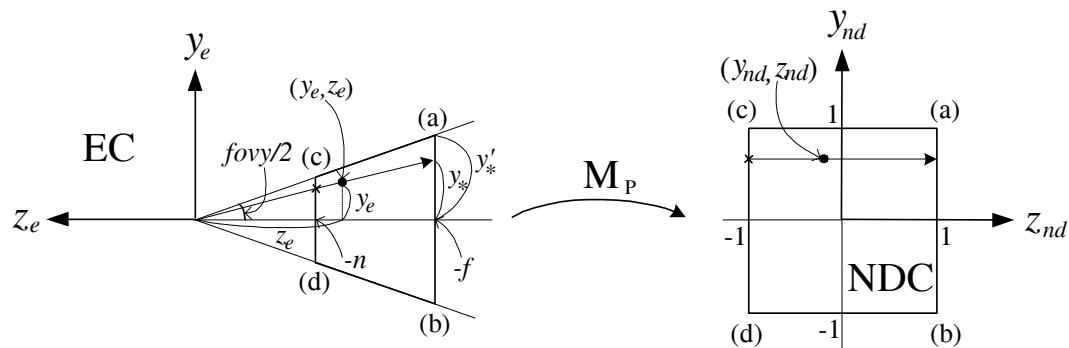


그림 2: 투영 변환 행렬의 유도

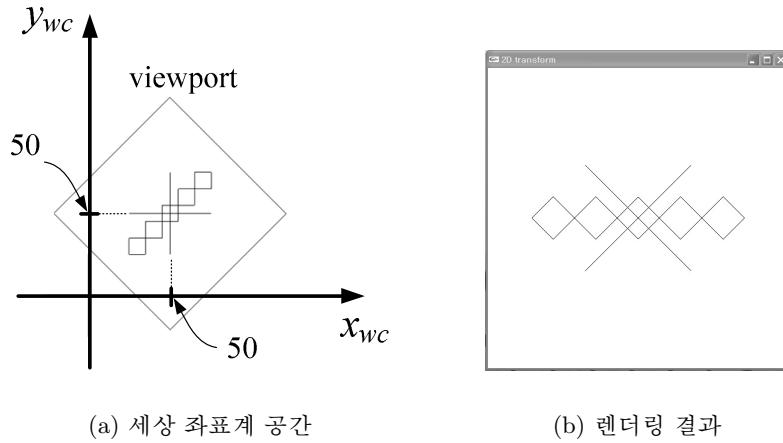


그림 3: 이차원 뷰잉

도록 하려면, 프로그램의 (A)에 어떤 내용이 들어가야할지, 이동, 회전, 크기 변환 등에 관련된 OpenGL API 함수들을 사용하여 기술하라.

- (b) 이제 세상 좌표계에서 p 를 중심으로 한 변의 길이가 100인 정사각형을 45도 회전 시켜 생성되는 영역을 뷰포트로 설정하여 한다. 그림 3(b)와 같이 렌더링이 되도록 하려면, 프로그램의 (B)에 어떤 문장(들)이 들어가야 할지 이동, 회전, 크기 변환 등에 관련된 OpenGL API 함수들을 사용하여 기술하라.
- (c) 위 문제와 같은 문제에 대해, 아래와 같은 문장을 반드시 사용하여 프로그램의 (B)에 들어갈 문장(들)을 기술하라.
`glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);`
- (d) 이 프로그램의 (A)와 (B)에 적절한 문장을 삽입하였다고 가정하자. 이때 만약 이 프로그램의 `glMatrixMode(GL_MODELVIEW);` 문장과

그 다음 문장을 제거한다면, 렌더링 결과에 어떤 변화가 있을까? 그 이유는?

- (e) 이 프로그램이 수행될 때 사용되는 4행 4열 뷰포트 변환 행렬 MVP 를 기술하라 (원도우 좌표계에서 깊이 정보는 0과 1사이로 매핑이 되도록 하고 (가까운 쪽이 0), 딥만 적을 것).

4. 다음은 3차원 공간에서의 뷰잉 변환에 관한 문제이다.

- (a) 카메라를 세상 좌표계의 좌표점 $(-10, 0, 0)$ 에 위치 시킨 후, 오른쪽, 위쪽, 그리고 시선의 방향을 각각 $(0, 0, -1)$, $(0, 1, 0)$, $(-1, 0, 0)$ 인 방향으로 설정하였다고 가정하자. 이 경우 뷰잉 변환을 설정해주는 코드를 이동, 회전, 크기 변환 등에 관련된 OpenGL API 함수들을 사용하여 구현하라.
- (b) 위 문제와 동일한 상황에서 `gluLookAt`(*) 함수를 사용하여 뷰잉 변환을 수행하라.

```

void draw_box(void) {
    glBegin(GL_QUADS);
    glVertex2f(0.5, 0.5);
    glVertex2f(-0.5, 0.5);
    glVertex2f(-0.5, -0.5);
    glVertex2f(0.5, -0.5);
    glEnd();
}

```

```

void draw_cross(void) {
    glBegin(GL_LINES);
    glVertex2f(-0.5, 0.0);
    glVertex2f(0.5, 0.0);
    glVertex2f(0.0, -0.5);
    glVertex2f(0.0, 0.5);
    glEnd();
}

```

```

void display(void) {
    int i;

    glClear(GL_COLOR_BUFFER_BIT);

    glViewport(0, 0, 500, 500);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    (B)

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    for (i = 3; i < 8; i++) {
        glPushMatrix();
        (A)
        draw_box();
        glPopMatrix();
    }

    glPushMatrix();
    glTranslatef(50.0, 50.0, 0.0);
    glScalef(50.0, 50.0, 1.0);
    draw_cross();
    glPopMatrix();

    glFlush();
}

```

그림 4: 이차원 기하 변환을 위한 OpenGL 코드

(c) 그림 6의 코드는 gluLookAt(*) 함수의 구현 예를 보여주고 있다. 만약 이 문제의 카메라 설정에 맞게 이 함수를 호출하였다 면, 이 함수의 마지막 문장이 수행되는 시점에 변수 z[0], z[1], z[2]에는 어떤 값이 저장되어 있을지 정확히 기술하라.

(d) 이 프로그램에서 잘못된 부분을 명시하고 옳게 고쳐라.

5. 시험지 뒤에 첨부한 두 장의 프로그램은 계층적 모델링 변환을 통하여 자동차를 그려주는 OpenGL 프로그램이다. 이 프로그램을 보면서 아래 문제에 답하라.

(a) 이 프로그램은 그림 7에 주어진 자동차에 대한 트리 구조를 어떤 방식으로 탐색하면서 자동차를 그려주고 있는데, 이때 사용하는 탐색 방법과 가장 자연스럽게 대응이 되는 자료 구조는 무엇일까?

(b) OpenGL 렌더링 파이프라인에서 뷰 매핑 (view mapping)은 어떤 좌표계에서 어떤 좌표계로의 변환에 해당하는지, OC, EC, NDC, WdC, CC, MC, WC 등의 기호를 사용하여 답하라.

(c) 이 프로그램에서 뷰 매핑과 가장 관련이 많은 문장의 번호를 기술하라.

(d) “이 프로그램을 수행시킬 때, CC에서의 좌표 값이 항상 1이므로 혹시 렌더링 파이프라인에서 원근 나눗셈 연산이 누락된다 해도 큰 문제가 없다”라는 주장이 옳은지 아닌지 답하고 그 이유를 간략히 설명하라.

(e) 이 프로그램에서 자동차를 WC로 배치해 주는 모델링 변환이 강체 변환인지 아닌지 답하고 그 이유를 간략히 설명하라.

(f) 이 프로그램에서 MS 윈도우 프로그래밍에서 WM_PAINT 메시지가 올 때 수행이 될 것으로 생각이 되는 함수의 이름은?

(g) 이 프로그램에서는 사용자가 어떠한 방식으로 자동차를 앞으로 움직이게 할 수 있을지 정확히 기술하라.

(h) 세상 좌표계를 기준으로 카메라가 세상을 바라보는 방향 벡터를 정확히 기술하라.

(i) 59번의 이동 변환 관련 문장이 수행되기 직전의 모델뷰 행렬 스택의 내용을 정확하게 그려라. M_V , M_P , M_{VP} 와 그림 7의 행렬 기호 등을 적절히 사용하라. 현재 draw_wheel_and_nut(angle) 함수는 100번 문장에서 호출한 상태임.

- (j) 52번 문장의 `draw_wheel_and_nut(angle)` 함수에서 이상한 부분을 지적하고 수정하라. 이유를 설명할 것.
- (k) 이 프로그램에서는 사용자가 자동차를 앞뒤로 움직여도 바퀴는 회전을 하지 않는다. 만약 자동차가 앞뒤로 움직일 때 매 프레임마다 자동차 바퀴를 z 축 둘레로 `angle_z` 각도 만큼 회전을 시키려면 그림 7에서 어떤 변환 행렬(들)을 수정해야 할까?
- (l) 바로 위 문제의 회전 변환을 적용하려면 이 프로그램의 어느 부분에 어떤 문장을 추가해야 할지 정확하게 기술하라. 필요할 경우 문장 번호를 사용할 것.
- (f) 정상적인 원근 투영 변환을 할 때, 꼭지점의 동차 좌표의 w 좌표값이 1이 아닌 값이 나타날 수 있는 부분은?
- (g) 픽셀 쉐이더 (pixel shader)와 가장 밀접한 연관이 있는 부분은?

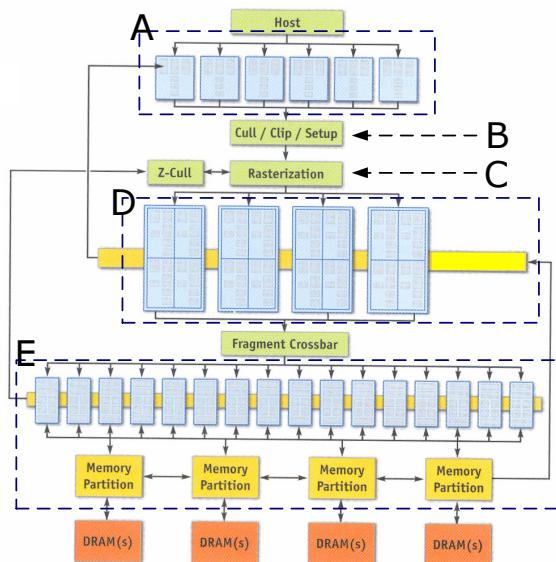


그림 5: NVIDIA GeForce 6 Series GPU 구조

6. (보너스 문제) 그림 5에는 NVIDIA사의 GeForce 6 Series의 GPU 구조가 도시되어 있다. 각 문제에 대하여 가장 연관성이 높은 파이프라인의 영역 이름 (A, B, C, D, E 중의 하나)을 기술하라.

- (a) 모델링 변환이 수행되는 부분은?
- (b) 원근 나눗셈 연산이 수행되는 부분은?
- (c) 깊이 버퍼 (depth buffer)를 사용하여 은연 제거를 수행하는 부분은?
- (d) 픽셀에 칠할 최종 색깔이 결정되는 부분은?
- (e) 연속 공간에서 주어진 기하 데이터가 이산 공간의 래스터 데이터로 변환이 되는 부분은?

```

void gluLookAt( GLdouble ax, GLdouble ay, GLdouble az, GLdouble bx, GLdouble by,
    GLdouble bz, GLdouble cx, GLdouble cy, GLdouble cz ) {
    GLdouble m[16]; GLdouble x[3], y[3], z[3]; GLdouble mag;

    z[0] = ax - bx; z[1] = ay - by; z[2] = az - bz;
    mag = sqrt( z[0]*z[0] + z[1]*z[1] + z[2]*z[2] );
    if (mag) { z[0] /= mag; z[1] /= mag; z[2] /= mag; }
    y[0] = cx; y[1] = cy; y[2] = cz;
    x[0] = y[1]*z[2] - y[2]*z[1]; x[1] = -y[0]*z[2] + y[2]*z[0];
    x[2] = y[0]*z[1] - y[1]*z[0];
    y[0] = z[1]*x[2] - z[2]*x[1]; y[1] = -z[0]*x[2] + z[2]*x[0];
    y[2] = z[0]*x[1] - z[1]*x[0];
    mag = sqrt( x[0]*x[0] + x[1]*x[1] + x[2]*x[2] );
    if (mag) { x[0] /= mag; x[1] /= mag; x[2] /= mag; }
    mag = sqrt( y[0]*y[0] + y[1]*y[1] + y[2]*y[2] );
    if (mag) { y[0] /= mag; y[1] /= mag; y[2] /= mag; }
#define M(row,col) m[row*4+col]

    M(0,0) = x[0]; M(0,1) = x[1]; M(0,2) = x[2]; M(0,3) = 0.0;
    M(1,0) = y[0]; M(1,1) = y[1]; M(1,2) = y[2]; M(1,3) = 0.0;
    M(2,0) = z[0]; M(2,1) = z[1]; M(2,2) = z[2]; M(2,3) = 0.0;
    M(3,0) = 0.0; M(3,1) = 0.0; M(3,2) = 0.0; M(3,3) = 1.0;
#undef M
    glMultMatrixd( m );
    glTranslated( -ax, -ay, -az );
}

