

[CSE4170: 기초 컴퓨터 그래픽스]

기 말 고 사

담당교수: 임 인 성

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

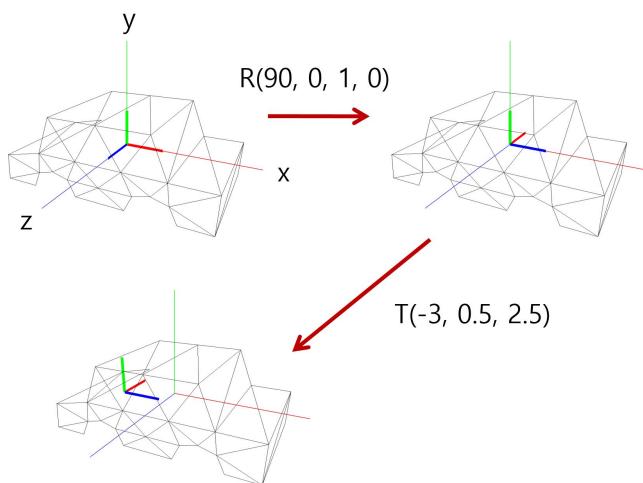


Figure 1: 자동차 운전석으로의 카메라의 배치

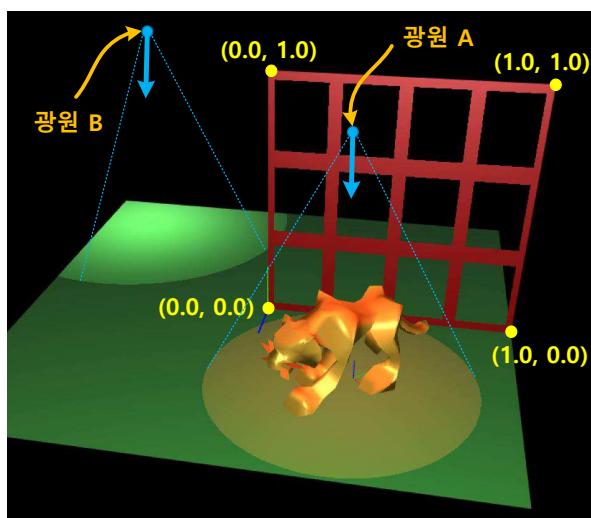


Figure 2: 스크린 효과 생성

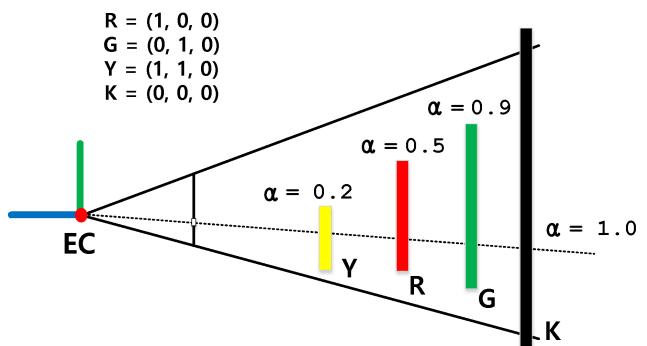


Figure 3: OpenGL을 사용한 색깔 혼합

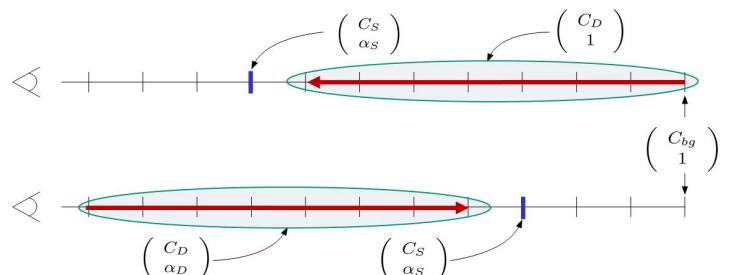


Figure 4: 색깔 합성을 통한 화소 색깔의 계산

1. 다음은 카메라의 배치에 관한 문제이다.

- (a) 카메라의 초기 상태에서 시작하여 카메라 프레임에 대하여 다음과 같은 순서로 아핀 변환을 가하였다면,

$$M_1 \rightarrow M_2 \rightarrow \cdots \rightarrow M_n$$

이에 대한 뷰잉 변환 행렬 M_V 는 다음과 같이 표현할 수 있다.

$$M_V = \{ \quad (A) \quad \}^{-1}$$

이때 (A)에 들어갈 수식을 정확히 기술하라.

- (b) 그림 1에는 자동차 몸체의 모델링 좌표계에서 회전 변환과 이동 변환을 통하여 운전석에 카메라 프레임을 배치하는 모습을 보여주고 있다. 지금 자동차 몸체에 대하여 다음과 같은 순서대로 아핀 변환을 가하여 세상 좌표계에 배치하려 한다.

$$R(90, 0, 1, 0) \rightarrow T(20, 4, 0) \rightarrow R(15, 0, 1, 0)$$

이 경우 자동차 운전석에 배치한 카메라도 같이 움직이게 되는데, 이에 대한 뷰잉 변환 행렬 M_V 를 $T(x, y, z)$ 와 $R(\alpha, x, y, z)$ 행렬들을 사용하여 표현하라. (주의: M_V 에 대한 식을 기술할 때 역행렬 기호를 사용하지 말 것)

- (c) 다음은 위에서 기술한 운전석에 배치한 카메라에 대한 뷰잉 변환 행렬 ViewMat을 설정해주는 어떤 OpenGL 코드이다.

```
void set_ViewMatrix_for_driver(void) {
    glm::mat4 Mat_CAMERA_driver_inv;

    Mat_CAMERA_driver_inv
        = ModelMat_CAR_BODY
        * ModelMat_CAR_BODY_to_DRIVER;

    ViewMat = glm::affineInverse(
        Mat_CAMERA_driver_inv);
    ViewProjectionMat
        = ProjectionMat * ViewMat;
}
```

이 함수가 호출되기 직전에 전역 변수인 4행 4열 행렬 ModelMat_CAR_BODY와 ModelMat_CAR_BODY_to_DRIVER 각각에는 어떤 내용의 아핀 변환이 계산되어 저장되어야 하는지, 그 내용을 $T(x, y, z)$ 와 $R(\alpha, x, y, z)$ 행렬들을 사용하여 표현하라. (주의: 각 행렬에 대한 식을 기술할 때 역행렬 기호를 사용하지 말 것)

2. 다음은 스크린 효과 생성을 위한 OpenGL 프래그먼트 쉐이더 코드의 일부이다.

```
=====
:
in vec2 v_pos_sc;
```

```
void main(void) {
    if (screen_effect) {
        float x_mod, y_mod;
        x_mod = mod(v_pos_sc.x, 1.0f);
        y_mod = mod(v_pos_sc.y, 1.0f);
        if ((x_mod > 0.1f) || (x_mod < 0.9f))
```

```
        || (y_mod > 0.1f) || (y_mod < 0.9f) )
            discard;
    }
    final_color = l_e(v_pos_EC,
                      normalize(v_norm_EC));
}
```

- (a) 그림 2의 스크린 사각형의 각 꼭지점에는 $v_pos_sc = (v_pos_sc.x, v_pos_sc.y)$ 속성으로 설정한 값이 주어져 있다. 이 쉐이더 코드에는 심각한 오류들이 있는데 이를 OpenGL Shading Language 문법에 맞게 정확히 수정하라. 즉 어느 부분을 어떻게 수정해야 할지 명확히 기술할 것. (참고: $\text{mod}(x, y)$ returns $x - y \cdot \text{floor}(\frac{x}{y})$.)

- (b) (위 문제를 정확히 해결하였다는 가정 하에) 만약 이 쉐이더 코드에서 첫 번째 $\text{mod}()$ 함수(즉 $x_mod = \text{mod}(\dots)$ 문장의)를 $\text{min}()$ 함수로 대체할 경우 스크린의 모습이 어떠할지 정확한 거리 척도를 기술하면서 그림으로 표현하라. (참고: $\text{min}(x, y)$ returns the lesser of two values.)

3. 역시 그림 2를 보면서 광원의 설정에 관한 문제에 답하라. 본 문제는 프래그먼트 쉐이더가 눈좌표계(Eye Coordinate)를 기준으로 라이팅 계산을 수행한다고 가정한다.

- (a) ‘광원 A’는 2번 광원으로서 호랑이 모델의 모델링 좌표계를 기준으로 기하 정보를 기술하고 있는데, 이 모델링 좌표계의 z_m 축의 정반대 방향을 중심으로 스포트 광원을 설정하려 한다. (아래 코드의 변수의 의미는 분명함.)

```
light[2].position[0] = 0.0f;
light[2].position[1] = -15.0f;
light[2].position[2] = 180.0f;
light[2].position[3] = 1.0f;
light[2].spot_direction[0] = (A);
light[2].spot_direction[1] = (B);
light[2].spot_direction[2] = (C);
```

이때 위의 (A), (B), 그리고 (C)에 들어갈 값을 기술하라.

- (b) 그림 5에는 디스플레이 컬백 함수에서 $\text{draw_tiger}()$ 함수를 통하여 호랑이 기하 모델을 그려주는 부분에 대한 코드가 주어져 있는데, 문맥상 각 변수들의 의미는 분

```

void display(void) {
    :
    glUseProgram(h_ShaderProgram_PS);

    ModelViewMatrix = glm::rotate(ViewMatrix, -rotation_angle_tiger, glm::vec3(0.0f, 1.0f, 0.0f));
    ModelViewMatrix = glm::translate(ModelViewMatrix, glm::vec3(150.0f, 0.0f, 0.0f));
    ModelViewMatrix = glm::rotate(ModelViewMatrix, -90.0f*TO_RADIAN, glm::vec3(1.0f, 0.0f, 0.0f));
    ModelViewMatrixInvTrans = glm::inverseTranspose(glm::mat3(ModelViewMatrix)); // Line (D)

    ModelViewProjectionMatrix = ProjectionMatrix * ModelViewMatrix;

    glUniformMatrix4fv(loc_ModelViewProjectionMatrix_PS, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
    glUniformMatrix4fv(loc_ModelViewMatrix_PS, 1, GL_FALSE, &ModelViewMatrix[0][0]);
    glUniformMatrix3fv(loc_ModelViewMatrixInvTrans_PS, 1, GL_FALSE, &ModelViewMatrixInvTrans[0][0]);

    glm::vec4 vector_1 = (E);
    glUniform4fv(loc_light[2].position, 1, &vector_1[0]);

    glm::vec3 vector_2 = normalize( (F) );
    glUniform3fv(loc_light[2].spot_direction, 1, &vector_2);

    set_material_tiger();
    draw_tiger();
    :
}

```

Figure 5: 모델링 좌표계 상에서의 광원의 설정

명하다. 이 코드의 (E) 부분에서는 `vec4` 타입의 벡터를 얻기 위하여 어떠한 계산을 해야하는지 이 코드의 변수들을 언급하면서 명확하고 구체적으로 기술하라.

- (c) (바로 위 문제에 이어서) 문맥상 이 코드의 (F) 부분에서는 `vec3` 타입의 벡터를 얻기 위하여 어떠한 계산을 해야하는지 이 코드의 변수들을 언급하면서 명확하고 구체적으로 기술하라.

- (d) (바로 위 문제에 이어서) 만약 Line (D)의 내용을 다음과 같이 변경할 경우 렌더링 결과에 어떤 변화가 있을까? ‘예/아니오’로 답하고 자신의 이유를 명확하고 구체적으로 기술하라.

```

ModelViewMatrixInvTrans =
    glm::mat3(ModelViewMatrix);

```

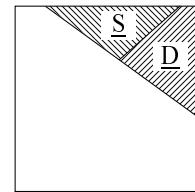
- (e) 그림 2에서 ‘광원 B’는 세상 좌표계를 기준으로 기하 정보가 설정되어 있다. 이 광원의 경우 어떠한 상황이 발생하였을 때마다 매번 광원의 기하 정보를 쉐이더의 uniform 변수들을 통하여 갱신해주어야 할까?

4. 다음은 색깔 혼합에 관한 문제이다. $(c_S \alpha_S)$ 와 $(c_D \alpha_D)$ 를 두 이미지 S와 D의 대응되는 화소

의 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) 아래 그림이 암시하는 합성 연산의 경우 해당하는 F_S 와 F_D 의 값은 각각 얼마인가?



- (b) 만약 S over D 연산을 적용한다면, 합성 후 결과 화소 O 에 칠할 실제 색깔(미리 곱한 색깔이 아니라) C_O 를 두 화소 S와 D의 실제 색깔 C_S 와 C_D 와 불투명도 α_S 와 α_D 값을 사용하여 정확히 기술하라.
- (c) 어떤 미리 곱한 색깔이 $(0.4, 0.4, 0.4, 0.8)$ 이라면, 이는 그 화소를 어떻게 칠해야 한다는 것인지 정확히 기술하라.
- (d) 그림 4는 주어진 화소에 대하여 여러 지점에서 샘플링한 색깔들을 각각 어떤 특정 순서로 합성을 하면서 합성을 하는 두 개의 과정을

보여주고 있다. 위쪽 방법과 아래쪽 방법 각각의 장점 한 가지씩 기술하라.

- (e) (바로 위 문제에 이어서) 위쪽에는 샘플링한 색깔들을 뒤에서 앞으로 가면서 합성하는 모습을 보여주고 있다. (가장 뒤쪽에서 샘플링한 배경 색깔의 불투명도는 1이라 가정함) 지금까지 합성한 결과의 실제 색깔이 C_D 이고 현재 합성하려는 지점의 실제 색깔과 불투명도가 각각 C_S 와 α_S 라 하자. 이때 합성 후의 미리 곱한 색깔 c_O 와 불투명도 α_O 를 다음과 같이 계산하려 할 때 x 와 y 에 들어갈 수식을 정확히 기술하라.

$$\begin{pmatrix} c_O \\ \alpha_O \end{pmatrix} = x \cdot \begin{pmatrix} C_S \\ 1 \end{pmatrix} + y \cdot \begin{pmatrix} C_D \\ 1 \end{pmatrix}$$

- (f) 그림 3의 상황을 고려하자. 영상 평면의 한 화소에 네 개의 물체 Y, R, G, 그리고 K가 순서대로 투영되고 있는데, 이 물체들은 각각 순서대로 ‘순수한’ 노란색, 빨간색, 초록색, 그리고 검정색을 띠고 있으며, 각 물체의 불투명도는 각 물체 옆에 기술되어 있다. 지금 이 물체들의 색깔을 앞에서 뒤로 가면서 (front-to-back order) **over** 연산을 사용하여 합성한다고 할 때, Y부터 G 물체까지 합성하였을 때의 불투명도 값이 무엇일지 기술하라.

- (g) (바로 위 문제에 이어서) 이 물체들의 색깔을 뒤에서 앞으로 가면서 (back-to-front order) **over** 연산을 사용하여 합성한다고 할 때, K부터 R 물체까지 합성하였을 때의 색깔이 무엇인지 불투명도도 포함한 미리곱한 색깔 형식으로 기술하라.