```

그림 6: gluLookAt(*) 함수

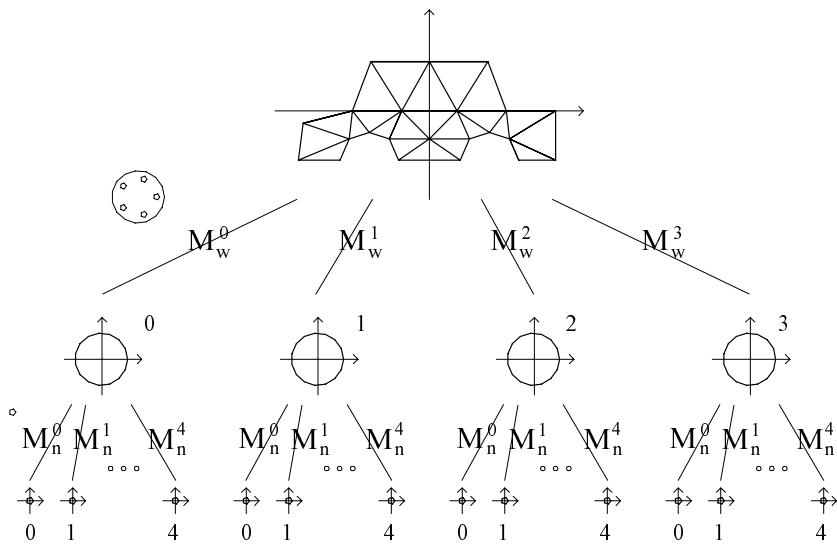
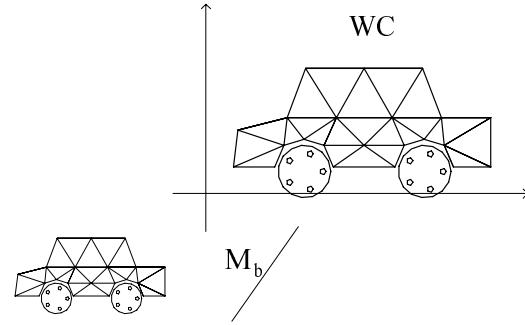


그림 7: 자동차의 계층적 표현

```

1 #include <stdio.h>
2 #include <math.h>
3 #include <GL/glut.h>
4
5 #define MAX_POLY 200
6 #define MAX_VERT 20
7 #define MAX_PATH 1000
8
9 #define DRAW_CAR_DUMMY 2001
10 #define DRAW_CAR_CORRECT 2002
11
12 typedef struct {
13     int invertex;
14     float poly[MAX_VERT][3];
15 } mypolygon;
16
17 mypolygon body[MAX_POLY], wheel[MAX_POLY], nut[MAX_POLY];
18 int npolyb, npolyw, npolyn;
19
20 float path[MAX_PATH][3];
21 int npath, path_exist, drawing_state = DRAW_CAR_CORRECT;
22
23 double dist;
24
25 int prev_i, cur_i = 0;
26 int rightbutbpressed = 0;
27
28 void read_object(char *file, mypolygon *object, int *npoly);
29 void read_path(char *file);
30 void read_objects(void);
31 void draw_axes(void);
32 void draw_path(void);
33 void draw_ground(void);
34
35 void draw_body(void) {
36     // Draw the body.
37     ...
38 }
39
40 void draw_wheel(float angle) {
41     // Draw the wheel.
42     ...
43 }
44
45 void draw_nut(void) {
46     // Draw the nut.
47     ...
48 }
49
50 #define rad 1.7
51 #define vw 1.0
52 void draw_wheel_and_nut(float angle) {
53     int i;
54     draw_wheel(angle); // draw wheel object
55     for (i = 0; i < 5; i++) {
56         // nut i
57         glPushMatrix();
58         glTranslate(rad-0.5, 0, vw); // rad = 1.7, vw = 1.0
59         glRotatef(72.0*i, 0.0, 0.0, 1.0);
60         draw_nut(); // draw nut object
61         glPopMatrix();
62     }
63 }
64
65 #define TO_DEG 57.29579
66 void normalize_vec3(float *v) {
67
68     ...
69 }
70
71 float dot_prod_vec3(float *u, float *v) {
72 }
73
74 float compute_length_mul_two_vec3(float *u, float *v) {
75
76     ...
77 }
78
79 float angle_between_two_vec3(float *u, float *v) {
80     ...
81 }
82
83 void cross_prod_vec3(float *u, float *v, float *n) {
84     // ???
85 }
86
87 float wheel_rot_angle_in_y(void) {
88     ...
89 }
90
91 float wheel_rot_angle_in_z(void) {
92     ...
93 }
94
95 void draw_car_dummy(void) {
96     draw_body(); // draw body object
97     glPushMatrix();
98     glTranslate(-3.9, -3.5, 4.5);
99     draw_wheel_and_nut(0.0); // wheel 0
100    glPopMatrix();
101
102    glPushMatrix();
103    glTranslate(-3.9, -3.5, 4.5);
104    draw_wheel_and_nut(0.0); // wheel 1
105    glPopMatrix();
106
107    glPushMatrix();
108    glTranslate(-3.9, -3.5, -4.5);
109    glScalef(1.0, 1.0, -1.0);
110    draw_wheel_and_nut(0.0); // wheel 2
111
112    glPopMatrix();
113
114    glPushMatrix();
115    glTranslate(3.9, -3.5, -4.5);
116    glScalef(1.0, 1.0, -1.0);
117    draw_wheel_and_nut(0.0); // wheel 3
118
119 }
120
121 void set_up_rot_mat(float *m, int i) {
122
123 }
124
125 void draw_fence(GLfloat r, GLfloat g, GLfloat b) {
126     ...
127 }

```

```

127 }
128 void draw_fences(void) {
129 ...
130 }
131 }

132 void draw_world (void) {
133     GLfloat m[16];
134     glMatrixMode(GL_MODELVIEW);
135     glLoadIdentity();
136     gluLookAt(-15.0, 20.0, 40.0, path[cur_i][0], 4.89, path[cur_i][2], 0.0, 1.0, 0.0);
137     gluPerspective(30.0, width/(double) height, 1.0, 150.0);
138 }

139 void draw_ground();
140 void draw_fences();
141 void draw_axes();
142 void draw_axis();
143 if (path_exist) draw_path();
144 set_up_rot_mat(m, cur_i);
145 glPushMatrix();
146 glTranslate(path[cur_i][0], 4.89, path[cur_i][2]);
147 glPushMatrix();
148 glTranslatef(path[cur_i][0], 4.89, path[cur_i][2]);
149 glMultMatrixf(m);
150 draw_car_dummy();
151 glPopMatrix();
152 glPopMatrix();
153 }

154 void render(void) {
155     glClear(GL_COLOR_BUFFER_BIT);
156     draw_world();
157     glutSwapBuffers();
158 }

159 }

160 void keyboard (unsigned char key, int x, int y) {
161 ...
162 }

163 }

164 int prevx_mouse;
165 void mousepress(int button, int state, int x, int y) {
166     if ((button == GLUT_RIGHT_BUTTON) && (state == GLUT_DOWN)) {
167         prevx.mouse = x;
168         rightbuttonpressed = 1;
169     }
170     else if ((button == GLUT_RIGHT_BUTTON) && (state == GLUT_UP))
171         rightbuttonpressed = 0;
172 }

173 }

174 void mousemove(int x, int y) {
175     double deltax;
176 }

177 if (rightbuttonpressed) {
178     deltax = x - prevx.mouse;
179     prevx.mouse = x;
180     if ((cur_i + deltax > 0) && (cur_i + deltax < npath-1)) {
181         prev_i = cur_i; cur_i += deltax;
182         dist = sqrt((path[cur_i][0]-path[prev_i][0])*(path[cur_i][0]-path[prev_i][1])
183 [0]) +
184         (path[cur_i][1]-path[prev_i][1])*(path[cur_i][1]-path[prev_i][2]) +
185         (path[cur_i][2]-path[prev_i][2])*(path[cur_i][2]-path[prev_i][2]);
186     if (deltax < 0.0) dist *= -1.0;

```

(43-170) 기초 컴퓨터 그래픽스 중간고사

(담당교수: 임인성)

학과: _____ 학번: _____ 이름: _____

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.

- 그림 1에서와 같이 원편의 윈도우의 내용을 오른쪽 윈도우로 매핑해주는 2차원 변환에 대한 3행 3열 행렬 M 을 기본 아핀 변환의 합성을 통하여 표현하고, 전체 곱 행렬을 구하라.

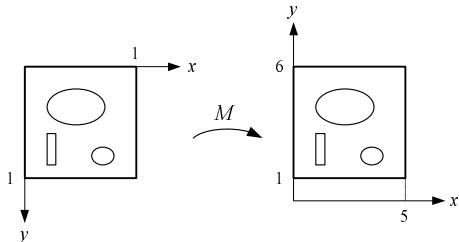


그림 1: 윈도우 매핑

- 그림 2는 모델링 좌표계 (MC)에 정의되어 있는 2차원 물체를 세상 좌표계 (WC)로 배치해 주는 과정이 도시되어 있다. 사용자의 의도대로 움직이는 물체의 위치와 방향이 WC에서 점 $e = (e_x \ e_y)$ 와 길이가 1이고 서로 수직인 두 벡터 $u = (u_x \ u_y)$ 와 $v = (v_x \ v_y)$ 로 표현되어 있다고 할 경우, 이때 필요한 모델링 변환에 대한 3행 3열 행렬 M 을 세 개의 기본 변환 행렬의 곱으로 표현하라 (각 3행 3열 행렬의 내용을 정확히 기술하되, 전체 곱 행렬은 계산할 필요가 없음).

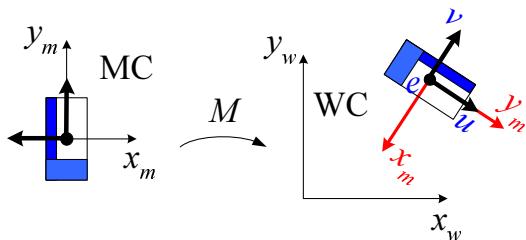


그림 2: 2차원 모델링 변환

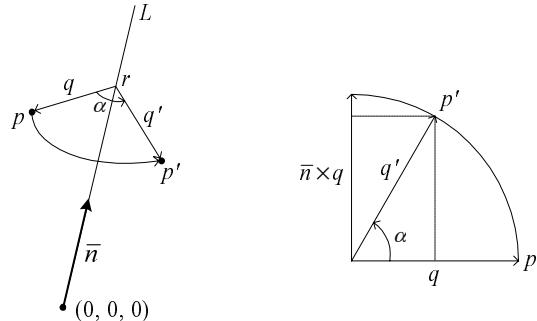


그림 3: 회전 변환의 유도

- 다음은 3차원 공간에서의 회전 변환 행렬의 유도에 관한 문제이다. 그림 3의 왼쪽에 원점을 지나고 단위 벡터 $\bar{n} = (\bar{n}_x \ \bar{n}_y \ \bar{n}_z)^t$ 의 방향으로 뻗은 직선 L 둘레로 한 점 $p = (x \ y \ z)^t$ 를 각도 α 만큼 회전하여 점 $p' = (x' \ y' \ z')^t$ 을 얻는 과정이 도시되어 있다. 이때 이 점은 직선 L 에 수직인 평면 위에서 회전을 하는데, 이 평면과 L 이 만나는 점을 r 이라 하고, r 에서 p 와 p' 를 향하는 벡터를 각각 q 와 q' 이라 하자.

- 벡터 q' 을 벡터 q 와 \bar{n} , 그리고 회전 각도 α 를 사용하여 표현하라.
- 벡터 p' 은 $p' = (A) + (p - (\bar{n} \cdot p)\bar{n}) \cos \alpha + (\bar{n} \times p) \sin \alpha$ 와 같이 표현할 수 있는데, 이 때 (A) 에 들어갈 수식을 기술하라.
- 바로 위 문제의 식은 주어진 점 p 에 대한 회전 변환을 통하여 결과 점 p' 를 어떻게 구하는지에 대한 정보를 제공하고 있다. 이 식중 $(\bar{n} \times p) \sin \alpha$ 은 $\{(\sin \alpha) S\}p$ 와 같이 3행 3렬 행렬 S 와 벡터 p 의 곱으로 표현이 가능한데, 이때 행렬 S 의 내용을 유도하라.

- 다음은 뷰잉 변환에 관한 문제이다.

- 세상 좌표계에서의 의미를 가지는 한 점 $e = (ex \ ey \ ez)^t$ 를 원점으로 서로 수직이고 길이가 1인 세 벡터 $u = (ux \ uy \ uz)^t$, $v = (vx \ vy \ vz)^t$, $n = (nx \ ny \ nz)^t$ 를 통

여 카메라의 프레임이 정의되어 있다 (여기서 u 와 v 는 각각 카메라를 중심으로 오른쪽과 위쪽 방향을 나타내며, 카메라는 n 의 정반대 방향을 응시하고 있다). 이에 해당하는 뷰잉 변환 M_V 를 4행 4열의 두 행렬의 곱으로 표현하라 (각 행렬의 정확한 내용, 즉 16개의 원소의 내용을 정확히 기술할 것).

- (b) 위에서 구한 뷰잉 변환을 다음과 같은 OpenGL 함수를 사용하여 구현하려 한다. 프로그램이 올바르게 수행이 되려면 마지막에서 두 번째 문장이 수행될 때, 1차원 배열 $m1[16]$ 에 저장되어 있어야 할 내용을 $m1[0], m1[1], m2[2], \dots, m1[15]$ 순서대로 정확히 기술하라.

```
GLfloat m1[16], m2[16];
:
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMultMatrixf(m1);
glMultMatrixf(m2);
```

- (c) 만약 위의 코드를 아래와 같은 내용으로 바꿀 경우, $a0, a1, a2, \dots, a5$ 에는 어떤 값을 지정하면 될지 각 변수의 값을 정확히 기술하라.

```
GLfloat m1[16], m2[16];
:
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(ex, ey, ez, a0, a1,
a2, a3, a4, a5);
```

5. 그림 4는 미국 Pixar사의 유명한 렌더링 소프트웨어인 RenderMan의 API 함수를 사용하여 코딩한 예를 보여주고 있다. 각 질문에 가장 직접적으로 연관된 문장의 번호 (각 문장의 왼쪽의 번호)를 기술하라 (각 문제에 대해 두 개 이상의 문장이 해당될 경우 모두 기술할 것).

- (a) 모델링 변환을 수행하고 있는 문장은?
(b) 원근 나눗셈 연산에 가장 직접적으로 영향을 미치는 문장은?
(c) 각 물체를 그리라는 명령을 수행할 때, OpenGL 관점에서 생각할 경우 모델뷰 행렬 스택의 탑의 행렬에 영향을 미칠 수 있는 문장은?
(d) OpenGL의 gluLookAt(*) 함수와 가장 밀접한 연관이 있는 문장은?

6. 다음은 OpenGL에서 사용하는 기하 파이프라인에 대한 단답식 문제이다. CC, EC, MC, NDC, WC, WdC 등의 좌표계 이름을 참고하여 적절히 답하라.

- (a) 그림 5의 A로 표시된 지점에 해당하는 OpenGL 좌표계의 이름은?
(b) 그림 5의 B로 표시된 지점에 해당하는 OpenGL 좌표계의 이름은?
(c) 그림 5의 C로 표시된 지점에 해당하는 OpenGL 좌표계의 이름은?
(d) OpenGL 렌더링 파이프라인에서 꼭지점별 쉐이딩 계산을 수행할 경우 어느 좌표계에서 꼭지점에 대한 색깔 계산이 일어날까?
(e) 정상적인 기하 변환을 할 경우 전체 파이프라인 과정에서 항상 강체 변환이 사용되는 부분은 어느 좌표계에서 어느 좌표계로의 변환에 해당하는가?
(f) OpenGL 렌더링 파이프라인에서 NDC까지 변환되어 온 한 점 p 의 z 좌표가 가질 수 있는 좌표값의 범위는?

7. 다음은 계층적 모델링에 관한 문제이다. 그림 6의 프로그램을 보고 답하라.

- (a) 이 코드는 네 개의 물체 A, B, C, 그리고 D로 구성된 복합 물체를 세상 좌표계로 변환해주고 있다. 각 물체에 대한 연관 관계를 트리 구조로 표현하라. 이때 트리의 노드는 해당 물체이어야 함.
(b) 7번 문장의 회전 변환이 영향을 미치는 물체를 모두 기술하라.
(c) 물체 B의 경우 자신의 모델링 좌표계에서 정의된 물체가 세상 좌표계로 변환될 때, 몇 배로 확대 또는 축소가 될까?
(d) 위 코드의 문장중 제거를 해도 문맥상 전혀 문제가 없는 문장이 있는데, 제거 가능한 문장 번호를 모두 기술하라.
(e) 만약 물체 B가 정의된 모델링 좌표계 공간의 한 점을 물체 C가 정의된 모델링 좌표계로 변환을 해야 할 경우 이에 대한 변환 행렬 M 을 $M_{(i)}$ 또는 $M_{(i)}^{-1}$ 행렬의 곱으로 표현하라 (여기서 $M_{(i)}$ 와 $M_{(i)}^{-1}$ 는 각각 i 문장을 수행시킬 때 계산이 되는 기본 행렬 또는 그것의 역행렬을 의미함).

```
(01) void main(void) {
        RtFloat fov = 45, intensity1 = 0.08, intensity2 = 0.8;
        :
(02) RtColor opacity1 = {0.3, 0.3, 0.3};
(03) char *txtname = "redchecker";

(04) RiBegin(RI_NULL);
(05) RiFormat(480, 360, 1);
(06) RiPixelSamples(2, 2);
(07) RiFrameBegin(1);
(08) RiDisplay("3spheres.tif", "file", "rgba", RI_NULL);
(09) RiProjection("perspective", "fov", &fov, RI_NULL);
(10) RiRotate(-110.0, 1, 0, 0);
(11) RiTranslate(0.0, -2.0, 8.0);
(12) RiWorldBegin();
(13)     RiLightSource("ambientlight", "intensity", &intensity1, RI_NULL);
        :
(14)     RiAttributeBegin();
(15)         RiTranslate(-2.25, 0.0, 2.0);
(16)         RiColor(sphcolor1);
        :
(17)         RiSphere(1.0, -1.0, 1.0, 360.0, RI_NULL);
(18)     RiAttributeEnd();
(19)     RiAttributeBegin();
(20)         RiTranslate(0.0, 0.0, 2.0);
(21)         RiColor(sphcolor2);
(22)         RiShadingRate(0.25);
(23)         RiShadingInterpolation("constant");
(24)         RiSurface("screen", "Ka", &Ka2, "Kd", &Kd2);
(25)         RiSphere(1.0, -1.0, 1.0, 360.0, RI_NULL);
(26)     RiAttributeEnd();
(27)     RiWorldEnd();
(28) RiFrameEnd();
(29) RiEnd();
(30) }
```

그림 4: RenderMan 프로그램 예

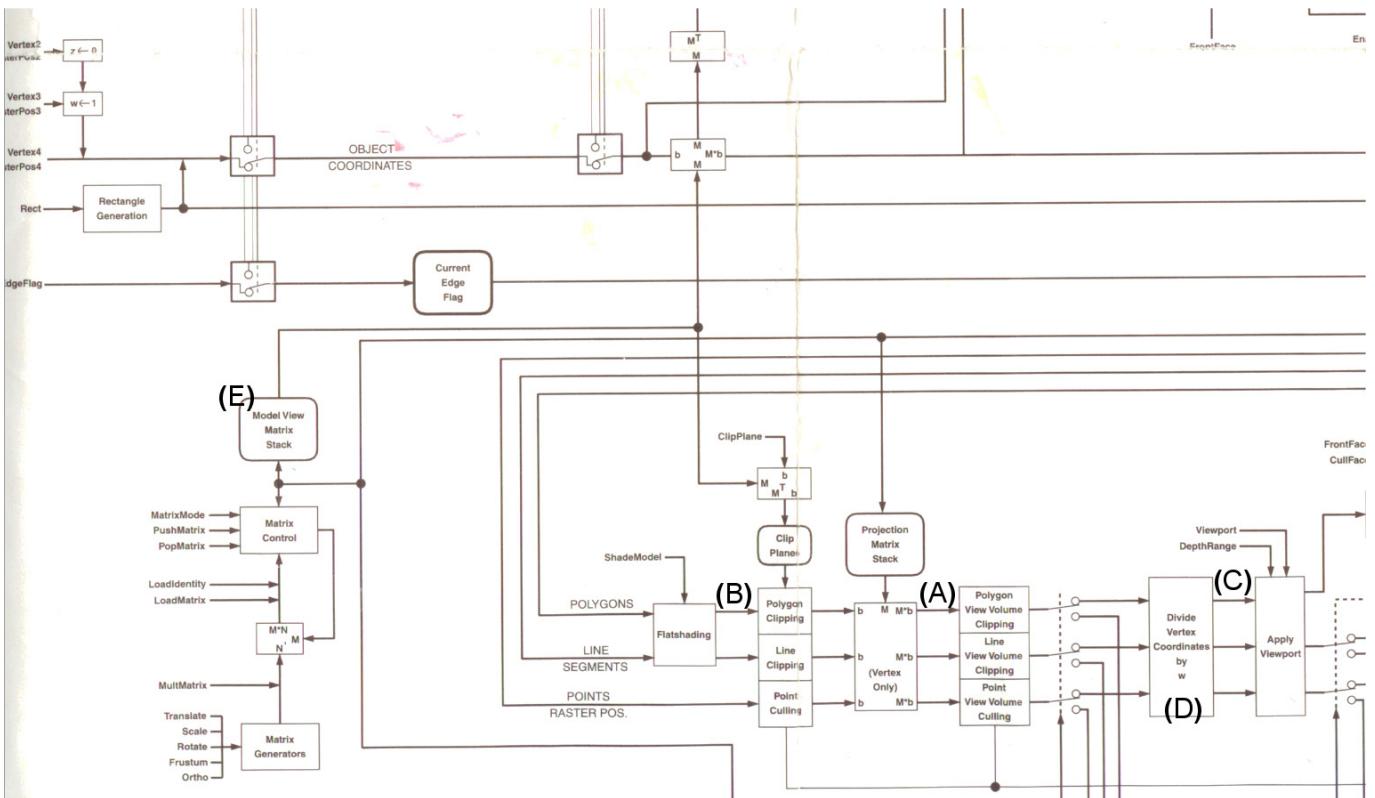


그림 5: OpenGL 파이프라인

```

(01) glPushMatrix();
(02) glRotatef(angle, 0.0, 1.0, 0.0);
(03) glTranslatef(0.0, 0.0, 5.0);
(04) glScalef(6.5, 6.5, 6.5);
(05) draw_cow(); // draw object A
(06) glPushMatrix();
(07) glRotatef(10.0*angle, 1.0, 0.0, 0.0);
(08) glTranslatef(-0.1, 0.0, 0.3);
(09) glScalef(0.025, 0.025, 0.025);
(10) draw_box(); // draw object C
(11) glPushMatrix();
(12) glRotatef(40.0*angle, 1.0, 0.0, 0.0);
(13) glTranslatef(0.0, 4.0, 0.0);
(14) glScalef(0.4, 0.4, 0.4);
(15) draw_box(); // draw object D
(16) glPopMatrix();
(17) glPopMatrix();
(18) glTranslatef(0.15, 0.3, 0.0);
(19) glScalef(0.3, 0.3, 0.3);
(20) draw_cow(); // draw object B
(21) glPopMatrix();

```

그림 6: 계층적 모델링의 예

기초 컴퓨터 그래픽스 중간고사 (43-170)

학과: _____ 학번: _____ 이름: _____

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.

1. 다음 각 설명과 가장 밀접한 관련이 있는 Win32 프로그래밍 모델에서의 event message ID를 기술하라 (대소문자를 구별할 것).

- (a) OpenGL 시스템에서 사용할 pixel format을 결정해주는 초기 코드를 수행하기에 가장 적합한 메시지
- (b) glutPostRedisplay() 함수가 궁극적으로 발생시키고자 하는 메시지
- (c) 주로 윈도우를 닫기 위해 윈도우즈 시스템 메뉴나 클로우즈 아이콘 등을 클릭했을 때 발생되는 메시지

2. 아래의 행렬 M_2 에 해당하는 3차원 공간에서의 기하 변환을 생각하자.

$$M_2 = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- (a) 3차원 공간에서 동일한 직선상에 존재하는 세 점 p_0, p_1, p_2 에 대하여 $\overline{p_0p_2} : \overline{p_0p_1} = 3 : 1$ 이라 하자. p'_0, p'_1, p'_2 를 각각 위의 세 점을 M_2 에 의해 변환한 점이라 할 경우 비율 $\overline{p'_0p'_2} : \overline{p'_0p'_1}$ 은 얼마가 될까?
- (b) M_2 는 강체 변환에 대한 행렬일까? 예 또는 아니오로 답하고 그 이유를 선형대수의 용어를 사용하여 답하라.
- (c) M_2 의 역행렬을 두 개의 기본 변환 행렬의 곱으로 표현하라.
- (d) OpenGL 파이프라인에서는 CC, EC, MC, NDC, WC, WdC 등의 좌표계를 사용하는데, 이 문제의 M_2 는 어느 좌표계에서 어느 좌표계로의 기하 변환과 가장 밀접한 연관이 있을까?

3. 그림 1에서 주어진 점 $p = (x \ y \ z \ 1)^t$ 을 $p' = (x' \ y' \ z' \ 1)^t$ 로 변환해주는 4행 4열의 투영 변환 행렬 M_3 을 유도하라.

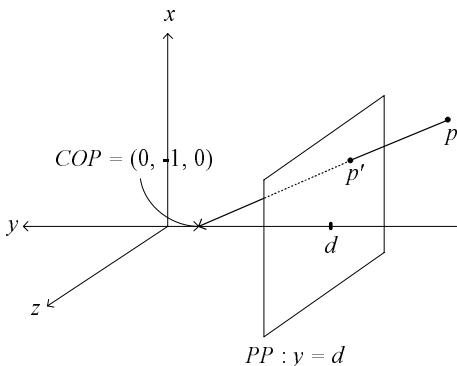


그림 1: 원근 투영 변환 문제

4. 그림 2에서 Frame 1과 Frame 2는 각각 원점을 중심으로 하고 세 개의 벡터 $u = (u_x \ u_y \ u_z)^t$, $v = (v_x \ v_y \ v_z)^t$, $n = (n_x \ n_y \ n_z)^t$ 와 $u' = (u'_x \ u'_y \ u'_z)^t$, $v' = (v'_x \ v'_y \ v'_z)^t$, $n' = (n'_x \ n'_y \ n'_z)^t$ 에 의해 정의가 되어 있다 (여기서 각 프레임의 세 벡터는 서로 수직인 단위 벡터들임). 지금 Frame 1을 Frame 2로 맞추어주는 변환에 해당하는 4행 4열 행렬 M_4 를 두 개의 회전 변환의 곱의 형태 $M_4 = R_1 \cdot R_2$ 로 표현할 경우 각 4행 4열 행렬 R_1 과 R_2 를 정확히 기술하라. 또한 M_4 행렬의 왼쪽-위의 3행 3열의 부행렬의 내용을 여섯 개의 벡터 u, v, n, u', v', n' 들의 곱 ·이나 내적 ◦을 사용하여 간결하게 표현하라.
5. 다음은 OpenGL에서 사용하는 기하 파이프라인에 대한 단답식 문제이다. CC, EC, MC, NDC, WC, WdC 등의 좌표계 이름을 참고하여 적절히 답하라.
 - (a) 3차원 공간에서 $[-1, 1] \times [-1, 1] \times [-1, 1]$ 영역만 고려하는 좌표계는 어느 좌표계일까?
 - (b) 연속 공간에서 꼭지점들로 정의된 다각형이 이산적인 픽셀들로 쪼개지는 좌표계는 어느 좌표계일까?