- (h) 다음은 투명한 물체들을 그려주는 어떤 프로그램의 프래그먼트 쉐이더 코드이다. 여기서 유니폼 변수 $u_flag_blending$ 이 0, 1, 또는 2 값을 가질 경우, 각각 혼합(blending) 기능을 사용하지 않거나(0), 뒤에서 앞으로 가면서 **over** 연산을 사용하여 합성을 하거나(1), 또는 앞에서 뒤로 가면서 **over** 연산을 사용하여 합성을 해주게 된다(2). 또한, 함수 $obj_C(*,*)$ 는 현재 프래그먼트에 칠할 실제 색깔(미리 곱한 색깔이 아닌)을 계산해주며, 불투명도 값은 프래그먼트 속성 변수 $frag_a$ 로 들어온다. 또한 $u.flag.blending$ 값을 2로 설정할 때 다음과 같이 합성 인자를 설정하였다.

```
glBlendFunc(GL_ONE_MINUS_DST_ALPHA,
            GL_ONE);
```

이때 올바른 색깔 합성 결과를 얻기 위하여 아래의 쉐이더 코드의 (A)에 들어갈 내용을 OpenGL Shading Language 문법에 맞게 정확히 기술하라.

```
#version 420
:
uniform int u_flag_blending = 0;
in vec3 v_position_EC;
in vec3 v_normal_EC;
in float frag_a;
layout (location = 0) out vec4 f_c;

vec3 obj_Color(in vec3 P_EC, in vec3 N_EC) {
    vec3 object_color;
    : // Compute the fragment's RGBA color.
    return object_color;
}

void main(void) {
    vec3 Color = obj_Color(v_position_EC,
                           normalize(v_normal_EC));
    if (u_flag_blending == 0)
        f_c = vec4(Color, 1.0f);
    else
        f_c = vec4( _____ (A) _____ , frag_a);
}
```

- (i) (바로 위 문제를 올바르게 풀었다는 가정하에) 만약 $u.flag.blending$ 값을 1로 설정할 경우, 이 프로그램이 제대로 작동하기 위하여 $glBlendFunc(*,*)$ 함수 호출에 필요한 두 인자를 아래에서 찾아 순서대로 기술하라.

```
GL_ZERO, GL_ONE,
GL_SRC_ALPHA, GL_DST_ALPHA,
GL_ONE_MINUS_SRC_ALPHA,
GL_ONE_MINUS_DST_ALPHA
```

5. 그림 6(a)에는 풍의 조명 모델 식의 두 예가 기술되어 있고, (b)-(d)에는 OpenGL 시스템에서 사용하는 기본 조명 공식이 주어져 있다.

- 문맥상 (a)의 공식에서 잘못된 곳이 있으면 모두 찾아 바르게 고쳐라(없을 경우 ‘없음’이라고 답할 것).
- (a)의 공식에서 H_i 를 이 공식의 다른 기호를 사용하여 정확히 표현하라.
- (a)의 공식에서 Lambert's cosine law를 표현해주는 부분을 정확히 기술하라.
- (a)의 공식에서 만약 점 광원을 사용할 경우 광원의 위치의 변화에 영향을 받는 변수를 모두 나열하라.

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (R \circ V)^n\} \quad (1)$$

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (N \circ H_i)^n\} \quad (2)$$

(a) 풍의 조명 모델 식의 두 예

$$\mathbf{c} = \mathbf{e}_{cm} + \mathbf{a}_{cm} * \mathbf{a}_{cs} + \sum_{i=0}^{n-1} (att_i)(spot_i)[\mathbf{a}_{cm} * \mathbf{a}_{cli} + (\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli}) \mathbf{d}_{cm} * \mathbf{d}_{cli} + (f_i)(\mathbf{n} \odot \hat{\mathbf{h}}_i)^{s_{rm}} \mathbf{s}_{cm} * \mathbf{s}_{cli}]$$

(b) OpenGL 시스템의 조명 공식

$$att_i = \begin{cases} \frac{1}{k_{0i} + k_{1i}\|\overrightarrow{\mathbf{VP}}_{pli}\| + k_{2i}\|\overrightarrow{\mathbf{VP}}_{pli}\|^2}, & \mathbf{P}_{pli}'s w \neq 0, \\ 1.0, & \text{otherwise} \end{cases}$$

(c) 빛의 감쇠 효과의 계산

$$spot_i = \begin{cases} (\overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{dli})^{s_{rli}}, & c_{rli} \neq 180.0 \& \overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{dli} < \cos c_{rli}, \\ 0.0, & c_{rli} \neq 180.0 \& \overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{dli} \geq \cos c_{rli}, \\ 1.0, & c_{rli} = 180.0 \end{cases}$$

(d) 스폷 광원 효과의 계산

Figure 6: 조명 모델 공식

- (e) (a)의 공식의 d_i 에 해당하는 값을 (b)의 공식의 변수들을 사용하여 기술하라.
- (f) 정반사 방향 (reflection direction), 즉 정반사 물질이 입사 광선을 가장 강하게 반사시키는 방향을 (b) 공식의 변수들을 사용하여 표현하라. ($\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli}$ 은 양수라고 가정함)
- (g) 아래의 식은 \mathbf{h}_i 를 계산하는 과정을 기술하고 있다.

$$\mathbf{h}_i = \begin{cases} \overrightarrow{\mathbf{VP}}_{pli} + \overrightarrow{\mathbf{VP}}_e, & v_{bs} = \text{TRUE}, \\ \overrightarrow{\mathbf{VP}}_{pli} + (0 \ 0 \ 1 \ 0)^t, & v_{bs} = \text{FALSE} \end{cases}$$

여기서 점 \mathbf{P}_e 의 동차 좌표를 기술하라.

- (h) (c)의 공식의 조건식 $\mathbf{P}_{pli}'s w \neq 0$ 이 만족되는 상황은 어떤 경우인지 정확히 기술하라.
- (i) (c)의 공식을 구현한 아래의 OpenGL 셰이더 코드를 보자.

```
=====
if (u_light_att_factors.w != 0.0f) {
    vec4 tmp_vec4;
```

```
tmp_vec4.x = _____ (A) _____;
tmp_vec4.z = dot(L_EC, L_EC);
tmp_vec4.y = sqrt(_____ (B) _____);
tmp_vec4.w = 0.0f;
att_factor = 1.0f / dot(tmp_vec4,
                        u_light_att_factors);
}
else {
    att_factor = 1.0f;
}
=====
```

vec4 타입의 변수 `u_light_att_factors`의 `x`, `y`, 그리고 `z` 필드에는 각각 k_{0i} , k_{1i} , 그리고 k_{2i} 가 저장되어 있다. 또한 (c) 공식에서와는 약간 다르게 네 번째 `w` 필드 값이 0이 아닐 때 빛의 감쇠 효과를 계산하려하고 있다. 이때 (A)와 (B)에 들어갈 내용을 OpenGL Shading Language 문법에 맞게 기술하라.

- (j) (d)의 공식은 스폷 광원으로 인한 입사 광선의 밝기를 계산하기 위한 식이다. 여기서 두 벡터 $\overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}}$ 와 $\hat{\mathbf{s}}_{dli}$ 가 의미하는 방향을 정확히

- 기술하라.
- (k) (d)의 공식에서 잘못된 것을 모두 찾아 바로 잡아라.
6. 문제지 뒤쪽의 그림 10에는 스포트 조명을 지원하는 풍 쇼이딩 기반의 버텍스 및 프래그먼트 쇼이더 코드가 주어져 있는데 각 변수들의 의미는 분명하다.
- (a) 다음은 Direct3D의 쇼이더인 HLSL Shader에 대한 설명의 일부이다.
- As a minimum, this shader must output this information in homogeneous clip space.
- 이 쇼이더 코드에서 this information과 가장 관련이 있는 변수 이름을 기술하라.
- (b) 이 쇼이더 코드에서 정반사로 인한 하이라이트 영역의 크기를 직접적으로 조절할 수 있는 변수를 기술하라.
- (c) (A)에 들어갈 내용을 OpenGL Shading Language 문법에 맞게 정확히 기술하라.
- (d) 문맥상 (B)와 (C)에 들어갈 내용을 OpenGL Shading Language 문법에 맞게 정확히 기술하라.
- (e) 문맥상 (D)에 들어갈 내용을 OpenGL Shading Language 문법에 맞게 정확히 기술하라.
- (f) Line (a)에서 계산해주는 변수 s 를 고려하자. 그림 6(a)에서 기술한 라이팅 모델 수식에서 이 변수와 가장 관련이 높은 기호 또는 수식을 찾아 정확히 기술하라.
- (g) (바로 위 문제에 이어서) (E)에 들어갈 내용을 OpenGL Shading Language 문법에 맞게 정확히 기술하라.
- (h) (F)에 들어갈 내용을 OpenGL Shading Language 문법에 맞게 정확히 기술하라.
- (i) 그림 6(b)의 수식에서 $\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli}$ 값을 이 프로그램의 변수 또는 수식을 사용하여 표현하라.
- (j) 이 쇼이더 프로그램에는 매우 심각한 버그가 존재한다. 어떤 문장에 버그가 있는지를 밝히고 OpenGL Shading Language 문법에 맞게 올바르게 고쳐라.
- (k) (이 프로그램의 버그를 수정하였다는 가정하에) Line (b)에서 $\max(\dot{\mathbf{h}}, \mathbf{n}, 0.0)$ 부분을 $\mathbf{vec3 r};$ 과 같이 정의되는 \mathbf{r} 벡터를 사용하여 $\max(\dot{\mathbf{r}}, \mathbf{v}, 0.0)$ 로 대치하려 한다. 이때 \mathbf{r} 벡터를 어떻게 계산을 할 지 이 쇼이더 코드의 변수를 사용하여 OpenGL Shading Language 문법에 맞게 정확히 기술하라.

7. 그림 11의 OpenGL ES rendering pipeline을 보고 답하라. 이 문제의 질문은 모두 ‘정상적인 렌더링 과정’을 가정한다. 상자를 치칭할 때는 ‘(A)’와 같이 해당 기호를 사용하여 기술하고, 필요할 경우 NDC, CC, WdC, MC, EC, 그리고 WC 등의 기호를 사용하여 답할 것.
- (a) 다음 그림이 암시하는 계산과 가장 직접적인 관련이 있는 상자의 기호는?
-
- $$P_1(x, y) = z(s, t)$$
- (b) (A) 박스는 OpenGL의 어느 좌표계에서 어느 좌표계까지의 기하 계산을 담당하는가?
- (c) 물체가 투영되는 픽셀에 칠해질 최종 색깔을 계산해주는 상자의 기호는?
- (d) 카메라 배치와 관련한 본 시험의 1번 문제와 가장 관련이 높은 상자의 기호는?
- (e) 본 시험의 4번 문제의 색깔 혼합 계산이 실제로 수행이 되는 상자의 기호는?
- (f) `glFrontFace(GL_CCW);` 문장이 가장 직접적으로 영향을 미치는 상자의 기호는?
- (g) 동차 좌표계의 네 번째 좌표 값 W 가 1인 아닌 상태에서 다시 1로 돌아오게 해주는 연산에 해당하는 박스는?
- (h) 다면체 쇼이딩 모델 중의 하나인 고우러드 쇼이딩 기법을 구현할 때, 실제 풍의 조명 모델 공식이 적용되는 상자의 기호는?
- (i) `glDrawArrays(GL_TRIANGLES, 0, 333);` 문장에서 상수 `GL_TRIANGLES`이 직접적으로 영향을 미치는 상자의 기호는?
- (j) 어느 두 상자 사이에서 꼭지점의 좌표값이 항상 -1과 1사이의 값만 가질지 그 두 상자 기호를 기술하라.
- (k) `glm::perspective()` 함수가 설정한 3차원 영역을 벗어난 기하 프리미티브들을 제거하는 계산이 수행되는 박스의 기호를 기술하라.
8. 다음과 같은 4행 4열 행렬 M 에 의해 정의되는 3차원 아핀 변환을 고려하자.

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

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

Figure 7: 회전 변환 행렬

- (a) 그림 8의 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.

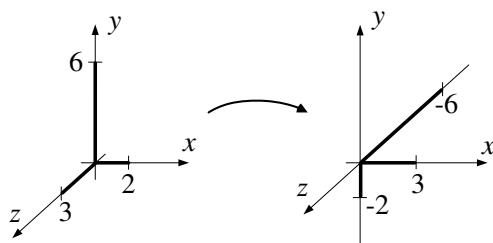
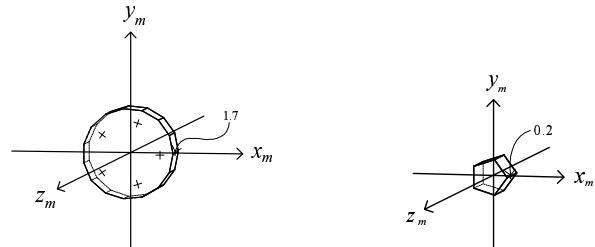


Figure 8: 3차원 공간에서의 회전 변환

(H), (I), 그리고 (J)에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라.



- (b) 바로 위 문제에서의 회전 변환은 $(n_x, 1, n_z)$ 벡터가 가리키는 직선 둘레로 α 도만큼 회전 시켜주는 변환에 해당한다. 과연 α 가 몇 도 인지 정확히 기술하라. (참고: 그림 7에 주어진 회전 변환 행렬을 참조하고, 각도는 0도와 180도 사이의 각으로 기술할 것)
9. 10쪽의 코드는 계층적 모델링 변환을 통하여 자동차를 세상 좌표계로 그려주는 OpenGL 프로그램이다. 이 코드와 그림 9을 보면서 답하라.

- (a) 프로그램 문맥상 62번 문장의 (A)에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라.
- (b) 프로그램 문맥상 62번 문장의 (B), (C), 그리고 (D)에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라. (힌트: 그림 9의 내용을 잘 살펴볼 것)
- (c) 이 프로그램의 26번 문장은 자동차가 움직임에 따라 두 앞 바퀴가 좌우로 회전하는 모습을 생성해주는 기하 변환을 계산하고 있다(`draw_car()` 함수에서 적용이 됨). 문맥상 (E), (F), 그리고 (G)에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라.
- (d) 이 프로그램의 29번 문장은 자동차가 움직임에 따라 네 바퀴가 진행 방향으로 회전하는 모습을 생성해주는 기하 변환을 계산하고 있다(`draw_car()` 함수에서 적용이 됨). 문맥상

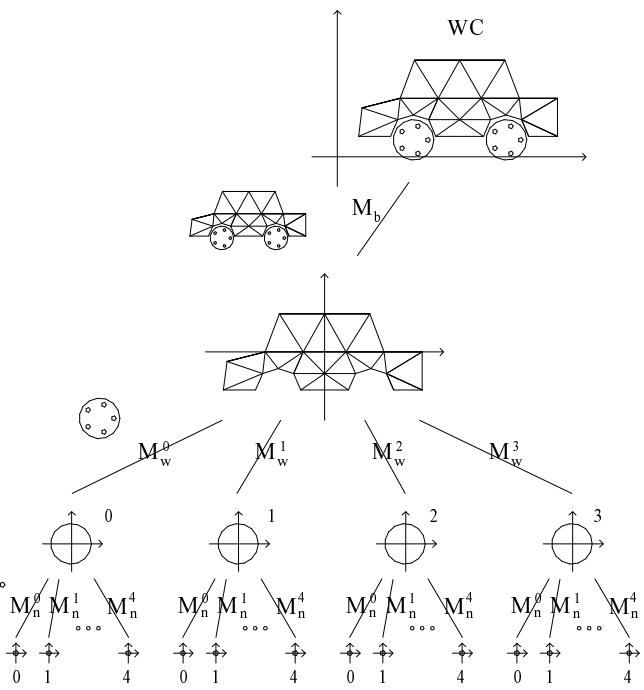


Figure 9: 자동차의 계층적 표현

- (e) 변수 `ModelMatrix_CAR_WHEEL`에 94번 문장 수행 후 저장되는 4행 4열의 내용을 그림 9의 행렬 기호를 사용하여 표현하라.
- (f) 57번 문장의 `draw_wheel_and_nut()` 함수가 92번 문장에서 호출되어 수행이 되는 과정에서, 변수 i 가 2일 때 63번 문장 수행 결과 변수 `ModelMatrix_CAR_NUT`에 저장되는 행렬의 내용을 그림 9의 행렬 기호를 사용하여 표현하라. 이 순간에 행렬 `car_status.Matrix_CAR_WHEEL_ROTATE`의 내용은 M_R 이라고 가정함.

```

#version 410

layout (location = 0) in vec3 VertexPosition;
layout (location = 1) in vec3 VertexNormal;
out vec3 Position; out vec3 Normal;

uniform mat4 ModelViewMatrix; uniform mat3 NormalMatrix;
uniform mat4 ProjectionMatrix; uniform mat4 MVP;

void main() {
    Normal = normalize(NormalMatrix*VertexNormal);
    Position = vec3( (A) );
    gl_Position = MVP * vec4(VertexPosition,1.0);
}
//********************************************************************/
#version 410

in vec3 (B);
in vec3 (C);

struct SpotLightInfo {
    vec4 position; // Position in eye coords
    vec3 intensity;
    vec3 direction; // Direction of the spotlight in eye coords
    float exponent; // Angular attenuation exponent
    float cutoff; // Cutoff angle (between 0 and 90)
};
uniform SpotLightInfo Spot;
uniform vec3 Kd; // Diffuse reflectivity
uniform vec3 Ka; // Ambient reflectivity
uniform vec3 Ks; // Specular reflectivity
uniform float Shininess; // Specular shininess factor

layout( location = 0 ) out vec4 FragColor;

vec3 adsWithSpotlight( ) {
    vec3 n = normalize(Normal);
    vec3 s = normalize(vec3(Spot.position) + (E)); // Line (a)
    vec3 spotDir = normalize(Spot.direction);
    float angle = acos(dot(-s, spotDir));
    float cutoff = radians(clamp(Spot.cutoff, 0.0, 90.0));
    vec3 ambient = Spot.intensity * Ka;

    if ( (F) ) {
        float spotFactor = pow(dot(-s, spotDir), Spot.exponent);
        vec3 v = vec3(-Position);
        vec3 h = normalize(v + s);
        return ambient + spotFactor*Spot.intensity*(Kd*max(dot(s, n), 0.0)
            + Ks*pow(max(dot(h, n), 0.0), Shininess)); // Line (b)
    }
    else {
        return ambient;
    }
}

void main() {
    (D) = vec4(adsWithSpotlight(), 1.0);
}

```

Figure 10: 스폿 광원을 지원하는 버텍스/프래그먼트 쉐이더 코드

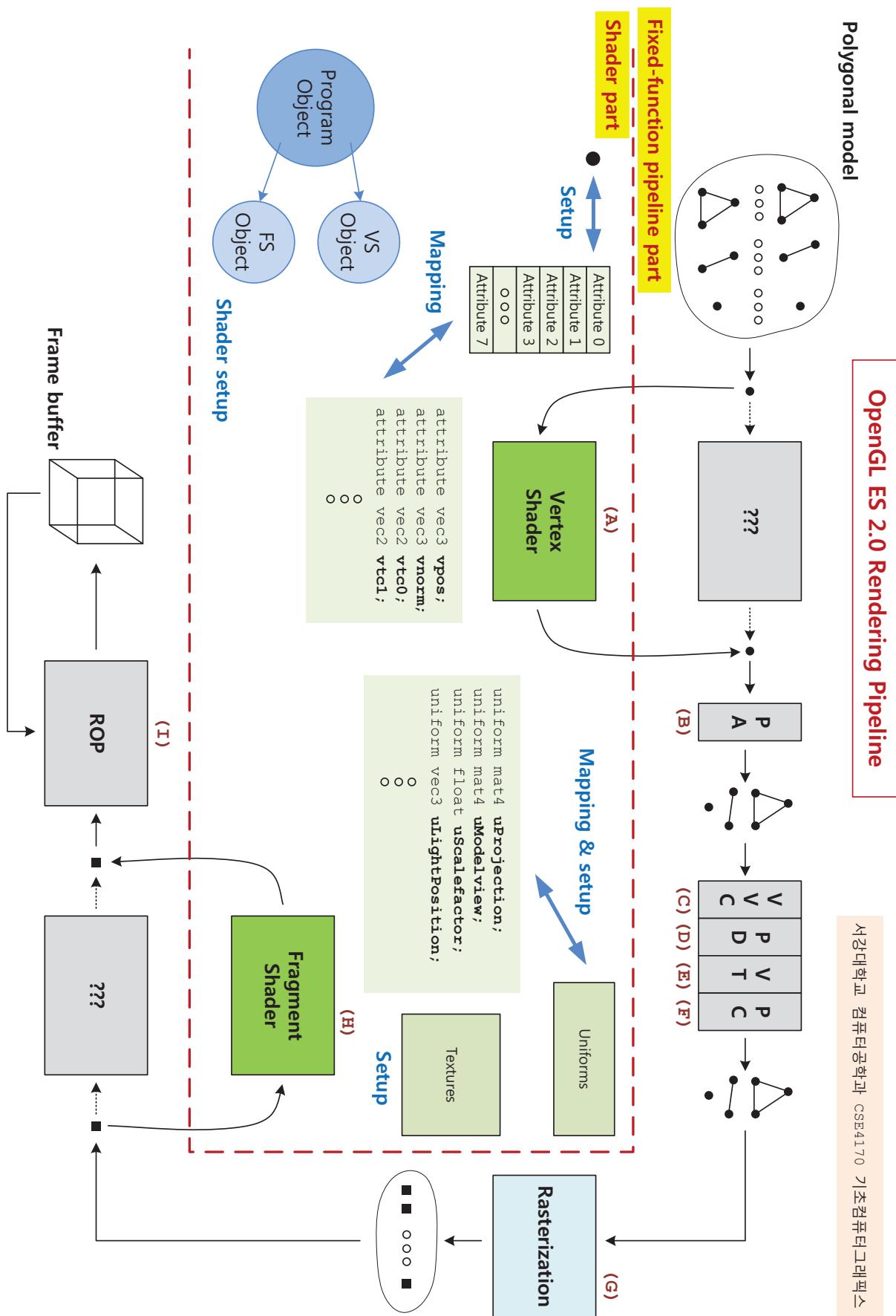


Figure 11: OpenGL ES 렌더링 파이프라인

```

1 #define WHEEL_NUT_RADIUS (1.7f - 0.5f)
2 #define WHEEL_NUT_Z_OFFSET 1.0f
3
4 #define FRONT_WHEEL_ROLL_ANGLE_DELTA 5.0f
5 #define FRONT_WHEEL_TURN_ANGLE_DELTA 2.5f
6 #define FRONT_WHEEL_TURN_ANGLE_MAX 45.0f
7 #define BODY_YAW_SENSITIVITY 0.1f
8
9 struct _CAR_STATUS {
10     float body_yaw_angle;
11     float front_wheel_roll_angle;
12     float front_wheel_turn_angle;
13     glm::mat4 Matrix_Car_Wheel_Rotate;
14     int flag_body_yaw_mode;
15 } car_status;
16
17 void update_car_body_transformation(void) {
18     ModelMatrix_Car_Body = glm::rotate(glm::mat4(1.0f),
19                                         TO_RADIAN*car_status.body_yaw_angle, glm::vec3(0.0f, 1.0f, 0.0f));
20
21     ModelMatrix_Car_BODY_INVERSE = glm::rotate(glm::mat4(1.0f),
22                                              TO_RADIAN*car_status.body_yaw_angle, glm::vec3(0.0f, 1.0f, 0.0f));
23 }
24
25 void update_car_front_wheel_rotate_matrix(void) {
26     car_status.Matrix_Car_WHEEL_ROTATE = glm::rotate(glm::mat4(1.0f),
27                                                       TO_RADIAN*car_status.front_wheel_turn_angle, glm::vec3((E), (F), (G)));
28
29     car_status.Matrix_Car_WHEEL_ROTATE = glm::rotate(car_status.Matrix_Car_WHEEL_ROTATE,
30                                                       TO_RADIAN*car_status.front_wheel_roll_angle, glm::vec3((H), (I), (J)));
31 }
32
33 void set_ViewMatrix_for_wor_id_viewer(void) {
34     ViewMatrix = glm::mat4(camera_a.wv.xaxis.x, camera_a.wv.xaxis.y, camera_a.wv.xaxis.z, 0.0f,
35                             camera_a.wv.xaxis.y, camera_a.wv.xaxis.y, 0.0f, camera_a.wv.xaxis.y, 0.0f,
36                             camera_a.wv.xaxis.z, camera_a.wv.xaxis.z, camera_a.wv.xaxis.z, 0.0f,
37                             0.0f, 0.0f, 1.0f);
38
39     ViewMatrix = glm::translate(ViewMatrix, -camera_a.wv.pos);
40
41 void set_ViewMatrix_for_driver(void) {
42     ViewMatrix = glm::affineInverse((K) * ModelMatrix_Car_BODY_to_DRIVER);
43 }
44
45 void initialize_car(void) {
46     car_status.body_yaw_angle = 0.0f;
47     update_car_body_transformation();
48     if (camera_a.type == CAMERA_DRIVER) {
49         set_ViewMatrix_for_driver();
50         ViewProjectionMatrix = ProjectionMatrix * ViewMatrix;
51     }
52     car_status.front_wheel_roll_angle = 0.0f;
53     car_status.front_wheel_turn_angle = 0.0f;
54     update_car_front_wheel_rotate_matrix();
55 }
56
57 void draw_wheel_and_nut() {
58     glUniform3f(loc_primitive_color, 0.000f, 0.808f, 0.820f); // color name: DarkTurquoise
59     draw_geom_obj(GEOM_OBJ_ID_CAR_WHEEL); // draw wheel
60
61 for (int i = 0; i < 5; i++) {
62     ModelMatrix_Car_Nut = glm::rotate(A, TO_RADIAN*2.0f*i, glm::vec3((B), (C), (D)));
63     ModelMatrix_Car_Nut = glm::translate(ModelMatrix_Car_Nut, glm::vec3(WHEEL_NUT_RADIUS, 0.0f, WHEEL_NUT_Z_OFFSET));
64     ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_Car_Nut;
65 }
66
67 void draw_car(void) {
68     glUniform3f(loc_primitive_color, 0.690f, 0.769f, 0.871f); // color name: LightSteelBlue
69     draw_geom_obj(GEOM_OBJ_ID_CAR_BODY); // draw i-th nut
70 }
71
72 void draw_wheel_and_nut() {
73     glUniform3f(loc_primitive_color, 0.498f, 1.000f, 0.831f); // color name: Aquamarine
74     draw_geom_obj(GEOM_OBJ_ID_CAR_WHEEL); // draw body
75
76     glLineWidth(5.0f);
77     draw_axes(); // draw MC axes of body
78     glLineWidth(1.0f);
79
80     ModelMatrix_Car_Driver = glm::translate(ModelMatrix_Car_BODY, glm::vec3(-3.0f, 0.5f, 2.5f));
81     ModelMatrix_Car_DRIVER = glm::rotate(ModelMatrix_Car_DRIVER, TO_RADIAN*0.0f, glm::vec3(0.0f, 1.0f, 0.0f));
82
83     ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_Car_DRIVER;
84     glUniformMatrix4fv(loc_ModelViewProjectionMatrix[0], 1, GL_FALSE, &ModelViewProjectionMatrix[0]);
85     glLineWidth(5.0f);
86     draw_axes(); // draw camera frame at driver seat
87     glLineWidth(1.0f);
88
89     ModelMatrix_Car_WHEEL *= car_status.Matrix_Car_WHEEL_ROTATE;
90     ModelMatrix_Car_WHEEL *= car_status.Matrix_Car_WHEEL_ROTATE;
91     glUniformMatrix4fv(loc_ModelViewProjectionMatrix[0], 1, GL_FALSE, &ModelViewProjectionMatrix[0]);
92
93     ModelMatrix_Car_WHEEL = glm::translate(ModelMatrix_Car_BODY, glm::vec3(-3.9f, -3.5f, 4.5f));
94     ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_Car_WHEEL;
95     glUniformMatrix4fv(loc_ModelViewProjectionMatrix[1], GL_FALSE, &ModelViewProjectionMatrix[1]);
96     glUniformMatrix4fv(loc_ModelViewProjectionMatrix[2], GL_FALSE, &ModelViewProjectionMatrix[2]);
97     draw_wheel_and_nut(); // draw wheel 0
98
99     ModelMatrix_Car_WHEEL = glm::translate(ModelMatrix_Car_BODY, glm::vec3(3.9f, -3.5f, 4.5f));
100    ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_Car_WHEEL;
101    ModelMatrix_Car_WHEEL = glm::scale(ModelMatrix_Car_WHEEL, glm::vec3(1.0f, 1.0f, -1.0f));
102    ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_Car_WHEEL;
103    glUniformMatrix4fv(loc_ModelViewProjectionMatrix[1], GL_FALSE, &ModelViewProjectionMatrix[1]);
104    draw_wheel_and_nut(); // draw wheel 2
105
106    ModelMatrix_Car_WHEEL = glm::translate(ModelMatrix_Car_BODY, glm::vec3(3.9f, -3.5f, -4.5f));
107    ModelMatrix_Car_WHEEL = glm::scale(ModelMatrix_Car_WHEEL, glm::vec3(1.0f, 1.0f, -1.0f));
108    ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_Car_WHEEL;
109    glUniformMatrix4fv(loc_ModelViewProjectionMatrix[1], GL_FALSE, &ModelViewProjectionMatrix[1]);
110    draw_wheel_and_nut(); // draw wheel 3

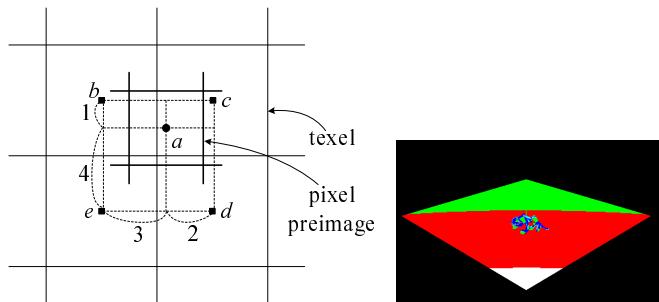
```