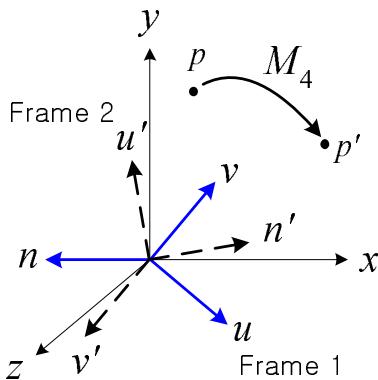


그림 2: 프레임 변환 문제

- (c) 카메라 필름에 맺힌 상을 인화지에 확대 및 인화해주는 과정은 어느 좌표계에서 어느 좌표계로의 변환에 해당하는가?
- (d) OpenGL에서 기하 변환을 위하여 주로 사용하는 두 개의 행렬 스택과 가장 연관성이 적은 좌표계는?
- (e) 정상적인 렌더링을 할 경우 전체 파이프라인 과정에서 항상 강체 변환이 사용되는 부분은 어느 좌표계에서 어느 좌표계로의 변환에 해당하는가?
- (f) 정상적인 투영 변환을 할 경우, 경우에 따라 우리에게 익숙한 아핀 공간에서 벗어날 수도 있게 되는 좌표계는 어떤 좌표계일까?
- (g) 'Fixed-function pipeline' 중의 하나인 원래의 OpenGL 파이프라인에서 각 꼭지점에 대하여 Gouraud 쉐이딩을 하기 위한 라ighting 계산이 일어나는 좌표계는 어느 좌표계일까?
- (h) 아래의 코드는 OpenGL 파이프라인의 일부를 대치해주는 vertex shader의 예이다. 그 내용을 추측건대 line (a)는 어느 좌표계에서 어느 좌표계로의 변환에 해당할까?

```
struct _output {
    float4 position: POSITION;
    float4 p: TEXCOORD0;
    float3 n: TEXCOORD1;
};

_output main(
    float4 position: POSITION,
    float4 normal: NORMAL,
    uniform float4x4 ModelViewProj,
    uniform float4x4 ModelView,
    uniform float4x4 ModelViewIT) {
```

```
_output OUT;

OUT.position= mul(ModelViewProj,
    position); // line (a)
OUT.p = mul(ModelView, position);
OUT.n = mul(ModelViewIT,
    normal).xyz;
```

```
    return OUT;
}
```

- (i) 바로 위 문제에 연속하여, 그 뒤의 두 문장은 어느 좌표계에서 무엇을 하기 위한 준비 과정일까?
- (j) 원근 투영 변환을 할 경우 원근감이 생성이 되는데, 이러한 원근감은 정확히 어느 좌표계에서 어느 좌표계로 이동할 때 생기는가?
- (k) 다음의 MS Direct3D 문장은 어느 좌표계에서 어느 좌표계로의 변환에 가장 밀접한 연관이 있을까?

```
D3DXMATRIX out;
D3DXVECTOR3 eye(2, 3, 3);
D3DXVECTOR3 at(2, 3, 3);
D3DXVECTOR3 up(2, 3, 3);
D3DXMatrixLookAtLH(&out, &eye, &at,
&up);
```

- 6. 그림 3은 OpenGL의 직교 투영 함수 glOrtho(*)에 관한 그림이다. 한 학생이 이 함수가 호출될 때 생성되는 행렬 M₆을 다음과 같은 형태로 유도를 하였는데, 이 때 α , β , 그리고 γ 에 들어갈 내용은 무엇인지 유도 하라 (이 과정에서 투영 방향이 반대로 바뀌게 되는 것을 상기할 것).

$$M_6 = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{f-n} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & \alpha \\ 0 & 1 & 0 & \beta \\ 0 & 0 & -1 & \gamma \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- 7. 다음은 수업 시간에 설명한 OpenGL 프로그램의 일부이다.

```
typedef struct _scene {
    int nobject;
    char *fname[256];
    Object *objects;
    int nlight;
    Light *lights;
} Scene;
```

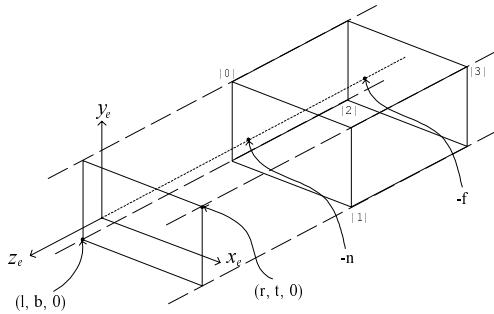


그림 3: glOrtho(l, r, b, t, n, f) 함수

```

typedef struct _cam {
    float pos[3];
    float uaxis[3], vaxis[3], naxis[3];
    GLfloat mat[16];      (A)
    GLfloat mat_inv[16];
    int move, upanddown;
    GLdouble fovy, aspect, near_c, far_c;
} Camera;

void reset_camera(Camera *cam) {
    float prp[3], vrp[3], vup[3];
    float x[3], y[3], z[3], mag;

    printf("[Proj. Ref. Pt.]: \n");
    scanf("%f %f %f", prp, prp+1, prp+2);
    printf("[View Ref. Pt.]: \n");
    scanf("%f %f %f", vrp, vrp+1, vrp+2);
    printf("[View-Up Vec.]: \n");
    scanf("%f %f %f", vup, vup+1, vup+2);
    :
    (B)
    cam->pos[0] = prp[0];
    cam->pos[1] = prp[1];
    cam->pos[2] = prp[2];
    cam->uaxis[0] = x[0];
    cam->uaxis[1] = x[1];
    cam->uaxis[2] = x[2];
    cam->vaxis[0] = y[0];
    cam->vaxis[1] = y[1];
    cam->vaxis[2] = y[2];
    cam->naxis[0] = z[0];
    cam->naxis[1] = z[1];
    cam->naxis[2] = z[2];

#define M(row,col) cam->mat[ (C) ]
    M(0,0) = x[0]; M(0,1) = x[1];
    M(0,2) = x[2]; M(0,3) = 0.0;
    M(1,0) = y[0]; M(1,1) = y[1];
    M(1,2) = y[2]; M(1,3) = 0.0;
    M(2,0) = z[0]; M(2,1) = z[1];
    M(2,2) = z[2]; M(2,3) = 0.0;
    M(3,0) = 0.0; M(3,1) = 0.0;
    M(3,2) = 0.0; M(3,3) = 1.0;
}

```

```

#undef M

printf("Field of View in y]: \n");
scanf("%f", &(cam->fovy));
:
}

void draw_axis (void) {
    glBegin(GL_LINES);
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(-5.0, 0.0, 0.0);
    glVertex3f(5.0, 0.0, 0.0);
    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(0.0, -5.0, 0.0);
    glVertex3f(0.0, 5.0, 0.0);
    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(0.0, 0.0, -5.0);
    glVertex3f(0.0, 0.0, 5.0);
    glEnd();
}

void cam_object_to_EC(Scene *scene) {
    glPushMatrix();
    glTranslatef( (D) );
    draw_object(&(scene->objects[6]));
    glPopMatrix();
}

void cam_object_to_there(Scene *scene,
                         Camera *cam) {
    glPushMatrix();
    glTranslatef(cam->pos[X], cam->pos[Y],
                cam->pos[Z]);
    glMultMatrixf(cam->mat_inv);
    cam_object_to_EC(scene);
    glPopMatrix();
}

void draw_scene(Scene *scene,
                Camera *cam) {
    int i, j;

    glViewport(530, 0, 750, 750);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(cam->fovy, cam->aspect,
                  cam->near_c, cam->far_c);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glMultMatrixf(cam->mat);
    glTranslatef(-cam->pos[X], -cam->pos[Y],

```

```

    -cam->pos[Z]);
}

draw_axis();

for (i = 0; i < scene->nobject; i++) {
    glPushMatrix();
    glMultMatrixf((GLfloat *)
        &(scene->objects[i].xform));
    draw_object(&(scene->objects[i]));
    glPopMatrix();
}
}

```

- (a) 어떤 순간에 (A)의 mat에 아래와 같은 행렬이 저장되어 있을 때, 매크로 그 아래 줄의 mat_inv에는 어떤 행렬이 저장되어 있어야 할까?

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

- (b) 함수 reset_camera(*)는 사용자로부터 카메라에 대한 성질을 입력 받아 Camera 타입의 변수에 적절한 내용을 저장해주는 역할을 한다. 여기서 (B)는 EC의 각 좌표축의 방향을 계산해주는 부분이다. prp = (10, 5, 5), vrp = (13, 5, 9), 그리고 vup = (0, 1, 0)과 같은 입력에 대해 (B) 부분이 수행이 된 후, z[0], z[1], z[2]에는 각각 어떤 값이 저장이 될까 (계산 과정을 명기할 것)?
- (c) 바로 위의 문제에 연속하여, x[0], x[1], x[2]에는 각각 어떤 값이 저장이 될까 (계산 과정을 명기할 것)?
- (d) 프로그램 매크로 상 (C)에 들어갈 C 언어 수식을 기술하라.
- (e) draw_axis(*)는 glVertex3f(*) 함수를 사용하여 좌표축을 그려주는 함수이다. 프로그램의 매크로 draw_scene(*) 함수의 draw_axis(); 문장은 CC, EC, MC, NDC, WC, WdC 등의 OpenGL 좌표계 중 어느 좌표계의 좌표축을 그리기 위한 것일까?
- (f) 만약 draw_scene(*) 함수 내에서 glViewport(530, 0, 750, 750); 문장을 glMatrixMode(GL_MODELVIEW); 문장의 바로 다음으로 옮길 경우 어떤 변화가 생길까?

(g) 지금 카메라의 움직임에 상관 없이 6번 물체 (scene->object[6])을 항상 카메라를 기준으로 오른쪽으로 3만큼, 위쪽으로 4만큼, 그리고 앞쪽으로 10만큼 들어간 위치에 그리려 한다 (예를 들어 ENG 카메라의 마이크의 일부가 항상 화면의 고정된 위치에 보이는 것처럼). 위의 코드에서 함수 cam_object_to_EC(*)를 사용하여 하는데, 이때 (D)에 들어갈 내용을 C 언어 문법에 맞게 기술하라.

(h) 바로 위 문제에서 설명한 바와 같이 카메라에 고정된 물체를 그리기 위해서는 cam_object_to_EC(scene); 과 같은 문장을 draw_scene(*) 함수 내에서 정확히 어떤 문장과 어떤 문장 사이에서 호출하는 것이 가장 자연스러울까?

(i) (이 문제는 바로 위의 두 문제와 독립적인 문제임) 만약 cam_object_to_there(scene, cam); 과 같은 문장을 draw_scene(*) 함수 내의 draw_axis(); 문장 바로 앞에서 수행시키려 한다. 이때 cam_object_to_there(*) 함수가 호출되어 cam_object_to_EC(scene); 문장이 수행되기 직전의 OpenGL 시스템의 모델 뷰 행렬 스택의 탑에는 어떤 내용의 4행 4열 변환 행렬이 저장되어 있을까?

(j) 바로 위 문제에서도 앞의 (g-h) 문제에서와 같이 카메라에 고정된 물체를 그리는 것을 목표로 한다. 만약 cam_object_to_there(*); 함수에서 glPushMatrix(); 문장과 glPopMatrix(); 문장을 제거해도 아무런 문제가 없을까? 예 또는 아니오로 답하고 그 이유를 설명하라.

8. 다음은 Direct3D로 작성한 프로그램의 일부이다 (보너스 문제).

```

protected void Render() {
    float angle = Environment.TickCount
        /500.0F;
    device.Clear(ClearFlags.Target,
        Color.Bisque, 1.0F, 0);
    device.BeginScene();
    device.Transform.World =
        Matrix.RotationY(angle);
    device.Transform.View =
        Matrix.LookAtLH(new Vector3(0,
        0.5F, -3), new Vector3(0, 0.5F,

```

```
    0), new Vector3(0, 1, 0));
device.Transform.Projection =
    Matrix.PerspectiveFovLH((float)
        Math.PI/4.0F, 1.0F, 1.0F, 5.0F);
device.SetStreamSource(0,
    vertices, 0);
device.DrawPrimitives(
    PrimitiveType.TriangleList, 0, 1);
device.EndScene();
device.Present();
}
```

- (a) 색깔 버퍼와 가장 관련이 있는 문장은?
- (b) OpenGL에서 x, y, z 좌표를 w 로 나누어 주는 연산과 가장 관련이 있는 문장은?
- (c) OpenGL의 기하 파이프라인에서 두 번째 만나는 스택과 직접적인 관련이 있는 문장을 모두 나열하라.
- (d) MC에서 WC로의 변환과 가장 관련이 있는 문장은?
- (e) OpenGL의 `glVertex3f(*)` 함수와 가장 관련이 있는 문장은?
- (f) 카메라에서 바라보는 방향을 변경하고자 할 때 내용을 수정해야하는 문장은?

- 답은 (3번째 문장)과 같은 식으로 할 것. 참고로 위 함수는 10개의 문장으로 구성되어 있음.

기초 컴퓨터 그래픽스 중간고사 (43-170)

반: _____ 학과: _____ 학번: _____ 이름: _____

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.

1. 다음 단답식 문제에 답하라.

- (a) Win32 프로그래밍 모델에서 각 설명에 해당하는 message ID를 기술하라.

(A) – This message is sent when the window is first produced, and gives you a chance to do any setup, initialization, or resource allocation.

(B) – This message is sent whenever your window's contents need repainting. This can occur for a number of reasons: the window was moved or resized by the user, another application popped up and obscured yours, and so on.

(C) – This message is sent to your window when the window is about to be killed. Usually, this is a direct result of the user clicking on the window's close icon or closing from the window's system menu. Either way, ...

- (b) 아래의 문장에 의하면 노란색의 보색(complementary color)은 어떤 색깔인가? RGB 색깔 모델의 값을 통하여 이유를 밝힐 것.

A pair of colors which can be additively combined to produce white light are called complementary colors, because together they complete the spectrum.

- (c) 아래의 문장에서 빈칸을 메꾸어라.

The vectors v_i are () if $\alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_m v_m = 0$

for constants α_i only if $v_1 = v_2 = \dots = v_m = 0$.

- (d) 동차 좌표로 표현된 3차원 공간의 점 $(1, 0, -3, 2)$ 와 동일한 점을 두 개 기술하라.
 (e) 어떤 4행 4열 행렬이 아핀 변환에 대한 변환 행렬이 되기 위한 조건은?
 (f) 만약 여러분이 glScalef(x, y, z) 함수를 가장 적은 회수의 덧셈/뺄셈, 곱셈, 그리고 나눗셈 연산을 사용하여 구현한다고 할 때 각각 몇 번씩 수행해야 할까? 여기서 스택에 있는 행렬은 임의의 값을 가질 수 있다고 가정하고, 답은 $+/-$ 번, $*=$ 번, $/=$ 번과 같이 기술하라.
 (g) 주어진 회전 변환 R 과 크기 변환 S 를 합성하여 원하는 변환을 만들어 내려 한다. 어떤 조건을 만족할 경우 이 두 변환을 행하는 순서가 중요하지 않게 될까?
 (h) 어떤 조건을 만족할 경우 주어진 3차원 아핀 변환 행렬이 순수한 회전 변환임을 알 수 있을까? 4행 4열의 행렬이 갖추어야 할 성질에 대하여 정확히 기술하라.
 (i) 원근 변환을 할 경우 기하 물체의 비율이 보존될까? 답을 말하고 그 이유를 밝혀라.
 (j) 다음 괄호 안에 들어갈 단어/문구/문장은?

투영 변환은 **(A)**, **(B)**, 그리고 **(C)**에 의해 정의된다. 직교 투영은 **(D)**인 경우의 평행 투영이다.

- (k) OpenGL의 뷰잉 파이프라인에서 '카메라의 위치와 방향을 설정'하는 변환은 꼭지점의 좌표를 어느 좌표계에서 어느 좌표계로 보내주는 변환인가?
 (l) OpenGL의 뷰잉 파이프라인에서 '사진의 크기 결정 및 인화' 과정과 관련이 있는 변환의 이름은 무엇인가?
 (m) OpenGL의 뷰잉 파이프라인에서 원근 투영시 비로소 어느 좌표계까지 왔을 때 원근감이 생성되는가?

2. 아래의 프로그램을 보고 답하라.

```

int angle_e = 0, angle_r = 0, angle_l =
0;
:
void keyboard(unsigned char key, int
x, int y) {
    switch (key) {
        case 'e':
            angle_e = (angle_e + 1)%360;
            glutPostRedisplay();
            break;
        case 'r':
            angle_r = (angle_r + 1)%360;
            glutPostRedisplay();
            break;
        case 'l':
            angle_l = (angle_l + 1)%360;
            glutPostRedisplay();
            break;
        case 'm':
            :
    }
}

void draw_box(void) {
    glBegin(GL_POLYGON);
    glVertex2f(0.5, 0.5);
    glVertex2f(0.5, -0.5);
    glVertex2f(-0.5, -0.5);
    glVertex2f(-0.5, 0.5);
    glEnd();
}

void practice_mode(void) {
    glPushMatrix();
    glTranslatef(250.0, 250.0, 0.0);
    glScalef(100.0, 100.0, 1.0);
    glRotatef(angle_e, 0.0, 0.0, 1.0);

    glPushMatrix();
    /* (A) */
    glScalef(1.0, 0.1, 1.0);
    glColor3f(0.0, 1.0, 0.0);
    draw_box()
    glPopMatrix();

    glPushMatrix();
}

```

```

glTranslatef(-0.5, 0.0, 0.0);
glRotatef(angle_l, 0.0, 0.0, 1.0);
glTranslatef(-0.2, 0.0, 0.0);
glScalef(0.5, 0.15, 1.0);
glColor3f(1.0, 0.0, 0.0);
draw_box();
glPopMatrix();

glPushMatrix(); /* (B) */
glTranslatef(0.5, 0.0, 0.0);
glRotatef(angle_r, 0.0, 0.0, 1.0);
glTranslatef(0.2, 0.0, 0.0);
glScalef(0.5, 0.15, 1.0);
glColor3f(0.0, 0.0, 1.0);
draw_box();
glPopMatrix(); /* (C) */

glPopMatrix();
}

```

여기서 `practice_mode()` 함수는 수업에서 예제로 설명한 프로그램에서와 같이 `practice` 모드에서 디스플레이 컬백 함수에 의해 호출이 된다. 위에서 보다시피 전역 변수 `angle_e`, `angle_r`, `angle_l`은 모두 0으로 초기화가 되어 있으며 사용자의 키 조작에 의해 키보드 컬백 함수 `keyboard()` 함수를 통하여 값이 변한다. 현재 500×500 크기의 윈도우가 화면에 떠 있으며(2차원 윈도우 좌표계는 MS 윈도우스 방식을 사용), 적절한 뷰잉 코드의 호출을 통하여 프로그래머는 이 윈도우 좌표계를 세상 좌표계로 사용할 수 있도록 하였다. 다시 말해서 위 코드의 변환은 모델링 좌표계에서 세상 좌표계로의 변환이다.

- (a) 이 프로그램이 처음 수행될 때 화면에 어떤 그림이 그려지는지 공학도 관점에서 가급적 정확하게 그려라.
- (b) 프로그램이 처음 수행이 된 후 사용자가 '1' 키와 'r' 키를 각각 45번 누른 후, 'e' 키를 90번 누를 경우 화면에 어떤 그림이 그려지는지 그려라.
- (c) 만약 `practice_mode()` 함수의 네 번째 줄에 있는 회전 함수를 (A) 위치로 옮긴 후 다시 컴파일 하여 바로 위의 문제처럼 키를 누를 경우 화면에 어떤 그림이 그려지는지 그려라.
- (d) 만약 원래의 프로그램에서 (B)와 (C) 줄에 있는 스택 푸쉬와 팝 함수를 지울 경우

어떤 현상이 일어나는지 밝히고 그 이유를 설명하라.

3. 다음은 뷰잉 변환에 관한 문제이다.

- 서로 수직이고 길이가 1인 세 벡터 $\mathbf{u} = (u_x \ u_y \ u_z)^t$, $\mathbf{v} = (v_x \ v_y \ v_z)^t$, $\mathbf{n} = (n_x \ n_y \ n_z)^t$ 가 주어진 점 $p = (e_x \ e_y \ e_z)^t$ 를 원점으로 하여 세상 좌표계에 떠 있다. 이 세 벡터의 방향이 눈 좌표계에서 각각 x , y , z 축의 방향이 되도록 해주는 뷰잉 변환에 대한 4행 4열 행렬 M_V 를 좌표계 변환 방식을 통하여 유도하라.
- 위에서 구한 행렬을 회전 변환 행렬 R 과 이동 변환 행렬 T 의 곱 $R \cdot T$ 의 형태로 분해하라. 반드시 유도 과정을 설명하라.
- 다음은 이동 변환과 회전 변환을 통하여 뷰잉 변환을 수행한 예를 보여주고 있다.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef(90.0, 1.0, 0.0, 0.0);
glTranslatef(-10.0, 0.0, 0.0);
```

이 코드에서 위의 이동 변환과 회전 변환 함수를 `gluLookAt()` 함수로 대치하려 할 경우 이 함수의 9개 인자를 정확히 기술하라. 어떤 방식으로 문제를 풀었는지 상세히 기술할 것.

- 다음은 직교 투영 변환과 관련한 문제이다. 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 기본 아핀 변환의 합성을 통하여 표현하라. 단 그림 1에서와 같이 z 축의 방향이 반대가 되도록 네 모서리를 마춰주어야 함.
- 그림 2는 과거에 사용되었던 PEXlib이라는 3차원 실시간 렌더링 API의 원근 투영에 관한 그림이다.

- PEXlib에서 OpenGL의 눈 좌표계와 대응이 되는 좌표계의 이름은 무엇일까?
- OpenGL에는 정규 디바이스 좌표계라는 좌표계에서 특정 영역을 사용하는데 PEXlib에서도 비슷한 과정이 존재함을 알 수 있다. 이와 관련하여 이 두 API의 차이점 두 가지를 기술하라.

- 다음은 Direct3D로 작성한 프로그램의 일부이다.

```
protected void SetupMatrices() {
    float angle = Environment.TickCount
        /500.0F;
    device.Transform.World =
        Matrix.RotationY(angle);
    device.Transform.View =
        Matrix.LookAtLH(new Vector3(0,
            0.5F, -3), new Vector3(0, 0.5F,
            0), new Vector3(0, 1, 0));
    device.Transform.Projection =
        Matrix.PerspectiveFovLH((float)
            Math.PI/4.0F, 1.0F, 1.0F, 5.0F);
}
```