[CSE4170: 기초 컴퓨터 그래픽스]

기 말 고 사

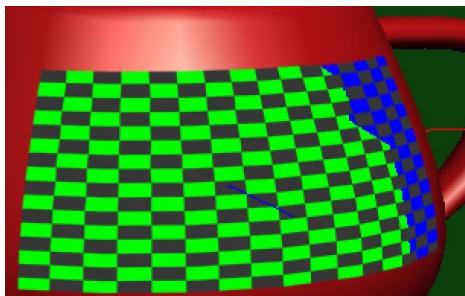
담당교수: 임 인 성

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

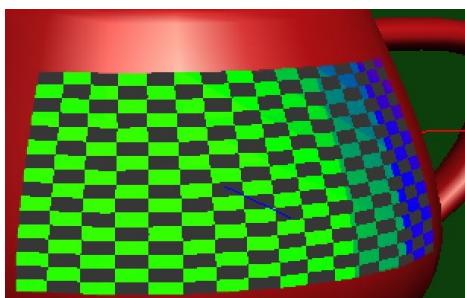


(a) 텍셀 색깔의 계산

(b) 맵맵 레벨의 결정



(c) 텍스춰 필터의 적용 결과 1



(d) 텍스춰 필터의 적용 결과 2

Figure 1: 텍스춰 필터링

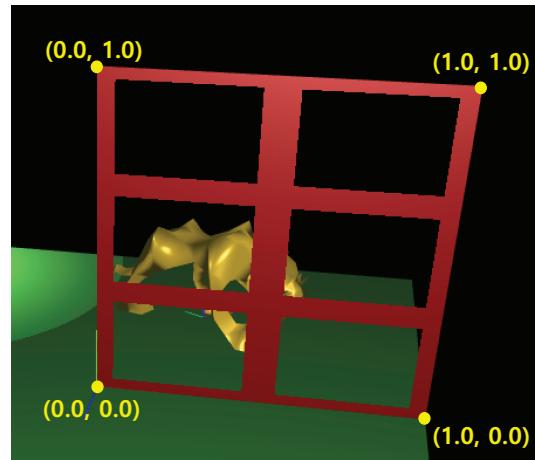
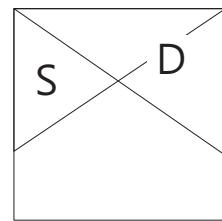
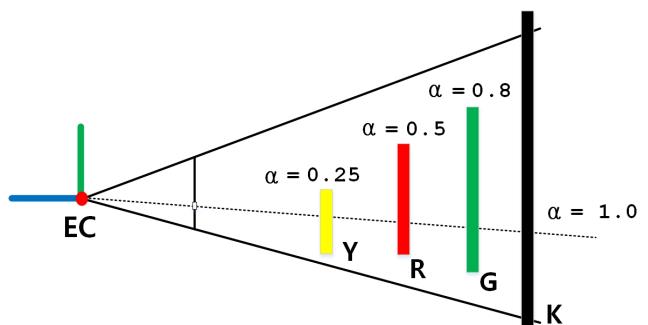


Figure 2: 스크린 효과 생성



(a) 색깔 혼합 예 1



(b) 색깔 혼합 예 2

Figure 3: OpenGL을 사용한 색깔 혼합

1. 다음은 텍스춰 필터링에 관한 문제이다. 그림 1 을 보면서 답하라.

- (a) 레벨 0의 밀맵 텍스처의 한 텍셀이 나타내는 영역의 면적은 레벨 3의 밀맵 텍스처의 한 텍셀이 나타내는 영역의 면적의 몇 배일까?
- (b) 텍셀당 두 바이트를 사용하는 해상도가 1024×1024 인 레벨 0의 텍스처에 대해 가능한 모든 레벨까지 밀맵을 구성할 경우 몇 바이트의 메모리 공간이 필요할까?
- (c) 그림 1(a)에서 a 는 픽셀에 대한 원상(pre-image)의 중점을 가리키고, b 부터 e 까지는 a 를 포함하는 주변 텍셀의 중심점을 나타낸다. GL_LINEAR가 의미하는 텍스처 필터링을 적용할 때 결과로 구해지는 색깔에서 d 텍셀의 색깔이 차지하는 비율을 기술하라.
- (d) 위 문제에서 GL_NEAREST가 의미하는 텍스처 필터링을 적용할 때의 비율을 기술하라.
- (e) 그림 1(b)에는 바닥에 대한 사각형(삼각형 두 개로 구성)에 한 개의 텍스처를 매핑한 후 밀매핑 기능을 사용할 때 적용되는 레벨 값으로 구역을 나눈 상태를 도시하고 있다. 현재 세 영역으로 구성이 되어 있는데, 어떤 순서대로 레벨 값이 높을지, 그리고 그 이유가 무엇인지 정확히 기술하라.
- (f) 그림 1(c)와 (d)는 축소 상황에서 각각 서로 다른(반대가 되는) 텍스처 필터링 방법을 적용한 렌더링 결과이다. 과연 두 텍스처 필터링 방법에 어떤 차이가 있는지 각 방법을 두 가지 방향에서 정확히 기술하라.

2. 다음은 스크린 효과 생성을 위한 OpenGL 프래그먼트 쉐이더 코드의 일부이다.

```
=====
:
in vec2 v_pos_sc;
:
void main(void) {
    if (screen_effect) {
        float x_mod, y_mod;
        x_mod = mod((A), 1.0f);
        y_mod = mod((B), 1.0f);
        if ( (x_mod > 0.1f) && (x_mod < 0.9f)
            && (y_mod > 0.1f) && (y_mod < 0.9f) )
            discard;
    }
    final_color = l_e(v_pos_EC,
                      normalize(v_norm_EC));
}
=====
```

(a) 그림 2의 스크린 사각형의 각 꼭지점에는 $v_pos_sc = (v_pos_sc.x, v_pos_sc.y)$ 속성으로 설정한 값이 주어져 있다. 이 프로그램이 올바르게 작동하기 위하여 (A) 와 (B) 에 들어갈 내용을 OpenGL Shading Language 문법에 맞게 정확히 기술하라. (참고: $mod(x, y)$ returns $x - y \cdot floor(\frac{x}{y})$.)

(b) (위 문제를 정확히 해결하였다는 가정 하에) 만약 두 번째 if 문장을 다음과 같이 바꿀 경우,

```
if ( (x_mod < 0.1f) || (x_mod > 0.9f) || (y_mod < 0.1f) || (y_mod > 0.9f) )
```

그림 2의 스크린 효과가 어떻게 바뀔지 기술하거나 그림으로 설명하라.

3. 다음은 색깔 혼합(Color Blending)에 관한 문제이다. $(c_S \alpha_S)$ 와 $(c_D \alpha_D)$ 를 두 이미지 S와 D의 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) 어떤 미리 곱한 색깔이 $(0.8, 0.8, 0.8, 0.8)$ 이라면, 이는 그 화소를 어떻게 칠해야 한다는 것인지 정확히 기술하라.

(b) 그림 3(a)가 암시하는 합성 연산의 경우 해당하는 F_S 와 F_D 의 값은 각각 얼마인가?

(c) 만약 S over D 연산을 적용한다면, 합성 후 α_O 는 어떤 값을 가질지를 S와 D의 관련 값을 사용하여 정확히 기술하라.

(d) 그림 3(b)의 상황을 고려하자. 결과 이미지의 한 화소에 네 개의 물체 Y, R, G, 그리고 K가 순서대로 투영되고 있는데, 이 물체들은 각각 순서대로 ‘순수한’ 노란색, 빨간색, 초록색, 그리고 검정색을 띠고 있으며, 각 물체의 불투명도(opacity)는 각 물체 옆에 기술되어 있다. 지금 이 물체들의 색깔을 앞에서 뒤로 가면서(front-to-back order) over 연산을 사용하여 합성한다고 할 때, Y와 R 물체까지 합성하였을 때의 불투명도 값이 무엇일지 기술하라.

(e) (바로 위 문제에 이어서) 이 물체들의 색깔을 뒤에서 앞으로 가면서(back-to-front order) over 연산을 사용하여 합성한다고 할 때, K, G 그리고 R 물체까지 합성하였을 때의 색깔이 무엇인지 미리곱한 색깔 형식으로 기술하라.

- (f) 다음은 투명한 물체들을 그려주는 어떤 프로그램의 프래그먼트 쉐이더 코드이다. 여기서 유니폼 변수 `u_flag_blending`이 0, 1, 또는 2 값을 가질 경우, 각각 합성 기능을 사용하지 않거나, 뒤에서 앞으로 가면서 `over` 연산을 사용하여 합성을 하거나, 또는 앞에서 뒤로 가면서 `over` 연산을 사용하여 합성을 해주게 된다. 또한, 함수 `obj_C(*,*)`는 현재 프래그먼트에 칠할 실제 색깔(미리 곱한 색깔이 아닌)을 계산해주며, 불투명도 값은 프래그먼트 속성 변수 `frag_a`로 들어온다. 만약 `u_flag_blending` 값을 1로 설정할 경우, 올바른 결과를 얻기 위하여 `glBlendFunc(*,*)` 함수 호출 시 사용할 두 인자를 아래에서 찾아 순서대로 기술하라.

```

GL_ZERO, GL_ONE,
GL_SRC_ALPHA, GL_DST_ALPHA,
GL_ONE_MINUS_SRC_ALPHA,
GL_ONE_MINUS_DST_ALPHA

=====
#version 420
:
uniform int u_flag_blending = 0;
in vec3 v_position_EC;
in vec3 v_normal_EC;
in float frag_a;
layout (location = 0) out vec4 f_c;

vec3 obj_C(in vec3 P_EC, in vec3 N_EC) {
    vec3 object_color;
    : // Compute the fragment's RGB color.
    return object_color;
}

void main(void) {
    vec3 C = obj_C(v_position_EC,
                    normalize(v_normal_EC));
    if (u_flag_blending == 0)
        f_c = vec4(C, 1.0f);
    else if (u_flag_blending == 1)
        f_c = vec4( (가) , frag_a);
    else if (u_flag_blending == 2)
        f_c = vec4( (나) , (다) );
}
=====
```

- (g) (바로 위 문제에 이어서) 이 경우 위 코드의 (가)에 들어갈 내용을 OpenGL Shading Language의 문법에 맞게 정확히 기술하라.
 (h) 만약 `u_flag_blending` 값을 2로 설정할 경

우, 이 프로그램이 제대로 작동하기 위하여 `glBlendFunc(*,*)` 함수 호출에 필요한 두 인자를 순서대로 기술하라.

- (i) (바로 위 문제에 이어서) 이 경우 위 코드의 (나)와 (다)에 들어갈 내용을 OpenGL Shading Language의 문법에 맞게 정확히 기술하라.
 (j) 다음과 같은 단순한 프래그먼트 쉐이더를 사용하는 색깔 합성 프로그램을 고려하자.

```
=====
#version 420
uniform vec4 u_fragment_color;
layout (location = 0) out vec4 final_color;
void main(void) {
    final_color = u_fragment_color;
}
```

아래의 OpenGL 코드에서는 그림 4에 도시한 바와 같이 서로 다른 영역을 가지는 세 개의 사각형을 뒤에서 앞으로 오면서 차례대로 그려주고 있다. 이 프로그램 수행 후 A 화소 지점에 해당하는 color buffer 영역에 저장되어 있는 (R,G,B,A) 값을 기술하라. 이 때 각 색깔 채널 값은 0과 1 사이의 값으로 정규화하여 분수로 답하라.
 (참고: 각 사각형을 구성하는 프래그먼트의 색깔은 대응하는 `glUniform4f()` 함수로 설정한 색깔로 설정이 됨)

```
=====
void display(void) {
    glDisable(GL_DEPTH_TEST);
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_DST_ALPHA);
    glUseProgram(h_ShaderProgram_PS);
    glUniform4f(loc_u_fragment_color,
                1.0f, 0.0f, 0.0f, 0.5f);
    draw_Rectangle_2();
    glUniform4f(loc_u_fragment_color,
                1.0f, 0.0f, 0.0f, 0.5f);
    draw_Rectangle_1();
    glUniform4f(loc_u_fragment_color,
                0.0f, 0.0f, 1.0f, 1.0f);
    draw_Rectangle_0();
    glUseProgram(0);
    glDisable(GL_BLEND);
    glEnable(GL_DEPTH_TEST);
    glutSwapBuffers();
}
```

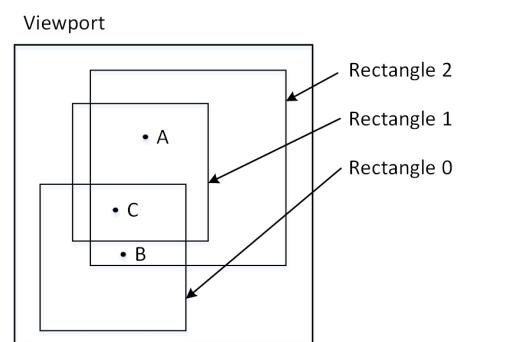


Figure 4: 세 개의 사각형 영역

- (k) 위 프로그램에서 blending factor를 설정하는 부분을 다음과 같이 변경할 경우,

```
glBlendFunc(GL_ONE_MINUS_SRC_ALPHA,
            GL_ONE);
```

이 프로그램 수행 후 A 화소 지점에 해당하는 color buffer 영역에 저장되어 있는 (R,G,B,A) 값을 기술하라. 이 때 각 색깔 채널 값은 0과 1 사이의 값으로 정규화하여 분수로 답하라.

- (l) (바로 위 문제에 이어서) 이 프로그램 수행 후 B 화소 지점에 해당하는 color buffer 영역에 저장되어 있는 (R,G,B,A) 값을 기술하라. 이 때 각 색깔 채널 값은 0과 1 사이의 값으로 정규화하여 분수로 답하라.

4. 다음은 안개 효과 생성을 위한 OpenGL 프래그먼트 쉐이더 코드의 일부이다. 아래의 코드와 그림 5를 보면서 답하라.

```
=====
fog_factor = (far_dist - length(Pe.xyz))
            / (far_dist - near_dist);
fog_factor = clamp(fog_factor, 0.0f,
                    1.0f);
final_color = mix(fog_color,
                  shaded_color, fog_factor);
=====
```

- (a) 이 코드는 OpenGL의 눈좌표계(EC)에서 계산을 하고 있다. 문맥 상 그림 5의 한 점 P_e 지점에서의 안개 효과 생성을 위하여 d_0 와 d_1 중 어떤 거리 척도를 사용하고 있는가?