```
protected void Render() {
    device.Clear(ClearFlags.Target,
        Color.Bisque, 1.0F, 0);
    device.BeginScene();
    SetupMatrices();
    device.SetStreamSource(0,
        vertices, 0);
    device.DrawPrimitives(
        PrimitiveType.TriangleList, 0, 1);
    device.EndScene();
    device.Present();
}
```

- 색깔 버퍼와 가장 관련이 있는 문장은?
- 원근 나눗셈과 가장 관련이 있는 문장은?
- OpenGL의 모델뷰 행렬 스택과 직접적인 관련이 있는 문장을 모두 나열하라.
- 모델링 좌표계에서 세상 좌표계로의 변환과 가장 관련이 있는 문장은?
- OpenGL의 `glVertex3f(*)` 함수와 가장 관련이 있는 문장은?
- 카메라에서 바라보는 방향을 변경하고자 할 때 내용을 수정해야하는 문장은?

- 답은 (함수 2 – 문장 3)과 같은 식으로 할 것. 참고로 (함수 1)과 (함수 2)는 각각 4개와 7개의 문장으로 구성되어 있음.

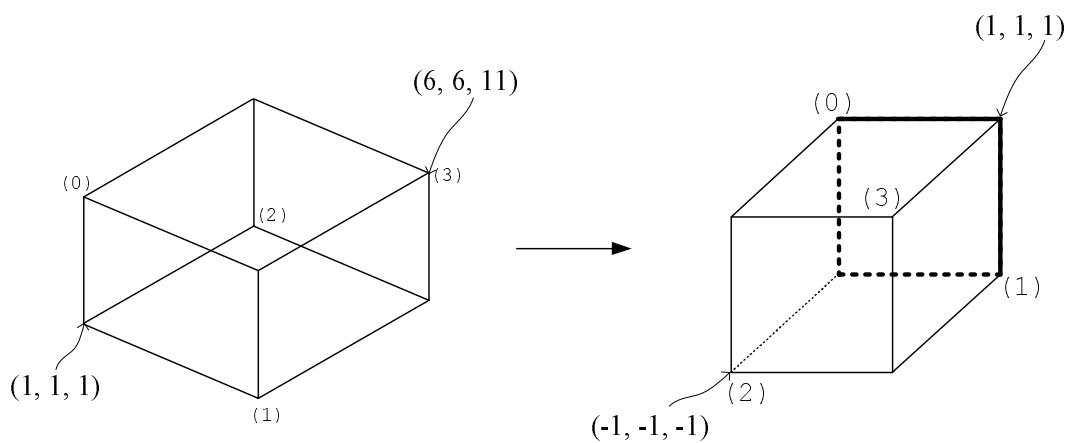


그림 1: 직교 투영 변환 문제

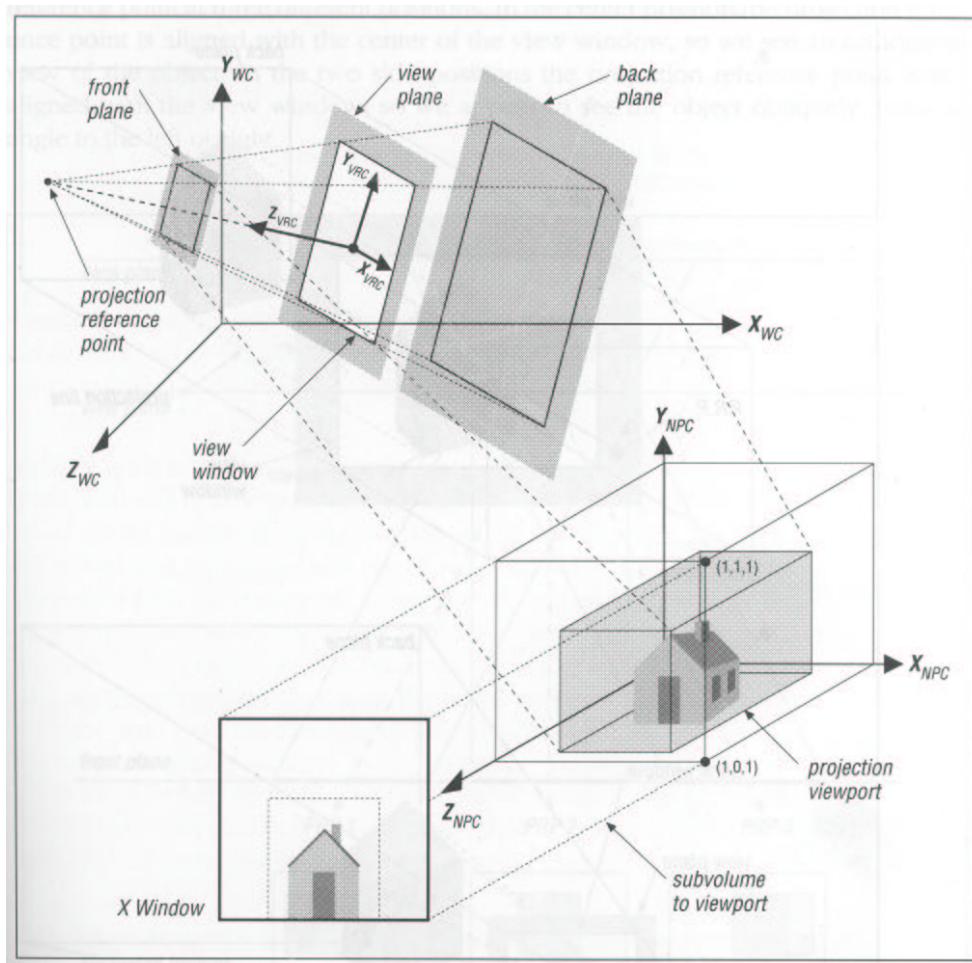


그림 2: PEXlib에서의 3차원 뷰 임

기초 컴퓨터 그래픽스 중간고사 (43-170)

반: _____ 학과: _____ 학번: _____ 이름: _____

1. 다음 단답식 문제에 답하라.

- (a) 좌표가 $(3.0, 6.0, -9.0, -2.0)$ 인 3차원 투영 공간의 점에 해당하는 아핀 공간의 점의 좌표 (x, y, z) 는?
- (b) RGB 모델로 값이 $(0.3, 0.5, 1.0)$ 인 색깔을 CMY 모델로 나타내면?
- (c) 3차원 아핀 변환을 나타내는 4행 4열 행렬 M 이 변환 후에도 물체의 크기와 모양을 보존해주기 위한 조건은?
- (d) gluPerspective($fovy$, asp , n , f) 함수에 해당하는 변환 행렬 M_{pers} 는 다음과 같다. 이 행렬을 이용하여 렌더링 시 원근감이 생성되는 과정을 설명하라.

$$\begin{bmatrix} \frac{\cot(\frac{fovy}{2})}{asp} & 0 & 0 & 0 \\ 0 & \cot(\frac{fovy}{2}) & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2nf}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

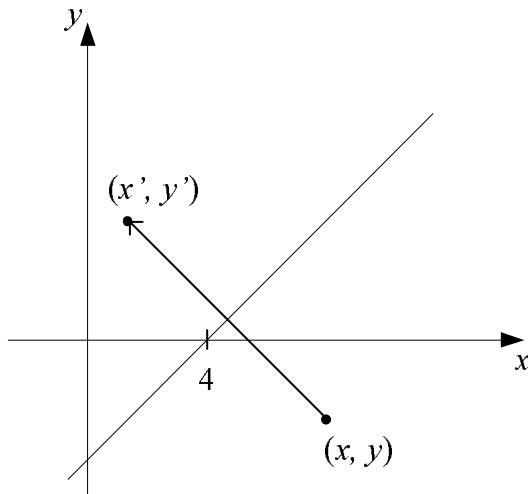


그림 1: 2차원 변환

- (e) 강체 변환은 아핀 변환 중 어떤 두 변환으로 합성을 할 수가 있는가?
- (f) 만약 여러분이 glscalef(x, y, z) 함수를 가장 적은 회수의 덧셈/뺄셈, 곱셈, 그리고 나눗셈 연산을 사용하여 구현한다고 할 때 각각 몇 번씩 필요할까? (여기서 스택에 있는 4행 4열 행렬은 임의의 값을 가질 수 있다고 가정함.)
- (g) OpenGL에서는 기본적으로 볼록 다각형만 지원을 하는데, 볼록 다각형의 수학적인 정의는 무엇인가?
- (h) 주어진 회전 변환 R 과 크기 변환 S 간에 교환 법칙이 성립하기 위한 조건은?
2. 그림 1에서와 같이 기울기가 45도인 직선 둘레로 반사를 시켜주는 아핀 변환을 유도하고, OpenGL 함수로 관련 부분을 구현하라.
3. 그림 2에는 한 카메라의 위치와 방향이 설정되어 있다. (바라보는 방향과 위쪽 방향은 각각 y_w 축과 x_w 축과 평행함.) 이 경우 뷔잉 변환 행렬

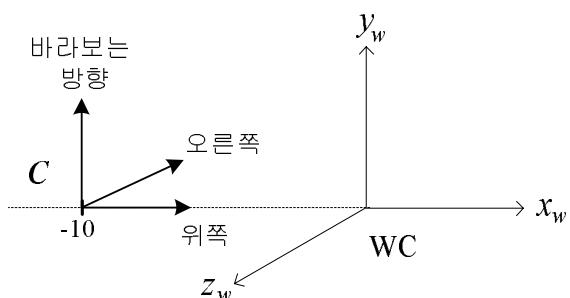


그림 2: 카메라 설정

M_V 를 구하라. (힌트: gluLookAt(*) 함수에 대한 유도 과정을 생각할 것.)

4. `glOrtho(-2.0, 2.0, -2.0, 2.0, 5.0, 10.0);`와 같은 OpenGL 함수를 호출할 경우 좌표가 (x_e, y_e, z_e) 인 EC의 점이 NDC의 (x_{nd}, y_{nd}, z_{nd}) 로 변환될 때 적용이 되는 4행 4열의 투영 변환 행렬 M_{ortho} 를 유도하라. (그림 3 참조)
5. 시험지 뒤에 첨부한 프로그램은 적절한 모델링 변환을 통하여 자동차를 그려주는 OpenGL 프로그램이다. 이 프로그램에 대하여 답하라.
 - (a) 이 프로그램은 그림 4에 주어진 자동차에 대한 트리 구조를 어떤 방식으로 탐색을 하고 있는가? 자료 구조 시간에 배운 용어를 사용할 것.
 - (b) 이 프로그램에서 원근 나눗셈(perspective division)과 가장 관련이 있는 문장의 번호는?
 - (c) 이 프로그램에서 MS 윈도우 프로그래밍에서 WM_PAINT 메시지가 올 때 수행이 될 것으로 생각이 되는 함수의 이름은?
 - (d) 이 프로그램에서 윈도우 프로그래밍에서 사용하는 PIXELFORMATDESCRIPTOR 자료 구조와 가장 관련이 있는 문장 번호는?
 - (e) 이 프로그램에서는 사용자가 어떤 방식으로 자동차를 앞으로 움직이게 할 수 있을까? 정확하게 기술할 것.
 - (f) 세상 좌표계를 기준으로 할 때 카메라가 세상을 바라보는 방향을 벡터로 표현하라.
 - (g) 눈 좌표계를 기준으로 할 때 카메라가 세상을 바라보는 방향을 벡터로 표현하라.
 - (h) 59번의 이동 변환 관련 문장이 수행되기 직전의 모델뷰 행렬 스택의 내용을 정확하게 그려라. M_V , M_P , M_{VP} 와 그림 4의 행렬 기호 등을 적절히 사용하라. 현재 `draw_wheel_and_nut(angle)` 함수는 100번 문장에서 호출한 상태임.
 - (i) 52번 문장의 `draw_wheel_and_nut(angle)` 함수에서 이상한 부분을 지적하고 수정하라. 이유를 설명할 것.
 - (j) 이 프로그램에서는 사용자가 자동차를 앞뒤로 움직여도 바퀴는 회전을 하지 않는다. 만약 자동차가 앞뒤로 움직일 때 매 프

레임마다 자동차 바퀴를 z 축 둘레로 `angle_z` 각도 만큼 회전을 시키려면 그림 4에서 어느 부분의 변환 행렬을 수정해야 할까?

- (k) 위의 문제에서의 변환을 고려하려면 아래의 프로그램의 어느 부분에 어떤 문장을 추가해야 할지 정확하게 기술하라.
- (l) 83번 문장의 `void cross_prod_vec3(float *u, float *v, float *n)` 함수는 3차원 벡터에 대한 외적, 즉 $n = u \times v$ 을 계산하기 위한 함수이다. 내용을 폐꾸어라.
- (m) 이 프로그램에서 사용하는 인자들을 사용하여 NDC의 좌표 (x_{nd}, y_{nd}) 를 WdC의 좌표 (x_{wd}, y_{wd}) 로 어떻게 변환 시켜주는지 뷔퍼 변환(x 와 y 좌표에 대해서만)을 유도하라.

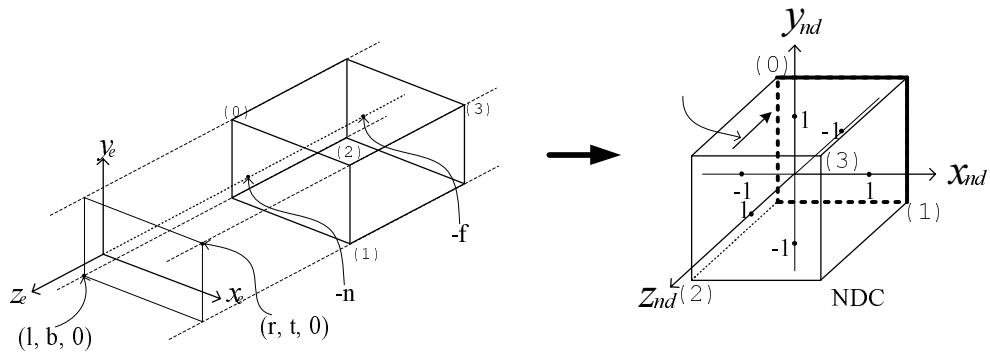


그림 3: glOrtho(l, r, b, t, n, f) 함수

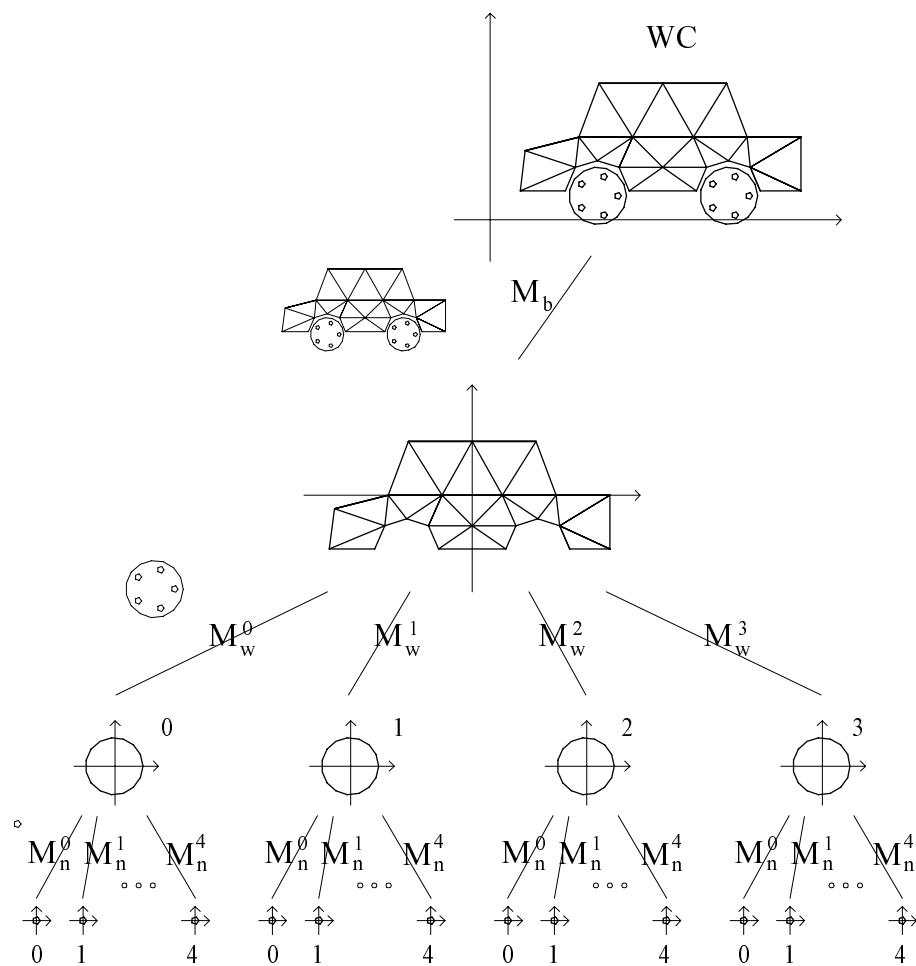


그림 4: 자동차의 계층적 표현