- (b) d_0 와 d_1 중 위 코드에서 사용하고 있는 것이 아닌 다른 척도를 사용하려면, 첫 번째 문장을 어떻게 수정을 해야할지, OpenGL

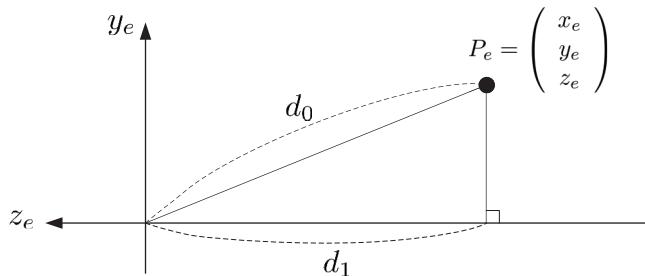


Figure 5: 안개 효과 생성

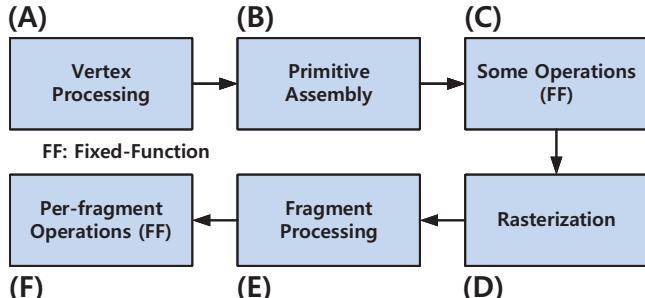


Figure 6: 실시간 렌더링 파이프라인

Shading Language의 문법에 맞게 기술하라. (참고: $0 < \text{near_dist} < \text{far_dist}$)

- (c) `mix()` 함수는 선형 보간에 기반을 두고 있다. 다음과 같은 변수 값에 대해,

```
fog_color = (0.8, 0.8, 0.8, 1.0),
shaded_color = (0.4, 0.2, 0.8, 1.0),
fog_factor = 0.4,
```

`final_color`의 결과 값을 기술하라.

5. 다음은 실시간 렌더링 파이프라인에 관한 문제이다. 그림 6의 렌더링 파이프라인을 보면서 한 학기 동안 수업에서 배운 OpenGL 렌더링 파이프라인을 바탕으로 답하라. 아래의 문제에서 별도로 기술하지 않으면 ‘정상적인 OpenGL 렌더링’ 과정을 가정한다. (필요할 경우 NDC, CC, WdC, MC, EC, 그리고 WC 등의 기호를 사용하여 답할 것)

- ‘정규화된 3차원 필름’을 기준으로 설정된 좌표계가 존재하는 박스의 기호를 기술하라.
- 투명한 물체 효과를 생성하기 위하여 색깔을 섞어주는 계산을 가장 자연스럽게 수행 할 수 있는 지점에 대한 박스의 기호를 기술하라.
- GL_LINEAR라는 OpenGL 인자가 직접적으로 적용이 되는 박스의 기호를 기술하라.

- (d) Gouraud 쉐이딩 계산 시 자연스럽게 lighting equation이 적용이 되는 지점에 대한 박스의 기호를 기술하라.
- (e) 삼각형을 구성하는 꼭지점들의 회전 방향을 통하여 그 삼각형의 앞면이 보이는지 뒷면이 보이는지를 판별하는 지점이 존재하는 박스의 기호를 기술하라.
- (f) 계층적 모델링을 위한 기하 변환이 빈번하게 일어나는 박스의 기호를 기술하라.
- (g) (A) 박스는 OpenGL의 어느 좌표계에서 어느 좌표계까지의 기하 계산을 담당하는가?
- (h) 꼭지점 속성 값이 프래그먼트로 전달이 되는 지점에 대한 박스의 기호를 기술하라.
- (i) (C) 박스는 OpenGL의 어느 좌표계에서 어느 좌표계까지의 기하 계산을 담당하는가?
- (j) 구조적으로 프레임 버퍼를 가장 빈번하게 접근해야하는 박스의 기호를 기술하라.
- (k) 뷰포트 변환이 수행이 되는 지점이 존재하는 박스의 기호를 기술하라.
- (l) 뷰잉 변환이 수행이 되는 지점이 존재하는 박스의 기호를 기술하라.
- (m) 원근 투영시 원근감이 생성이 되는 지점이 존재하는 박스의 기호를 기술하라.
- (n) 카메라를 기준으로 설정된 좌표계가 존재하는 박스의 기호를 기술하라.
- (o) `glm::perspective()` 함수가 설정한 3차원 영역을 벗어난 기하 프리미티브들을 제거하는 계산이 수행되는 박스의 기호를 기술하라.
- (p) 픽셀 단위의 안개 효과를 가장 자연스럽게 생성할 수 있는 지점에 대한 박스의 기호를 기술하라.

6. 다음은 어떤 버텍스 쉐이더 코드의 일부인데, 여기서 `Pos`, `pos`, `Norm`, 그리고 `norm`은 모두 `vec3` 타입의 변수들이다.

```
:
void main(void) {
    gl_Position = M_t*vec4(pos, 1.0f);
    Pos = vec3(M_p*vec4(pos, 1.0f));
    Norm = normalize(M_n*norm);
}
```

- (a) 마지막 두 문장은 좌표와 법선 벡터 방향이 각각 `pos`와 `norm`인 점에 대한 아핀 변환을 수행하고 있다. 보편적인 렌더링 상황에서 `M_p` 행렬이 다음과 같을 때, `M_n`은 어떤 행렬이어야 하는지 정확히 기술하라. (주의:

행과 열의 개수가 정확해야 함)

$$M_p = \begin{bmatrix} 0 & 0 & -1 & 2 \\ 0 & 1 & 0 & -3 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- (b) `norm` 변수에는 항상 길이가 1인 법선 벡터 값이 저장이 되어 있다고 가정하자. 만약 `M_p`가 위 문제와 같을 때 과연 마지막 문장에서의 `normalize()` 함수 호출이 필요한지 아니면 불필요한지를 정확한 수학적인 개념을 사용하여 기술하라.
- (c) 만약 `M_p` 행렬이 다음과 같을 때, `M_n`은 어떤 행렬이어야 하는지 정확히 기술하라.
(주의: 행과 열의 개수가 정확해야 함)

$$M_p = \begin{bmatrix} 0 & 0 & -2 & 1 \\ 0 & 2 & 0 & -2 \\ 2 & 0 & 0 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- (d) 이 코드에서 래스터화 과정 중 프래그먼트 쉐이더로 전달이 되는 정보를 저장하고 있는 변수들을 모두 기술하라.
- (e) 이 코드에서 버텍스 쉐이더를 수행하기 전에 기하 모델을 구성하는 매 꼭지점마다 설정해주어야 하는 정보를 저장하고 있는 변수들을 모두 기술하라.
- (f) 이 코드에서 버텍스 쉐이더를 수행하기 전에 기하 모델을 구성하는 매 꼭지점마다 설정할 필요가 없이 필요할 때만 설정해주면 되는 정보를 저장하고 있는 변수들을 모두 기술하라.
- (g) 만약 첫 문장 바로 직후에 다음과 같은 문장을 삽입하면,

`gl_Position.y *= -1.0f;`
렌더링 결과가 어떻게 바뀔지 기술하라.

7. 그림 7(a)에는 두 가지 형태의 풍의 조명 모델식이 주어져 있고, (b)-(d)에는 OpenGL 시스템에서 사용하는 기본 조명 공식이 기술되어 있다.

- (a) (a)의 공식에서 Lambert의 코사인 법칙을 표현해주는 부분만 정확히 기술하라.
- (b) (a)의 공식에서 평행 광원과 평행 투영을 사용할 경우, 물체의 위치와 방향이 바뀌었을 때 영향을 받는 변수를 모두 나열하라 (d_i 는 제외).
- (c) (a)의 공식에서 카메라에서 바라보는 방향에 직접적으로 영향을 받는 변수를 모두 나열하라.

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (R_i \circ V)^n\} \quad (1)$$

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (N \circ H_i)^n\} \quad (2)$$

(a) 풍의 조명 모델 식의 두 예 ($\|N\| = \|V\| = \|L_i\| = \|R_i\| = \|H_i\| = 1$)

$$\begin{aligned} \mathbf{c} = & \mathbf{e}_{cm} + \mathbf{a}_{cm} * \mathbf{a}_{cs} + \\ & \sum_{i=0}^{n-1} (att_i)(spot_i)[\mathbf{a}_{cm} * \mathbf{a}_{cli} + (\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli}) \mathbf{d}_{cm} * \mathbf{d}_{cli} + (f_i)(\mathbf{n} \odot \hat{\mathbf{h}}_i)^{s_{rm}} \mathbf{s}_{cm} * \mathbf{s}_{cli}] \end{aligned}$$

(b) OpenGL 시스템의 조명 공식

$$att_i = \begin{cases} \frac{1}{k_{0i} + k_{1i}\|\overrightarrow{\mathbf{VP}}_{pli}\| + k_{2i}\|\overrightarrow{\mathbf{VP}}_{pli}\|^2}, & \mathbf{P}_{pli}'s w \neq 0, \\ 1.0, & \text{otherwise} \end{cases}$$

(c) 빛의 감쇠 효과의 계산

$$spot_i = \begin{cases} (\overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{cli})^{s_{rl}}, & c_{rl} \neq 180.0 \text{ and } (A) \geq (B), \\ 0.0, & c_{rl} \neq 180.0 \text{ and } (A) < (B), \\ 1.0, & c_{rl} = 180.0 \end{cases}$$

(d) 스폷 광원 효과의 계산

Figure 7: 조명 모델 공식 예

- (d) (a)의 공식에서 현재 풍의 조명 모델 식이 적용되고 있는 물체 표면 지점의 좌표를 P 라고 하자. (b)의 공식에서 이 P 에 해당하는 변수(또는 수식)를 기술하라.
- (e) (b)의 공식에서 (a)의 공식의 d_i 에 해당하는 변수(또는 수식)를 기술하라.
- (f) (b)의 공식에서 \odot 은 주어진 두 벡터 \mathbf{u} 와 \mathbf{v} 에 대하여 어떠한 값을 계산해주는 연산자인지 정확히 기술하라.
- (g) (b)의 공식에서 $\mathbf{0}$ 벡터가 아닌 임의의 벡터 \mathbf{v} 에 대해 $\hat{\mathbf{v}}$ 는 무엇을 의미하는가?
- (h) 정반사 방향(reflection direction), 즉 정반사 물질이 입사 광선을 가장 강하게 반사시키는 방향을 (b) 공식의 변수들을 사용하여 표현하라. (참고: $\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli}$ 은 양수가 가정함)
- (i) (b)의 공식에서 아래 그림이 암시하는 효과를 조절할 수 있는 변수를 기술하라.



- (j) (b)의 공식에서 간접 조명(indirect illumination) 효과를 (근사적이나마) 생성하는데 필요한 간접적으로 들어오는 빛, 즉 간접적인 입사 광선과 직접적인 연관성이 있는 변수들을 모두 기술하라.
- (k) (d)의 공식에서 문맥상 (A)와 (B)에 공통으로 들어갈 수식을 정확히 기술하라.
- 8. 9쪽에는 OpenGL Shading Language를 사용하여 가급적 그림 7(b)의 풍의 조명 모델 식에 기반을 두어 풍 쉐이딩 방법을 구현한 버텍스 쉐이더와 프래그먼트 쉐이더 프로그램이 주어져 있다.
 - (a) 이 두 쉐이더 코드에서 OpenGL의 절단 좌표계(Clip Coordinate) 공간에서의 값을 가지는 변수들을 모두 기술하라.
 - (b) 이 프래그먼트 쉐이더에서 i 번째 광원 $u_light[i]$ 가 평행 광원(parallel or direc-

- tional light)이라면, 문맥 상 이 구조변수의 어떤 필드값이 어떤 값을 가져야 할까?
- (c) 73번 문장 수행 후 `tmp_vec4.y`에 저장이 되는 값의 정확한 기하적인 의미를 기술하라.
- (d) 점 광원을 사용할 경우 그림 7(a)의 조명 공식 (1)에서의 벡터 L_i 값이 계산이 되는 문장 번호를 기술하라.
- (e) 그림 7(c)의 i 번째 광원에 해당하는 k_{2i} 값을 저장하고 있는 변수 (또는 식)를 이 코드에서 찾아 OpenGL Shading Language 문법에 맞게 정확히 기술하라.
- (f) 104번 문장에서 `-normalize(P_EC)`는 정확히 어느 지점에서 어느 지점을 향한 벡터인지 기술하라.
- (g) 104번 문장에서 계산해주는 벡터 `H_EC`의 이름을 기술하라.
- (h) 104번 문장의 (가) 부분에 들어갈 수식을 OpenGL Shading Language 문법에 맞게 정확히 기술하라.
- (i) 105번 문장의 (나)와 (다)에 들어갈 두 벡터에 해당하는 기호 각각을 그림 7(a)의 조명 공식 (2)에서 찾아 기술하라. (참고: 순서는 상관 없음)
- (j) 111번 문장의 (라)에 들어갈 변수의 값을 그림 7(b)의 조명 공식의 기호들을 사용하여 기술하라.
- (k) 84번 문장의 (마) 부분에 들어갈 수식을 OpenGL Shading Language 문법에 맞게 정확히 기술하라.
- (l) 그림 7(b)의 조명 식의 변수 f_i 의 사용 목적이 이 쉐이더에서는 어떻게 달성이 되는지 관련 문장 번호를 지칭하면서 정확히 설명하라.
9. 10쪽의 코드는 계층적 모델링 변환을 통하여 자동차를 세상 좌표계로 그려주는 OpenGL 프로그램이다. 이 코드와 그림 8을 보면서 답하라.
- (a) 이 프로그램은 그림 8에 주어진 자동차를 구성하는 트리 구조를 어떤 방식으로 탐색하고 있는지 자료 구조 시간에 배운 컴퓨터공학 용어를 사용하여 기술하라.
- (b) 프로그램 문맥상 62번 문장의 (A)에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라.
- (c) 프로그램 문맥상 62번 문장의 (B), (C), 그리고 (D)에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라. (힌트: 그림 8의 내용을 잘 살펴볼 것)

- (d) 이 프로그램의 26번 문장은 자동차가 움직임에 따라 두 앞 바퀴가 좌우로 회전하는 모습을 생성해주는 기하 변환을 계산하고 있다(`draw_car()` 함수에서 적용이 됨). 문맥상 (E), (F), 그리고 (G)에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라.
- (e) 이 프로그램의 29번 문장은 자동차가 움직임에 따라 네 바퀴가 진행 방향으로 회전하는 모습을 생성해주는 기하 변환을 계산하고 있다(`draw_car()` 함수에서 적용이 됨). 문맥상 (H), (I), 그리고 (J)에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라.
- (f) 이 프로그램에서 94번 문장 수행 후 변수 `ModelMatrix_CAR_WHEEL`에 저장되는 4행 4열의 내용을 그림 8의 행렬 기호를 사용하여 표현하라.
- (g) 57번 문장의 `draw_wheel_and_nut()` 함수가 92번 문장에서 호출되어 수행이 되는 과정에서, 변수 i 가 2일 때 63번 문장 수행 결과 변수 `ModelMatrix_CAR_NUT`에 저장되는 행렬의 내용을 그림 8의 행렬 기호를 사용하여 표현하라. 이 순간에 행렬 `car_status.Matrix_CAR_WHEEL_ROTATE`의 내용은 M_R 이라고 가정함.
- (h) 이 프로그램의 85번 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). 이 자동차의 운전석에서 세상을 바라볼 때 오른쪽 방향은 자동차 바디(CAR_BODY)의 모델링 좌표계를 기준으로 어느 축 방향을 향하고 있을까? “양의 x 축” 또는 “음의 x 축”과 같이 해당 축과 방향을 정확히 기술할 것.
- (i) 이 프로그램의 92번 문장과 97번 문장이 그려주는 0번과 1번 바퀴와는 달리 104번 문장과 110번 문장이 그려주는 2번과 3번 바퀴에 대해서는 크기변환(scale)이 수행되는 이유는 무엇일까?
- (j) 이 프로그램의 42번 문장은 카메라를 운전석에 배치할 때의 뷰잉 변환 행렬을 계산해주고 있다. 이 문장에서 변수 `ModelMatrix_CAR_BODY_to_DRIVER`는 자동차 바디의 모델링 좌표계의 원점에 정렬되어 있는 카메라 프레임을 이 좌표계 상에서 운전석의 위치로 변환 시켜주는 강체변환을 저장하고 있다. 문맥 상 이 프로그램이

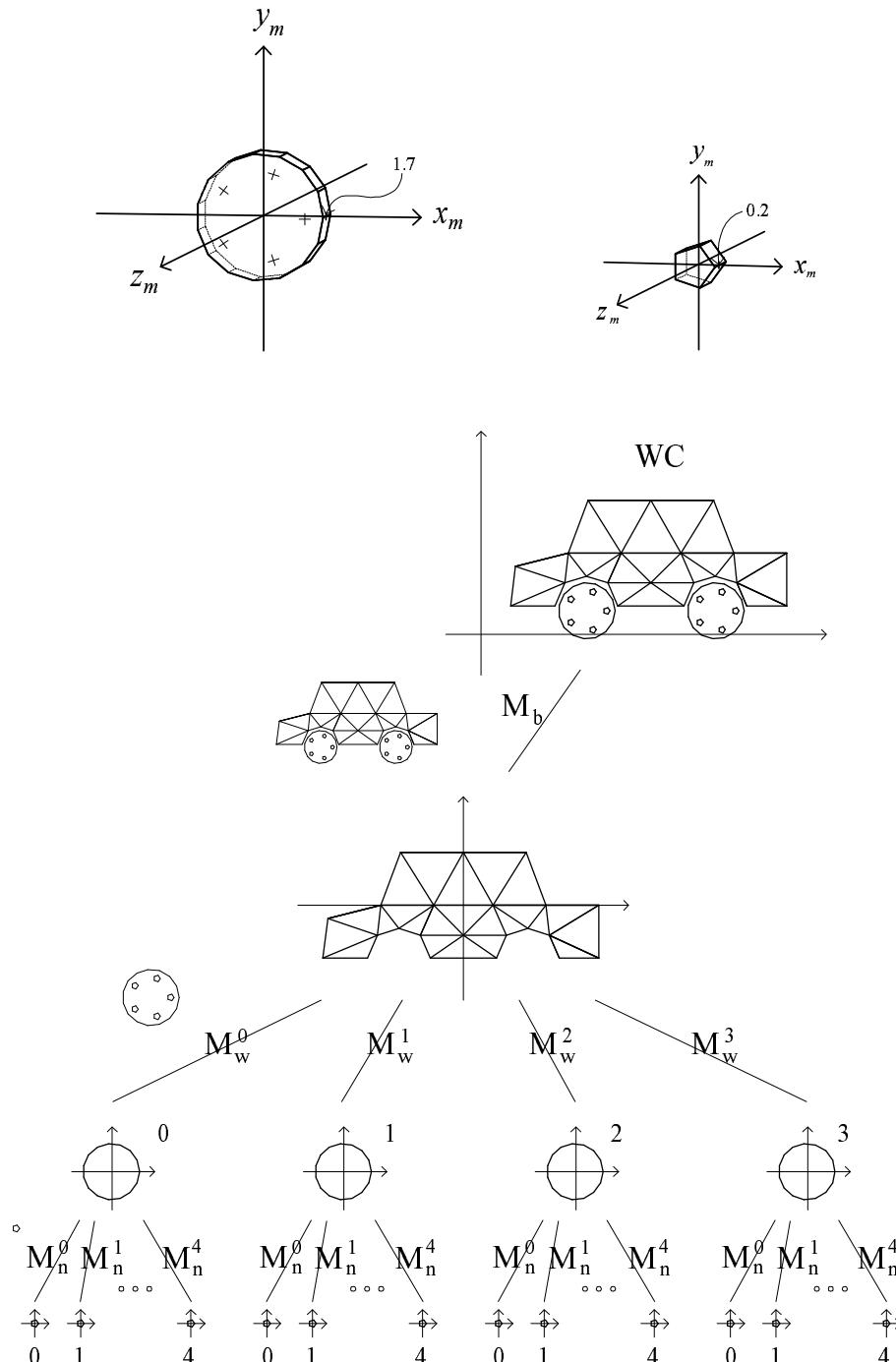


Figure 8: 자동차의 계층적 표현

제대로 작성하기 위하여 (K)에 들어갈 내용을 이 프로그램에서 사용한 변수를 사용하여 정확히 기술하라. (힌트: 뷔잉 변환은 세상 좌표에 존재하는 카메라 프레임을 좌

표계 원점을 중심으로 정렬시켜주는 변환임을 상기할 것)

```

1 /***** VERTEX SHADER *****/
2 #version 430
3
4 uniform mat4 u_ModelViewProjectionMatrix;
5 uniform mat4 u_ModelViewMatrix;
6 uniform mat3 u_ModelViewMatrixInvTrans;
7
8 layout(location = 0) in vec3 a_position;
9 layout(location = 1) in vec3 a_normal;
10 out vec3 v_position_EC;
11 out vec3 v_normal_EC;
12
13 void main(void) {
14     v_position_EC = vec3(u_ModelViewMatrix*vec4(a_position, 1.0f));
15     v_normal_EC = normalize(u_ModelViewMatrixInvTrans*a_normal);
16
17     gl_Position = u_ModelViewProjectionMatrix*vec4(a_position, 1.0f);
18 }
19
20 /***** FRAGMENT SHADER *****/
21 #version 430
22
23 struct LIGHT {
24     vec4 position;
25     vec4 ambient_color, diffuse_color, specular_color;
26     vec4 light_attenuation_factors; // compute this effect only if .w != 0.0f
27     vec3 spot_direction;
28     float spot_exponent;
29     float spot_cutoff_angle;
30     bool light_on;
31 };
32 }
33 struct MATERIAL {
34     vec4 ambient_color;
35     vec4 diffuse_color;
36     vec4 specular_color;
37     vec4 emissive_color;
38     float specular_exponent;
39 };
40 }
41
42 uniform vec4 u_global_ambient_color;
43 #define NUMBER_OF_LIGHTS_SUPPORTED 4
44 uniform LIGHT u_light[NUMBER_OF_LIGHTS_SUPPORTED];
45 uniform MATERIAL u_material;
46
47 const float zero_f = 0.0f;
48 const float one_f = 1.0f;
49
50 in vec3 v_position_EC;
51 in vec3 v_normal_EC;
52 layout(location = 0) out vec4 final_color;
53
54 vec4 lighting_equation(in vec3 P_EC, in vec3 N_EC) {
55     vec4 color_sum;
56     float local_scale_factor, tmp_float;
57     vec3 L_EC;
58
59     color_sum = u_material.emissive_color + u_global_ambient_color * u_material.ambient_color;
60
61     for (int i = 0; i < NUMBER_OF_LIGHTS_SUPPORTED; i++) {
62         if (!u_light[i].light_on) continue;
63
64         local_scale_factor = one_f;
65         if (u_light[i].position.w != zero_f) {

```

```

66             L_EC = u_light[i].position.xyz - P_EC.xyz;
67
68         if (u_light[i].light_attenuation_factors.w != zero_f) {
69             vec4 tmp_vec4;
70
71             tmp_vec4.x = one_f;
72             tmp_vec4.z = dot(L_EC, L_EC);
73             tmp_vec4.y = sqrt(tmp_vec4.z);
74             tmp_vec4.w = zero_f;
75             local_scale_factor = one_f / dot(tmp_vec4, u_light[i].light_attenuation_factors);
76         }
77
78         L_EC = normalize(L_EC);
79
80         if (u_light[i].spot_cutoff_angle < 180.0f) { // [0.0f, 90.0f] or 180.0f
81             float spot_cutoff_angle = clamp(u_light[i].spot_cutoff_angle, zero_f, 90.0f);
82             vec3 spot_dir = normalize(u_light[i].spot_direction);
83
84             tmp_float = dot( (N), spot_dir);
85             if (tmp_float >= cos(radians(spot_cutoff_angle))) {
86                 tmp_float = pow(tmp_float, u_light[i].spot_exponent);
87             }
88             else
89                 tmp_float = zero_f;
90             local_scale_factor *= tmp_float;
91         }
92     }
93     else {
94         L_EC = normalize(u_light[i].position.xyz);
95     }
96
97     if (local_scale_factor > zero_f) {
98         vec4 local_color_sum = u_light[i].ambient_color * u_material.ambient_color;
99
100        tmp_float = dot(N_EC, L_EC);
101        if (tmp_float > zero_f) {
102            local_color_sum += u_light[i].diffuse_color*u_material.diffuse_color*tmp_float;
103
104            vec3 H_EC = normalize( (G) - normalize(P_EC));
105            tmp_float = dot( (L-H), (D) );
106            if (tmp_float > zero_f) {
107                local_color_sum += u_light[i].specular_color
108                    *u_material.specular_color*pow(tmp_float, u_material.specular_exponent);
109            }
110        }
111        color_sum += (R) *local_color_sum;
112    }
113 }
114 return color_sum;
115 }
116
117 void main(void) {
118     final_color = lighting_equation(v_position_EC, normalize(v_normal_EC));
119 }
120

```

```

1 #define WHEEL_NUT_RADIUS (1.7f - 0.5f)
2 #define WHEEL_NUT_Z_OFFSET 1.0f
3
4 #define FRONT_WHEEL_ROLL_ANGLE_DELTA 5.0f
5 #define FRONT_WHEEL_TURN_ANGLE_DELTA 2.5f
6 #define FRONT_WHEEL_TURN_ANGLE_MAX 45.0f
7 #define BODY_YAW_SENSITIVITY 0.1f
8
9 struct _CAR_STATUS {
10     float body_yaw_angle;
11     float front_wheel_roll_angle;
12     float front_wheel_turn_angle;
13     glm::mat4 Matrix_CAR_WHEEL_ROTATE;
14     int flag_body_yaw_mode;
15 } car_status;
16
17 void update_car_body_transformation(void) {
18     ModelMatrix_CAR_BODY = glm::rotate(glm::mat4(1.0f),
19         TO_RADIAN*car_status.body_yaw_angle, glm::vec3(0.0f, 1.0f, 0.0f));
20
21     ModelMatrix_CAR_BODY_INVERSE = glm::rotate(glm::mat4(1.0f),
22         -TO_RADIAN*car_status.body_yaw_angle, glm::vec3(0.0f, 1.0f, 0.0f));
23 }
24
25 void update_car_front_wheel_rotate_matrix(void) {
26     car_status.Matrix_CAR_WHEEL_ROTATE = glm::rotate(glm::mat4(1.0f),
27         TO_RADIAN*car_status.front_wheel_turn_angle, glm::vec3( (E), (F), (G) ));
28
29     car_status.Matrix_CAR_WHEEL_ROTATE = glm::rotate(car_status.Matrix_CAR_WHEEL_ROTATE,
30         -TO_RADIAN*car_status.front_wheel_roll_angle, glm::vec3( (H), (I), (J) ));
31 }
32
33 void set_ViewMatrix_for_world_viewer(void) {
34     ViewMatrix = glm::mat4(camera_wv.uaxis.x, camera_wv.vaxis.x, camera_wv.naxis.x, 0.0f,
35         camera_wv.uaxis.y, camera_wv.vaxis.y, camera_wv.naxis.y, 0.0f,
36         camera_wv.uaxis.z, camera_wv.vaxis.z, camera_wv.naxis.z, 0.0f,
37         0.0f, 0.0f, 0.0f, 1.0f);
38     ViewMatrix = glm::translate(ViewMatrix, -camera_wv.pos);
39 }
40
41 void set_ViewMatrix_for_driver(void) {
42     ViewMatrix = glm::affineInverse( (K) * ModelMatrix_CAR_BODY_to_DRIVER);
43 }
44
45 void initialize_car(void) {
46     car_status.body_yaw_angle = 0.0f;
47     update_car_body_transformation();
48     if (camera_type == CAMERA_DRIVER) {
49         set_ViewMatrix_for_driver();
50         ViewProjectionMatrix = ProjectionMatrix * ViewMatrix;
51     }
52     car_status.front_wheel_roll_angle = 0.0f;
53     car_status.front_wheel_turn_angle = 0.0f;
54     update_car_front_wheel_rotate_matrix();
55 }
56
57 void draw_wheel_and_nut() {
58     glUniform3f(loc_primitive_color, 0.000f, 0.808f, 0.820f); // color name: DarkTurquoise
59     draw_geom_obj(GEOM_OBJ_ID_CAR_WHEEL); // draw wheel
60
61     for (int i = 0; i < 5; i++) {
62         ModelMatrix_CAR_NUT = glm::rotate( (A), TO_RADIAN*72.0f*i, glm::vec3( (B), (C), (D) ));
63         ModelMatrix_CAR_NUT = glm::translate(ModelMatrix_CAR_NUT, glm::vec3(WHEEL_NUT_RADIUS, 0.0f, WHEEL_NUT_Z_OFFSET));
64         ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_NUT;

```

```

65         glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
66
67         glUniform3f(loc_primitive_color, 0.690f, 0.769f, 0.871f); // color name: LightSteelBlue
68         draw_geom_obj(GEOM_OBJ_ID_CAR_NUT); // draw i-th nut
69     }
70 }
71
72 void draw_car(void) {
73     glUniform3f(loc_primitive_color, 0.498f, 1.000f, 0.831f); // color name: Aquamarine
74     draw_geom_obj(GEOM_OBJ_ID_CAR_BODY); // draw body
75
76     glLineWidth(5.0f);
77     draw_axes(); // draw MC axes of body
78     glLineWidth(1.0f);
79
80     ModelMatrix_CAR_DRIVER = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(-3.0f, 0.5f, 2.5f));
81     ModelMatrix_CAR_DRIVER = glm::rotate(ModelMatrix_CAR_DRIVER, TO_RADIAN*90.0f, glm::vec3(0.0f, 1.0f, 0.0f));
82     ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_DRIVER;
83     glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
84     glLineWidth(5.0f);
85     draw_axes(); // draw camera frame at driver seat
86     glLineWidth(1.0f);
87
88     ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(-3.9f, -3.5f, 4.5f));
89     ModelMatrix_CAR_WHEEL *= car_status.Matrix_CAR_WHEEL_ROTATE;
90     ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
91     glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
92     draw_wheel_and_nut(); // draw wheel 0
93
94     ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(3.9f, -3.5f, 4.5f));
95     ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
96     glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
97     draw_wheel_and_nut(); // draw wheel 1
98
99     ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(-3.9f, -3.5f, -4.5f));
100    ModelMatrix_CAR_WHEEL *= car_status.Matrix_CAR_WHEEL_ROTATE;
101    ModelMatrix_CAR_WHEEL = glm::scale(ModelMatrix_CAR_WHEEL, glm::vec3(1.0f, 1.0f, -1.0f));
102    ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
103    glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
104    draw_wheel_and_nut(); // draw wheel 2
105
106    ModelMatrix_CAR_WHEEL = glm::translate(ModelMatrix_CAR_BODY, glm::vec3(3.9f, -3.5f, -4.5f));
107    ModelMatrix_CAR_WHEEL = glm::scale(ModelMatrix_CAR_WHEEL, glm::vec3(1.0f, 1.0f, -1.0f));
108    ModelViewProjectionMatrix = ViewProjectionMatrix * ModelMatrix_CAR_WHEEL;
109    glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
110    draw_wheel_and_nut(); // draw wheel 3
111 }

```

[CSE4170: 기초 컴퓨터 그래픽스]

기 말 고 사

담당교수: 임 인 성

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

1. 다음은 색깔 혼합(Color Blending)에 관한 문제이다. $(c_S \alpha_S)$ 와 $(c_D \alpha_D)$ 를 두 이미지 S와 D의 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.