```

1 #include <stdio.h>
2 #include <math.h>
3 #include <GL/glut.h>
4
5 #define MAX_POLY 200
6 #define MAX_VERT 20
7 #define MAX_PATH 1000
8
9 #define DRAW_CAR_DUMMY 2001
10 #define DRAW_CAR_CORRECT 2002
11
12 typedef struct {
13     int invertex;
14     float poly[MAX_VERT][3];
15 } mypolygon;
16
17 mypolygon body[MAX_POLY], wheel[MAX_POLY], nut[MAX_POLY];
18 int npolyb, npolyw, npolyn;
19
20 float path[MAX_PATH][3];
21 int npath, path_exist, drawing_state = DRAW_CAR_CORRECT;
22
23 double dist;
24
25 int prev_i, cur_i = 0;
26 int rightbutbpressed = 0;
27
28 void read_object(char *file, mypolygon *object, int *npoly);
29 void read_path(char *file);
30 void read_objects(void);
31 void draw_axes(void);
32 void draw_path(void);
33 void draw_ground(void);
34
35 void draw_body(void) {
36     // Draw the body.
37     ...
38 }
39
40 void draw_wheel(float angle) {
41     // Draw the wheel.
42     ...
43 }
44
45 void draw_nut(void) {
46     // Draw the nut.
47     ...
48 }
49
50 #define rad 1.7
51 #define vw 1.0
52 void draw_wheel_and_nut(float angle) {
53     int i;
54     draw_wheel(angle); // draw wheel object
55     for (i = 0; i < 5; i++) {
56         // nut i
57         glPushMatrix();
58         glTranslate(rad-0.5, 0, vw); // rad = 1.7, vw = 1.0
59         glRotatef(72.0*i, 0.0, 0.0, 1.0);
60         draw_nut(); // draw nut object
61         glPopMatrix();
62     }
63 }
64
65 #define TO_DEG 57.29579
66 void normalize_vec3(float *v) {
67
68     ...
69 }
70
71 float dot_prod_vec3(float *u, float *v) {
72 }
73
74 float compute_length_mul_two_vec3(float *u, float *v) {
75
76     ...
77 }
78
79 float angle_between_two_vec3(float *u, float *v) {
80     ...
81 }
82
83 void cross_prod_vec3(float *u, float *v, float *n) {
84     // ???
85 }
86
87 float wheel_rot_angle_in_y(void) {
88     ...
89 }
90
91 float wheel_rot_angle_in_z(void) {
92     ...
93 }
94
95 void draw_car_dummy(void) {
96     draw_body(); // draw body object
97     glPushMatrix();
98     glTranslate(-3.9, -3.5, 4.5);
99     draw_wheel_and_nut(0.0); // wheel 0
100    glPopMatrix();
101
102    glPushMatrix();
103    glTranslate(-3.9, -3.5, 4.5);
104    draw_wheel_and_nut(0.0); // wheel 1
105    glPopMatrix();
106
107    glPushMatrix();
108    glTranslate(-3.9, -3.5, -4.5);
109    glScalef(1.0, 1.0, -1.0);
110    draw_wheel_and_nut(0.0); // wheel 2
111
112    glPopMatrix();
113
114    glPushMatrix();
115    glTranslate(3.9, -3.5, -4.5);
116    glScalef(1.0, 1.0, -1.0);
117    draw_wheel_and_nut(0.0); // wheel 3
118
119 }
120
121 void set_up_rot_mat(float *m, int i) {
122
123 }
124
125 void draw_fence(GLfloat r, GLfloat g, GLfloat b) {
126     ...
127 }

```

```

127 }
128 void draw_fences(void) {
129     ...
130 }
131 }
132 void draw_world (void) {
133     GLfloat m[16];
134     glMatrixMode(GL_MODELVIEW); // Modeling Transformation
135     glLoadIdentity();
136     gluLookAt(-15.0, 20.0, 40.0, path[cur_i][0], 4.89, path[cur_i][2], 0.0, 1.0, 0.0);
137
138     draw_ground();
139     draw_fences();
140     draw_axes();
141     if (path_exist) draw_path();
142
143     set_up_rot_mat(m, cur_i);
144
145     glPushMatrix();
146     glTranslatef(path[cur_i][0], 4.89, path[cur_i][2]);
147     glMultMatrixf(m);
148     draw_car_dummy();
149
150     glPopMatrix();
151     draw_car_dummy();
152
153 }
154
155 void render(void) {
156     glClear(GL_COLOR_BUFFER_BIT);
157     draw_world();
158     glutSwapBuffers();
159 }
160
161 void keyboard (unsigned char key, int x, int y) {
162     ...
163 }
164
165 int prevx_mouse;
166 void mousepress(int button, int state, int x, int y) {
167     if ((button == GLUT_RIGHT_BUTTON) && (state == GLUT_DOWN)) {
168         prevx_mouse = x;
169         rightbuttonpressed = 1;
170     }
171     else if ((button == GLUT_RIGHT_BUTTON) && (state == GLUT_UP))
172         rightbuttonpressed = 0;
173 }
174
175 void mousemove(int x, int y) {
176     double deltax;
177
178     if (rightbuttonpressed) {
179         deltax = x - prevx_mouse;
180         prevx_mouse = x;
181         if ((cur_i + deltax > 0) && (cur_i + deltax < npath-1)) {
182             prev_i = cur_i; cur_i += deltax;
183             dist = sqrt((path[cur_i][0]-path[prev_i][0])*(path[cur_i][0]-path[prev_i][0]) +
184             (path[cur_i][1]-path[prev_i][1])*(path[cur_i][1]-path[prev_i][1]) +
185             (path[cur_i][2]-path[prev_i][2])*(path[cur_i][2]-path[prev_i][2]));
186
187     }
188 }
189 }
190 }
191
192 void reshape(int width, int height) {
193     glViewport(0, 0, width, height);
194
195     glMatrixMode(GL_PROJECTION);
196     glLoadIdentity();
197     gluPerspective(30.0, width/ (double) height, 1.0, 150.0);
198 }
199
200 void init_OpenGL(void) {
201     ...
202 }
203
204 void init_windows(void) {
205     glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
206     glutInitWindowSize(1280, 1024);
207     glutCreateWindow("Car in Hierarchy 2");
208     glutDisplayFunc(render);
209     glutKeyboardFunc(keyboard);
210     glutMouseFunc(mousepress);
211     glutMotionFunc(mousemove);
212     glutReshapeFunc(reshape);
213 }
214
215 void main(int argc, char **argv) {
216     read_objects();
217     glutInit(&argc, argv);
218     init_windows();
219     init_OpenGL();
220     glutMainLoop();
221 }
222 }
```

기초 컴퓨터 그래픽스 중간고사 (43-170)

반: _____ 학과: _____ 학번: _____ 이름: _____

1. 다음 단답식 문제에 답하라.

- (a) 좌표가 $(-2.0, 6.0, 10.0, -2.0)$ 인 3차원 투영 공간의 점에 해당하는 아핀 공간의 점의 좌표 (x, y, z) 는?
(답) _____
- (b) 윈도우 서버가 주기적으로 시그널을 보내 주도록 하기 위하여 사용하는 GLUT 컬백 함수의 이름은?
(답) _____
- (c) WM_PAINT 메시지와 가장 밀접한 연관이 있는 GLUT 컬백 함수의 이름은?
(답) _____
- (d) 어떤 4행4열짜리 행렬이 아핀 변환에 대한 변환 행렬이 되려면 어떤 제약 조건을 만족해야 하는가?
(답) _____
- (e) 기본 아핀 변환 중 직교 행렬과 가장 연관이 있는 변환은 무엇인가?
(답) _____
- (f) 2차원 공간에서의 기본 아핀 변환 중 항상 역변환이 존재하는 변환을 모두 기술하라.
(답) _____
- (g) 원근 투영에 대한 계산을 할 경우 바로 이 계산이 이뤄질 때 원근감이 생기게 되는데, 이 계산의 이름은?
(답) _____
- (h) 강체 변환은 아핀 변환 중 어떤 두 변환으로 합성을 할 수가 있는가?
(답) _____
- (i) OpenGL 렌더링 파이프라인에서 각 꼭지점에 대한 색깔 계산은 어떤 좌표계에서 이뤄지는가?
(답) _____
- (j) OpenGL 렌더링 파이프라인에서 일단 NDC로 옮겨온 후 어떤 형태의 투영 변환만 일어나게 되는가? (정확히 기술할 것.)
(답) _____

- (k) 사진 촬영 후 인화지에 사진을 프린트하는 과정과 가장 밀접한 OpenGL 변환의 이름은?

(답) _____

- (l) OpenGL의 뷰잉 파이프라인에서 `glVertex*()` 함수를 사용하여 설정한 꼭지점에 곱해지는 두 번째 행렬 스택의 탑에 있는 변환은 일반적으로 어떤 좌표계에서 어떤 좌표계로의 변환을 위하여 사용하는가?

(답) _____

- (m) OpenGL 뷰잉 파이프라인에서 사용자가 임의로 설정한 절단 평면에 대한 절단 계산이 일어나는 좌표계는 무엇인가?

(답) _____

- (n) 만약 여러분이 `glTranslatef(x, y, z)` 함수를 가장 적은 회수의 덧셈/뺄셈, 곱셈, 그리고 나눗셈 연산을 사용하여 구현한다고 할 때 각각 몇 번씩 필요할까? (여기서 스택에 있는 행렬은 임의의 값을 가질 수 있다고 가정함.)

(답) $+/- =$ 번, $* =$ 번, $/ =$ 번

- (o) OpenGL에서는 기본적으로 볼록 다각형만 지원을 하는데, 볼록 다각형의 수학적인 정의는 무엇인가?

(답) _____

- (p) OpenGL 렌더링 파이프라인 중 초반에는 꼭지점 별 연산(per-vertex operation 또는 per-primitive operation)이 수행이 되다가 렌더링 과정 이후 다른 형태의 연산으로 바뀌게 된다. 그러한 연산을 무엇이라 부르는가?

(답) _____

2. 그림 1에서와 같이 윈도우의 내용을 오른쪽 윈도우로 매핑을 해주는 2차원 변환에 대한 3행 3열 행렬을 기본 아핀 변환의 합성을 통하여 구하라.

(답) 1쪽 뒷면에 기술할 것.

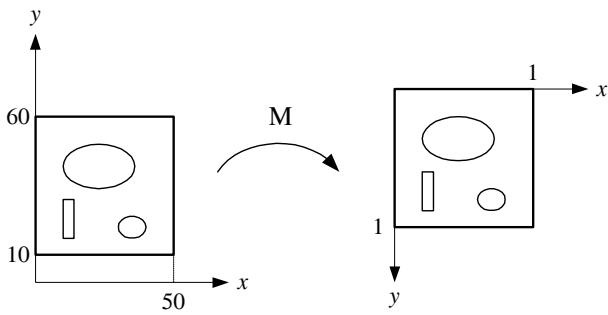


그림 1: 2차원 변환

3. 다음은 뷰잉 변환에 관련된 문제이다.

- (a) 그림 2에는 카메라의 위치와 방향이 설정되어 있다. 이 경우 뷰잉 변환을 설정해주는 코드를 `glTranslatef(x, y, z)` 함수와 `glRotatef(a, x, y, z)` 등의 OpenGL 함수만을 사용하여 가능한한 적은 회수의 함수 호출을 통하여 구현하라.
(답)

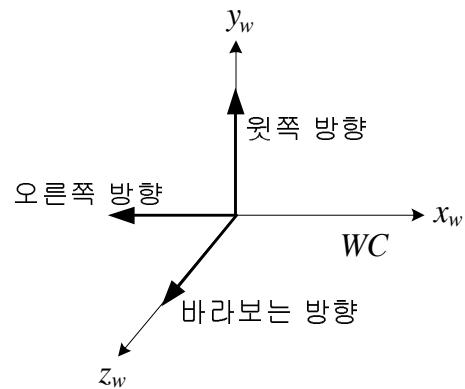


그림 2: 뷰잉 변환

4. 다음은 투영 변환에 관련된 문제이다.

- (b) 마지막 쪽의 그림 4의 프로그램은 `gluLookAt(*)` 함수를 구현한 C 프로그램이다. 뷰잉 변환을 정의하는데 필요한 PRP, VRP, VPN 값을 이 함수의 마지막 문장이 수행되는 시점에서의 지역 변수의 값을 이용하여 표현하라.

(답) $PRP = (\quad, \quad, \quad)$

$VRP = (\quad, \quad, \quad)$

$VPN = (\quad, \quad, \quad)$

- (c) 이 함수의 입력 인자 ax , ay , az 등 9개의 값을 OpenGL 뷰잉 파이프라인 중 어떤 좌표계에서 정의된 값인가?

(답) _____

- (d) 이 프로그램에서 잘못된 부분을 명시하고 옳게 고쳐라.

(답)

(답) 2쪽 뒷면에 기술할 것.

- (b) z_{nd} 의 경우 $z_{nd} = \alpha + \frac{\beta}{z_e}$ 와 같은 형태의 변환을 사용하여 유도한다. 이렇게 비선형적인 변환을 할 경우 발생할 수 있는 바람직하지 못한 결과 한 가지를 들어라.

(답)

5. 다음은 PEXlib이라하는 X 윈도우 환경에서 사용되었던 3차원 그래픽스 라이브러리에 관한 질문이다.

- (a) PEXlib에는 다음과 같은 API 함수가 있다. 이 함수는 OpenGL 관점에서 볼 경우 어떤 좌표계에서 어떤 좌표계로 변환을 해주는 함수일까?