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

- 만약 결과 값이 $(0.8, 0.0, 0.0, 0.8)$ 이라 할 때, 이는 그 화소를 어떻게 칠해야 한다는 것인지 정확히 기술하라.
- 그림 1이 암시하는 합성 연산의 경우 해당하는 F_S 와 F_D 의 값은 각각 얼마인가?

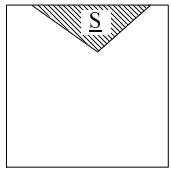


Figure 1: 색깔 혼합 예 1

- 만약 S over D 연산을 적용한다면, 합성 후 α_O 는 어떤 값을 가질지를 S와 D의 관련 값을 사용하여 정확히 기술하라.
- 그림 2의 상황을 고려하자. 지금 결과 이미지의 한 화소에 네 개의 물체 Y, R, G, 그리고 K가 순서대로 투영되고 있는데, 이 물체들은 각각 순서대로 ‘순수한’ 노란색, 빨간색, 초록색, 그리고 검정색을 띠고 있으며, 각 물체의 불투명도 (opacity)는 각 물체 옆에 기술되어 있다. 지금 이 물체들의 색깔을 앞에서 뒤로 가면서 (front-to-back order) over 연산을 사용하여 합성한다고

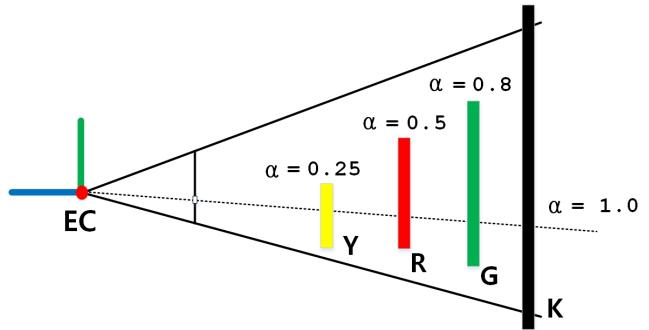


Figure 2: 색깔 합성 예 2

할 때, Y, R, 그리고 G 물체까지 합성하였을 때의 불투명도 값이 무엇일지 기술하라.

- (바로 위 문제에 이어서) 네 개의 물체 모두에 대한 합성이 끝났을 때, 최종 결과 색깔을 미리 곱한 색깔 형식으로 표현하라.
- 아래에는 투명한 물체들을 그려주는 프로그램의 fragment shader가 주어져 있다. 여기서 uniform variable `u_flag_blending`이 0, 1, 또는 2 값을 가질 경우, 각각 합성 기능을 사용하지 않거나, 뒤에서 앞으로 가면서 합성을 하거나, 또는 앞에서 뒤로 가면서 합성을 해주게 된다. 또한, 함수 `obj_c(*,*)`는 현재 fragment에 칠할 실제 색깔(미리 곱한 색깔이 아닌)을 계산해주며, 각 물체를 그리기 직전 해당하는 불투명도 값을 uniform variable인 `u_frag_a`에 설정을 해주도록 되어 있다. 만약 `u_flag_blending` 값을 1로 설정할 경우, 올바른 결과를 얻기 위하여, `glBlendFunc(*,*)` 함수 호출 시 사용할 두 인자를 아래에서 찾아 순서대로 기술하라.

```
GL_ZERO, GL_ONE,
GL_SRC_ALPHA, GL_DST_ALPHA,
GL_ONE_MINUS_SRC_ALPHA,
GL_ONE_MINUS_DST_ALPHA
```

```
=====
#version 420
:
uniform int u_flag_blending = 0;
uniform float u_frag_a = 1.0f;
in vec3 v_position_EC;
in vec3 v_normal_EC;
layout (location = 0) out vec4 f_c;

vec3 obj_c(in vec3 P_EC, in vec3 N_EC) {
    vec3 object_color;
    :
    // Compute the object's color.
    return object_color;
}

void main(void) {
    vec3 C = obj_c(v_position_EC,
                    normalize(v_normal_EC));
    if (u_flag_blending == 0)
        f_c = vec4(C, 1.0f);
    else if (u_flag_blending == 1)
        f_c = vec4( (가) , (나) );
    else if (u_flag_blending == 2)
        f_c = vec4( (다) , u_frag_a);
}
=====
```

- (g) (바로 위 문제에 이어) 이 경우 fragment shader의 (가)와 (나)에 들어갈 내용을 OpenGL Shading Language의 문법에 맞게 정확히 기술하라.
- (h) 만약 u_flag_blending 값을 2로 설정할 경우, 이 프로그램이 제대로 작동하기 위하여, glBlendFunc(*,*) 함수 호출에 필요한 두 인자를 순서대로 기술하라.
- (i) (바로 위 문제에 이어) 이 경우 fragment shader의 (다)에 들어갈 내용을 OpenGL Shading Language의 문법에 맞게 정확히 기술하라.
- (j) 다음과 같은 단순한 fragment shader를 사용하는 색깔 합성 프로그램을 고려하자.

```
=====
#version 420
uniform vec4 u_fragment_color;
layout (location = 0) out vec4 final_color;
void main(void) {
    final_color = u_fragment_color;
}
=====
```

아래의 OpenGL 코드에서는 그림 3에 도시한 바와 같이 서로 다른 영역을 가지는 세 개의 사각형을 뒤에서 앞으로 오면서

차례대로 그려주고 있다. 이 프로그램 수행 후 A 화소 지점에 해당하는 color buffer 영역에 저장되어 있는 실제 색깔(미리 곱한 색깔이 아닌) (R, G, B) 값을 기술하라. (주의: RGB 각 채널 값은 0과 1 사이의 값으로 정규화하여 답할 것)

```
=====
void display(void) {
    glDisable(GL_DEPTH_TEST);
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_DST_ALPHA);
    glUseProgram(h_ShaderProgram_PS);
    glUniform4f(loc_u_fragment_color,
                1.0f, 0.0f, 0.0f, 0.5f);
    draw_Rectangle_2();
    glUniform4f(loc_u_fragment_color,
                1.0f, 0.0f, 0.0f, 0.5f);
    draw_Rectangle_1();
    glUniform4f(loc_u_fragment_color,
                0.0f, 0.0f, 1.0f, 1.0f);
    draw_Rectangle_0();
    glUseProgram(0);
    glDisable(GL_BLEND);
    glEnable(GL_DEPTH_TEST);
    glutSwapBuffers();
}
=====
```

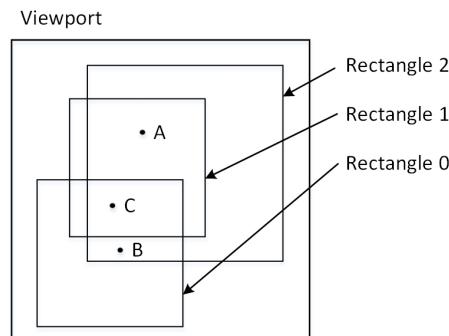


Figure 3: 세 개의 사각형 영역

- (k) (바로 위 문제에 이어) 이 프로그램 수행 후 B 화소 지점에 해당하는 color buffer 영역에 저장되어 있는 실제 색깔(미리 곱한 색깔이 아닌) (R, G, B) 값을 기술하라. (주의: RGB 각 채널 값은 0과 1 사이의 값으로 정규화하여 답할 것)
- (l) 위 프로그램에서 blending factor를 설정하

는 부분을 다음과 같이 변경할 경우,

```
glBlendFunc(GL_ONE_MINUS_DST_ALPHA,
            GL_ZERO);
```

이 프로그램 수행 후 C 화소 지점에 해당하는 color buffer 영역에 저장되어 있는 실제 색깔(미리 곱한 색깔이 아닌) (R, G, B) 값을 기술하라. (주의: RGB 각 채널 값은 0과 1 사이의 값으로 정규화하여 답할 것)

2. 다음은 카메라의 설정에 관한 문제이다. 그림 4에는 OpenGL Compatibility Profile에서 제공하는 `gluLookAt(*)` 함수의 호출에 대하여 뷰잉변환 행렬을 계산하는 과정(이하 “이 그림”)이 주어져 있으며, 그림 5에는 `glm::lookAt()` 함수에 대한 구현 코드(이하 “이 함수”)가 주어져 있는데, 이들을 참조하여 답하라.

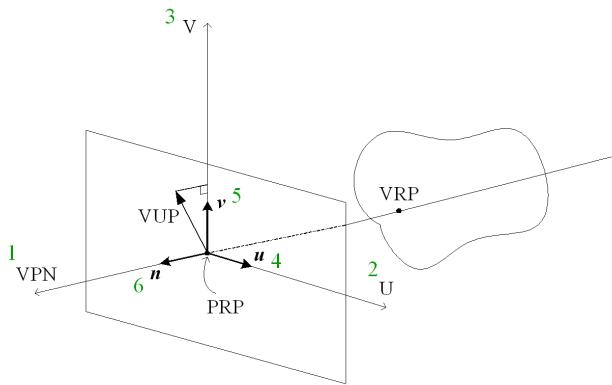


Figure 4: `gluLookAt(*)` 함수를 통한 카메라의 설정

- 이 그림에서 이 함수의 인자 중의 하나인 3차원 벡터 `eye`에 해당하는 기호의 이름을 기술하라.
- 이 그림의 u , v , 그리고 n 벡터는 각각 카메라를 기준으로 오른쪽, 위쪽, 그리고 바라보는 반대 방향에 대한 단위 벡터(unit vector)들이다. 이 함수의 Line (a) 문장 수행 후 벡터 `f`에 저장되는 값을 위의 벡터들을 사용하여 정확히 기술하라.
- 문맥 상 이 함수의 Line (b)와 Line (c) 문장의 `smile(*,*)` 함수는 두 개의 3차원 벡터를 인자로 받아 어떠한 계산을 해줄까?
- 이 함수의 Line (d) 문장 수행 결과 4행 4열 행렬 `Result`에는 어떤 값이 저장될까?
- 문맥 상 이 함수의 Line (e)와 Line (f)의 빈곳에 들어갈 내용을 이 프로그램의 문법에 맞게 정확히 기술하라.

3. 이 문제는 어떤 이진트리 구조를 사용하여 트리에 저장되어 있는 다각형들을 주어진 카메라 시점에 기준으로 앞에서 뒤로 가면서 정렬한 순서로 (in front-to-back order) 그려주는 문제에 관한 것이다. 우선 이 트리는 다음과 같이 정의된다.

```
=====
typedef struct _Polygon {
    int n_v;
    float *vertex;
    float *normal;
    float plane[4];
} Polygon;

typedef struct _TREE {
    Polygon *polygon;
    struct _TREE *fc, *bc;
} TREE;
=====
```

- (a) 다각형 한 개에 대한 정보를 저장해주는 `Polygon` 타입의 변수의 `plane[4]` 필드는 나머지 세 개의 변수들에 의해 정의되는 다각형을 포함하는 평면의 식 $a \cdot x + b \cdot y + c \cdot z + d = 0$ 의 네 개의 상수 a, b, c , 그리고 d 를 순서대로 저장하고 있는데, 이 다각형의 법선 벡터는 이 평면의 식에 반영이 되어 있다. 만약 이 다각형이 점 $(1, 1, 1)$ 을 지나고, 법선 벡터가 $(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$ 이라면, 배열 `plane[4]`에 들어갈 네 개의 값을 순서대로 기술하라.

- (b) 다음은 실제로 포인터 변수 `tree`가 가리키는 이진트리에 저장된 다각형들을 앞에서 기술한 순서대로 그려주는 함수이다. 문맥상 이 함수의 또 다른 인자 `p`는 어떤 값을 저장한 배열에 대한 포인터 변수일까?

```
=====
void draw_polygons(TREE *tree,
                   float *p) {
    if (tree == NULL) return;
    if (check_side(p, tree->polygon)
        == TREE_FRONT) {
        draw_polygons(tree->fc, p);
        draw_a_polygon(tree->polygon);
        draw_polygons(tree->bc, p);
    }
    else {
        (A)
    }
}
```

```

namespace glm {
    ...
    template <typename T, precision P> GLM_FUNC_QUALIFIER tmat4x4<T, P>
    lookAt(tvec3<T, P> const &eye, tvec3<T, P> const &center, tvec3<T, P> const &up) {
        tvec3<T, P> const f(normalize(center - eye)); // Line (a)
        tvec3<T, P> const s(normalize(smile(f, up))); // Line (b)
        tvec3<T, P> const u(smile(s, f)); // Line (c)
        tmat4x4<T, P> Result(1); // Line (d)
        Result[0][0] = s.x; Result[1][0] = s.y; Result[2][0] = s.z;
        Result[0][1] = u.x; Result[1][1] = u.y; Result[2][1] = u.z;
        Result[0][2] = -f.x; Result[1][2] = -f.y; Result[2][2] = -f.z;
        Result[3][0] = -dot(s, eye);
        Result[3][1] = ; // Line (e)
        Result[3][2] = ; // Line (f)
        return Result;
    }
}

```

Figure 5: glm::lookAt() 함수의 구현

```

}
=====

(c) 다음은 위 문제 코드의 check_side(*,*) 함수에 대한 코드인데, p가 가리키는 점이 tree->polygon->plane[4]가 정의하는 평면의 앞쪽에 있으면 TREE_FRONT를, 아니면 TREE_BACK을 리턴해주도록 되어 있다(편의상 이 점이 평면 상에 있을 경우 뒤쪽에 있다고 가정함). 이 함수의 (B) 부분에 들어갈 내용을 C/C++ 문법에 맞게 정확히 기술하라.

```

```

=====
#define TREE_BACK 0
#define TREE_FRONT 1
int check_side(float *p,
               Polygon *poly) {
    if ( (B) )
        return TREE_FRONT;
    else
        return TREE_BACK;
}
=====
```

(d) 앞의 코드에서 draw_polygons(*,*) 함수가 올바르게 작동하기 위하여 (A) 부분에 들어갈 내용을 C/C++ 문법에 맞게 정확히 기술하라.