```
int PEXViewOrientationMatrix
( PEXCoord *view_ref_pt,
PEXVector *view_up_vec, PEXVector
*view_plane_normal, PEXMatrix
```

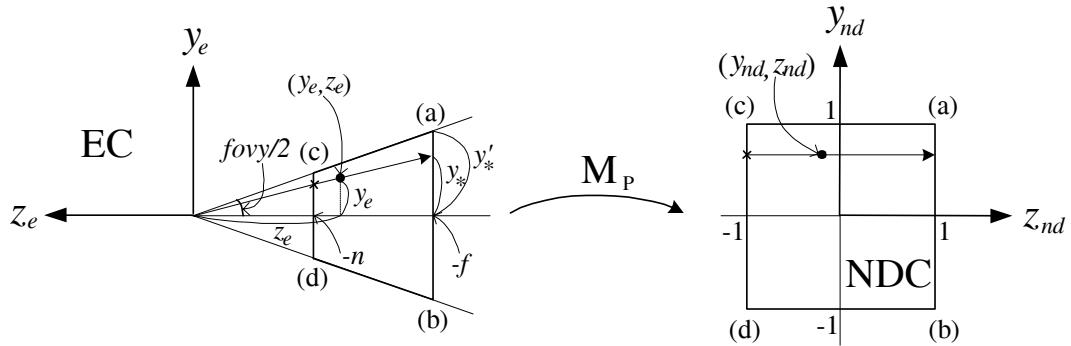


그림 3: 투영 변환

`Orientation_matrix);`

(답) _____

- (b) 다음은 PEXlib에 대한 설명의 일부이다.
At their lowest level, views are defined by two viewing transforms. These are applied to primitives after the local and global (A) transforms have been applied in the (A) stage. The viewing transform first applied orients the model to our viewing position; it is called the view orientation transform. The second viewing transform defines the projection of the scene to 2D; it is called the view mapping transform. The viewing and (A) transforms are applied in the following order: $p_{npc} = V_M \cdot V_O \cdot G \cdot L \cdot p_{mc}$
여기서 (A)에 들어갈 가장 적합한 단어는 무엇인가?

(답) _____

- (c) 위의 $p_{npc} = V_M \cdot V_O \cdot G \cdot L \cdot p_{mc}$ 는 점 p_{mc} 의 p_{npc} 로의 변환을 의미한다. 이 식의 네 개의 변환 행렬 중 OpenGL의 glOrtho(*) 함수와 가장 관련이 있는 변환 행렬은 무엇인가?

(답) _____

- (d) 위 문제의 식에서 카메라의 위치와 방향을 잡는 것에 관련된 변환 행렬은 무엇인가?

(답) _____

- (e) 위에서 p_{mc} 에 대하여 오른쪽에서 두 개, 즉 L 과 G 를 곱했을 때 들어가는 좌표계에 해당하는 OpenGL 좌표계의 이름은 무엇일까?

(답) _____

```
void gluLookAt( GLdouble ax, GLdouble ay, GLdouble az, GLdouble bx, GLdouble by,
    GLdouble bz, GLdouble cx, GLdouble cy, GLdouble cz ) {
    GLdouble m[16]; GLdouble x[3], y[3], z[3]; GLdouble mag;

    z[0] = ax - bx; z[1] = ay - by; z[2] = az - bz;
    mag = sqrt( z[0]*z[0] + z[1]*z[1] + z[2]*z[2] );
    if (mag) { z[0] /= mag; z[1] /= mag; z[2] /= mag; }

    y[0] = cx; y[1] = cy; y[2] = cz;
    x[0] = y[1]*z[2] - y[2]*z[1]; x[1] = -y[0]*z[2] + y[2]*z[0];
    x[2] = y[0]*z[1] - y[1]*z[0];
    y[0] = z[1]*x[2] - z[2]*x[1]; y[1] = -z[0]*x[2] + z[2]*x[0];
    y[2] = z[0]*x[1] - z[1]*x[0];

    mag = sqrt( x[0]*x[0] + x[1]*x[1] + x[2]*x[2] );
    if (mag) { x[0] /= mag; x[1] /= mag; x[2] /= mag; }

    mag = sqrt( y[0]*y[0] + y[1]*y[1] + y[2]*y[2] );
    if (mag) { y[0] /= mag; y[1] /= mag; y[2] /= mag; }

#define M(row,col) m[row*4+col]
    M(0,0) = x[0]; M(0,1) = x[1]; M(0,2) = x[2]; M(0,3) = 0.0;
    M(1,0) = y[0]; M(1,1) = y[1]; M(1,2) = y[2]; M(1,3) = 0.0;
    M(2,0) = z[0]; M(2,1) = z[1]; M(2,2) = z[2]; M(2,3) = 0.0;
    M(3,0) = 0.0; M(3,1) = 0.0; M(3,2) = 0.0; M(3,3) = 1.0;
#undef M
    glMultMatrixd( m );
}

glTranslated( -ax, -ay, -az );
}
```

그림 4: gluLookAt(*) 함수

기초 컴퓨터 그래픽스 Quiz 1 (43-170)

반: _____ 학과: _____ 학번: _____ 이름: _____

1. 다음 물음에 답하라.

- (a) (2점) 동차 좌표로 표현된 3차원 공간의 점 $(-2.0, 0.0, 8.0, -0.5)$ 에 해당하는 3차원 유clidean 공간에서의 점의 좌표는?

(답)

- (b) (2점) RGB 색깔 모델로 표현된 색깔 $(0.7, 0.6, 0.1)$ 을 CMY 색깔 모델을 사용하여 표현했을 때의 값은?

(답)

- (c) (3점) OpenGL에서는 기본적으로 볼록 다각형만 지원하는데, 볼록 다각형의 수학적인 정의는 무엇인가?

(답)

- (d) (3점) PRP (Projection Reference Point)가 무한대 점 (point at infinity)에 위치한 투영 변환에 관련된 OpenGL 함수의 이름은?

(답)

- (e) (5점) OpenGL 함수 `void glScalef(GLfloat x, GLfloat y, GLfloat z);`를 최소한의 부동 소수점 연산을 사용하여 구현한다고 할 때, 몇 번의 부동 소수점 덧셈/뺄셈과 곱셈/나눗셈을 수행해야 할까? 반드시 이유를 밝혀라.

(답)

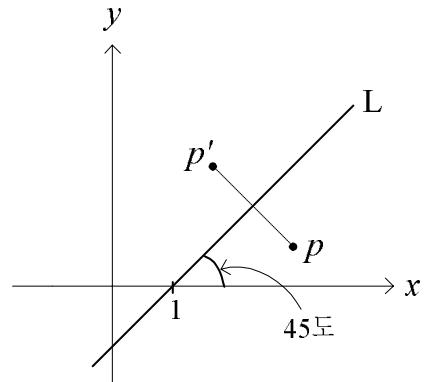


그림 1: 2번 문제

2. (15점) 그림 1에서와 같이 2차원 공간의 점 p 를 직선 L 에 대하여 반사시켜 p' 으로 변환시키는 좌표 변환을 `glTranslatef(x, y, z);`, `glScalef(x, y, z);`, `glRotatef(angle, x, y, z);` 중 적절한 함수들을 사용하여 OpenGL로 구현을 하라. 함수 호출 순서에 주의할 것.

(답)

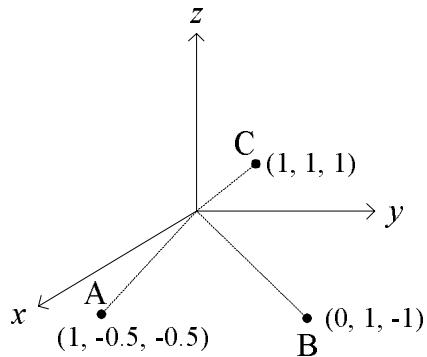


그림 2: 3번 문제

3. 다음 물음에 답하라.

- (a) (2점) 어떤 두 벡터에 대하여 내적 (inner product)를 취했을 때, 그 값이 0이면 이 두 벡터 사이에는 어떠한 성질이 존재하는가?

(답)

- (b) (3점) 3차원 공간에 주어진 세 개의 벡터가 직교(orthogonal)한다면, 이 벡터들 사이에는 어떠한 성질이 존재할까?

(답)

- (c) (3점) 한 3행 3열 행렬이 직교 행렬 (orthogonal matrix)이라면, 그의 역행렬은 어떻게 구할 수 있을까?

(답)

- (d) (20점) 그림 2를 보면 3차원 공간에 세 개의 점이 주어져 있다. 이때 각 점 A, B, C를 각각 x, y, z축에 옮겨주는 아핀 변환에 해당하는 4행 4열 행렬 M 을 구하라. 유도 과정을 반드시 기술할 것.

(답)

- (e) (3점) (d)번을 풀었을 경우 위에서 구한 아핀 변환은 강체 변환(rigid body transformation)인가? 답을 말하고 그에 대한 이유를 기술하라.

(답)

4. 다음의 OpenGL Viewing과 관련한 물음에 답하라. (참고: NDC, MC, EC, WdC, OC, CC, WC)

- (a) (3점) View mapping 연산은 어떤 좌표계에서 어떤 좌표계로의 변환에 해당하는가?

(답)

- (b) (3점) gluLookAt() 함수는 정확하게 말해서 어떤 좌표계에서 어떤 좌표계로의 변환을 위한 것인가?

(답)

- (c) (3점) View volume이 설정이 되는 좌표계는?

(답)

(d) (3점) NDC에서 그 다음 좌표계로 변환을 하기 위한 OpenGL 함수 이름은?
 (답)

(e) (3점) 원근 투영시 비로소 어떤 좌표계까지 왔을 때 원근감이 생성이 되는가?
 (답)

5. (15점) 그림 3에서와 같이 왼쪽 좌표계의 윈도우의 내용을 오른쪽 좌표계의 윈도우 안으로 매핑을 해주는 2차원 기하 변환에 대한 3행 3열 행렬 M 을 구하라. 어떻게 합성을 하는지 분명히 한 후, M 의 내용을 정확하게 계산하라.

(답)

(b) (7점) 문장 11 직후에 $glMultMatrixf(m);$ 과 같은 문장을 삽입하면, 그 결과 화면의 그려지는 내용에 어떠한 변화가 올까?

(답)

(c) (8점) 문장 11의 $gluLookAt();$ 함수를 다음과 같은 두 문장으로 대치를 하려한다.

$glRotatef(*, *, *, *);$
 $glTranslatef(*, *, *);$

이때 동일한 결과를 얻기 위해 이 함수들의 인자로 어떤 값을 사용을 해야 할까?

(답)

$glRotatef(, , ,);$

$glTranslatef(, ,);$

6. 그림 4의 프로그램을 보고 답하라.

(a) (5점) 이 프로그램에서 아핀 변환과 관련이 있는 문장을 모두 나열하라.

(답)

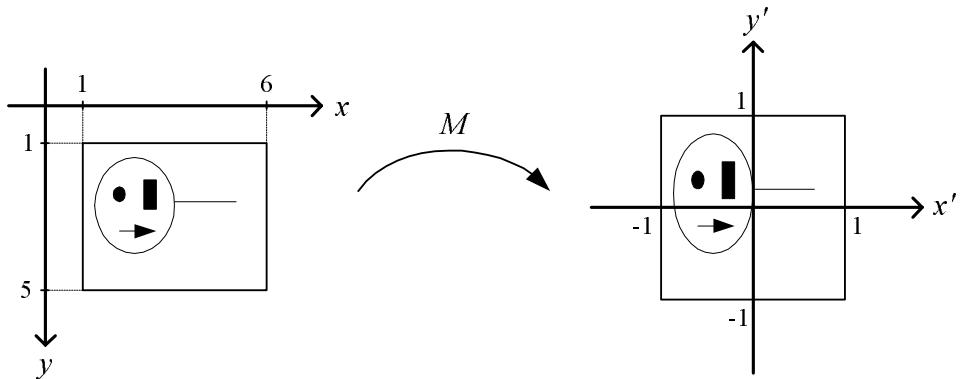


그림 3: 5번 문제

```

void render(void) {
    GLfloat m[16] = { 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0,
                      0.0, 0.0, 1.0, 0.0, 0.0, 0.0, -3.0, 1.0 };

    glClearColor(0.0, 0.0, 0.0, 1.0); // 문장 1
    glClear(GL_COLOR_BUFFER_BIT); // 문장 2
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE); // 문장 3
    glViewport(0.0, 0.0, 800.0, 600.0); // 문장 4
    glMatrixMode(GL_PROJECTION); // 문장 5
    glLoadIdentity(); // 문장 6
    if (ortho)
        glOrtho(-3.6, 3.6, -2.7, 2.7, 5, 19.0); // 문장 7
    else
        gluPerspective(28.0, 4.0/3.0, 5.0, 19.0); // 문장 8
    glMatrixMode(GL_MODELVIEW); // 문장 9
    glLoadIdentity(); // 문장 10
    gluLookAt(-10.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); // 문장 11
    draw_object();
    glFlush(); // 문장 12
}

```

그림 4: 6번 문제