(e) 이 이진트리에 저장되어 있는 다각형의 개

수가 n 이라할 때, draw_polygons(*,*) 함수의 시간 복잡도를 Big-O 기호를 사용하여 기술하라.

(f) 이 문제에서 사용한 이진트리의 영문 이름을 기술하라.

4. (바로 위 문제에 이어) 이번 문제도 바로 앞 문제에서 사용한 이진트리의 응용에 관한 문제이다. 지금 가상의 3차원 실내 공간을 자유롭게 돌아다니려 하는데, 사용자가 벽면을 뚫고 지나가지 못하도록 벽면들에 대한 다각형 정보를 저장하고 있는 이 이진트리를 사용하여 다음과 같은 함수 호출을 통하여 충돌검사를 수행 하려 한다.

LOS(tree, A, B);

LOS(*,*,*) 함수의 목적은 3차원 공간의 두 점 A와 B에 의해 정의된 선분이 실내 공간의 벽면에 해당하는 임의의 다각형과 교차를 하는지 판별하는 것인데, 이 함수는 다음과 같이 정의할 수 있다.

```

=====
int LOS(TREE *T, float *s, float *e) {
    int side_s, side_e, side_f, side_b;

    if (!T) return (A) ;
    side_s = check_side(s, T->polygon);
    side_e = check_side(e, T->polygon);
    if ((side_s == TREE_FRONT)
        && (side_e == TREE_FRONT)) {
```

```

        return LOS( C );
    }
    else if ((side_s == TREE_BACK)
              && (side_e == TREE_BACK)) {
        return LOS( D );
    }
    else {
        if ( (B) ) return 0;
        else {
            side_f = LOS(T->fc, s, e);
            side_b = LOS(T->bc, s, e);
            return (E);
        }
    }
}
=====
```

- (a) 이 함수는 s 와 e 가 가리키는 두 점을 연결한 선분이 최소한 한 개의 벽면 다각형과 교차하면 0을 아니면 1을 리턴해주도록 설계되어 있다. 문맥 상 이 코드의 (A)에 들어갈 상수 값을 기술하라.
- (b) 문맥 상 (B) 부분의 조건식은 언제 TRUE 값을 가지게 될지 정확히 기술하라.
- (c) 문맥 상 (C)에 들어갈 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라.
- (d) 문맥 상 (D)에 들어갈 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라.
- (e) 문맥 상 (E)에 들어갈 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라.

5. 그림 6은 자동차 차체 모델을 기준으로 회전변환 R 과 이동변환 T 를 통하여 운전석에 카메라를 배치한 후, 강체변환 M_B 를 통하여 차체를 세상 좌표계에 배치하는 모습을 도시하고 있다. 이때의 뷔잉변환 행렬 M_V 를 위의 세 개의 4행 4열 행렬을 사용하여 표현하라.
6. 그림 7(a)에는 두 가지 형태의 풍의 조명 모델식이 주어져 있고, (b)-(d)에는 OpenGL 시스템에서 사용하는 기본 조명 공식이 기술되어 있다.

- (a) (a)의 공식에서 H_i 를 이 공식의 다른 기호를 사용하여 수학적으로 정확히 표현하라.
- (b) (a)의 공식에서 Lambert's cosine law를 표현해주는 부분만 정확히 기술하라.
- (c) (a)의 공식에서 만약 평행 광원과 평행 투영을 사용할 경우, 물체의 위치와 방향이 바뀌었을 때 영향을 받는 변수를 모두 나열하라 (d_i 는 제외).

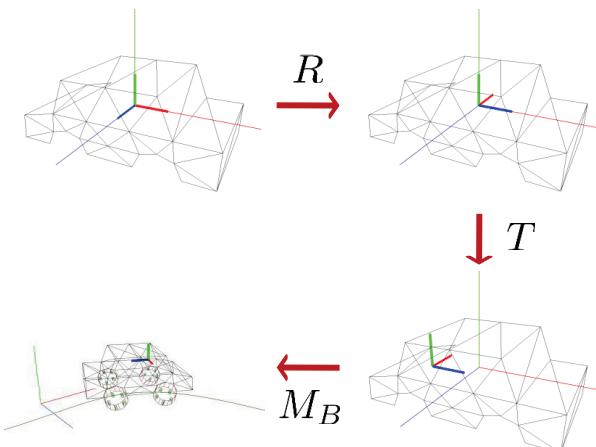


Figure 6: 자동차 운전석에 대한 카메라의 배치

- (d) (a)의 공식에서 카메라에서 바라보는 방향에 직접적으로 영향을 받는 변수를 모두 나열하라.
- (e) (a)의 공식에서 만약 점 광원을 사용할 경우 광원의 위치의 변화에 영향을 받는 변수를 모두 나열하라.
- (f) (a)의 공식에서 현재 풍의 조명 모델식이 적용되고 있는 물체 표면 지점의 좌표를 P 라고 하자. (b)의 공식에서 이 P 에 해당하는 변수(또는 수식)를 기술하라.
- (g) (b)의 공식에서 (a)의 공식의 d_i 에 해당하는 변수(또는 수식)를 기술하라.
- (h) (b)의 공식에서 \odot 은 주어진 두 벡터 u 와 v 에 대하여 어떠한 값을 계산해주는 연산자인지 정확히 기술하라.
- (i) (b)의 공식에서 $\mathbf{0}$ 벡터가 아닌 임의의 벡터 v 에 대해 \hat{v} 는 무엇을 의미하는가?
- (j) 정반사 방향(reflection direction), 즉 정반사 물질이 입사 광선을 가장 강하게 반사시키는 방향을 (b) 공식의 변수들을 사용하여 표현하라. ($\mathbf{n} \odot \overrightarrow{VP}_{pli}$ 은 양수가 가정함)
- (k) (b)의 공식에서 정반사로 인한 하이라이트 영역의 크기를 조절할 수 있는 변수를 기술하라.
- (l) (b)의 공식에서 간접 조명(indirect illumination) 효과를 (근사적이나마) 생성하는데 필요한 간접적으로 들어오는 빛, 즉 간접적인 입사 광선과 직접적인 연관성이 있는 변수들을 모두 기술하라.
- (m) (c)의 공식의 조건식 \mathbf{P}_{pli} 's $w \neq 0$ 이 만족되는 상황은 어떤 경우인지 정확히 기술하라.
- (n) (d)의 공식에서 문맥상 (A)와 (B)에 공통으로 들어갈 수식을 정확히 기술하라.

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (R_i \circ V)^n\} \quad (1)$$

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (N \circ H_i)^n\} \quad (2)$$

(a) 풍의 조명 모델 식의 두 예

$$\mathbf{c} = \mathbf{e}_{cm} + \mathbf{a}_{cm} * \mathbf{a}_{cs} + \sum_{i=0}^{n-1} (att_i)(spot_i)[\mathbf{a}_{cm} * \mathbf{a}_{cli} + (\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli}) \mathbf{d}_{cm} * \mathbf{d}_{cli} + (f_i)(\mathbf{n} \odot \hat{\mathbf{h}}_i)^{s_{rm}} \mathbf{s}_{cm} * \mathbf{s}_{cli}]$$

(b) OpenGL 시스템의 조명 공식

$$att_i = \begin{cases} \frac{1}{k_{0i} + k_{1i}\|\mathbf{VP}_{pli}\| + k_{2i}\|\mathbf{VP}_{pli}\|^2}, & \mathbf{P}_{pli}'s w \neq 0, \\ 1.0, & \text{otherwise} \end{cases}$$

(c) 빛의 감쇠 효과의 계산

$$spot_i = \begin{cases} (\overrightarrow{\mathbf{VP}}_{pli} \odot \hat{\mathbf{s}}_{cli})^{s_{rl}}, & c_{rl} \neq 180.0 \text{ and } (A) \geq (B), \\ 0.0, & c_{rl} \neq 180.0 \text{ and } (A) < (B), \\ 1.0, & c_{rl} = 180.0 \end{cases}$$

(d) 스폷 광원 효과의 계산

Figure 7: 조명 모델 공식 예

7. 그림 8의 OpenGL 렌더링 파이프라인을 보면서 답하라. 이 문제의 질문은 모두 ‘정상적인 렌더링 과정’을 가정한다. (어떤 지점을 지칭할 때에는 해당 상자의 기호 ((A)부터 (I)까지)를 사용할 것)

- (a) 이 그림에서 `out vec3 v_position_EC;` 문장과 `in vec3 v_position_EC;` 문장에서 정의된 동일한 이름의 두 변수간에 직접적으로 정보가 전달되는 지점은?
- (b) ‘정규화된 3차원 필름’이 정의되는 좌표계는 어느 상자와 어느 상자 사이의 지점에 해당할까?
- (c) 거리 값에 기반을 둔 ‘은면 제거 (hidden surface removal)’ 계산이 수행되는 지점은?
- (d) 화면의 윈도우 영역 중 실제로 그림이 그려지는 영역을 결정하는 계산이 수행되는 지점은?
- (e) `glDrawArrays(GL_TRIANGLES, 0, 900);` 문장에서 상수 `GL_TRIANGLES`이 가장 직접적으로 영향을 미치는 상자의 기호는?

- (f) 비로소 CC (Clip Coordinate)의 꼭지점을 NDC (Normalized Device Coordinate)의 꼭지점으로 변환해주는 상자의 기호는?
- (g) 꼭지점의 좌표를 (x, y, z, w) 라 할 때 보편적인 렌더링 과정 중 w 값이 1이 아닌 값을 가질 수 있는 지점을 모두 기술하라.
- (h) 사용자가 설정한 카메라의 위치와 방향 정보가 반영되는 지점은?
- (i) 보편적인 렌더링 과정 중 mipmapping 과정이 가장 빈번하게 수행되는 지점은?
- (j) Polygonal shading model 중의 하나인 Gouraud shading 기법을 구현할 때, 실제 Phong의 illumination model 식이 적용되는 상자의 기호는?
- (k) `glBlendFunc(GL_SRC_ALPHA, GL_DST_ALPHA);` 문장이 직접적으로 영향을 미치는 계산이 수행되는 지점은?
- (l) 보편적인 렌더링 과정에서 다음 기하변환

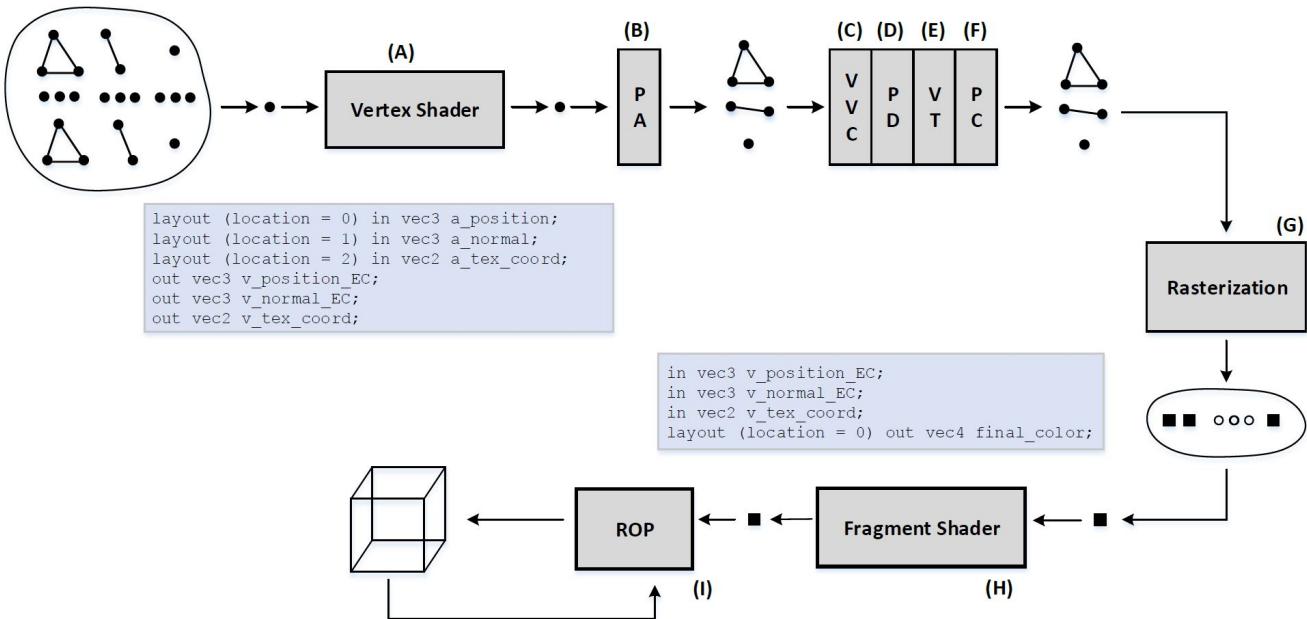
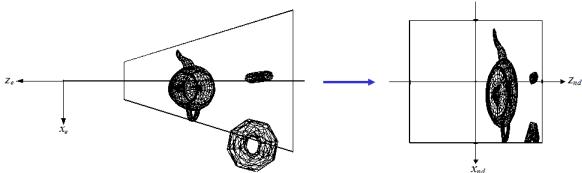


Figure 8: OpenGL 렌더링 파이프라인

행렬과 가장 관련이 높은 상자의 번호는?

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

- (m) 위의 행렬에서 f 값에 의해 정의되는 평면 뒤쪽에 있는 기하 물체가 화면에서 사라지게 되는 지점은?
- (n) 삼각형의 꼭지점의 나열 순서를 통하여 기하 물체의 일부를 제거할 수 있는 지점은?
- (o) `glFrontFace(GL_CW)`; 문장이 가장 직접적으로 영향을 미치는 지점은?
- (p) 다음 그림이 암시하는 렌더링 계산과 가장 직접적으로 관련이 있는 두 상자의 기호를 기술하라.



8. 다음은 텍스처 필터링에 관한 문제이다. 아래 그림을 보면서 답하라.

- (a) 레벨 4의 mip맵 텍스처의 한 텍셀이 나타내는 영역의 면적은 레벨 0의 mip맵 텍스처의 한 텍셀이 나타내는 영역의 면적의 몇 배일까?
- (b) 해상도가 512×512 이고 각 텍셀 당 GL_UNSIGNED_BYTE와 GL_RGBA 형식을 사용하는 텍스처 이미지에 대하여 모든 가능한 레벨에 대하여 mip맵을 구성할 경우 총 몇 바이트의 텍스처 메모리가 필요할까?
- (c) 그림 9(a)에서 a 는 픽셀에 대한 원상(pre-image)의 중점을 가리키고, b 부터 e 까지는 a 를 포함하는 주변 텍셀의 중심점을 나타낸다. b, c, d , 그리고 e 지점의 텍셀 색깔이 각각 $(1, 1, 1), (1, 0, 0), (1, 1, 0)$, 그리고 $(0, 1, 1)$ 이라고 하자. 각각 최근 필터와 선형 필터를 사용할 경우 a 지점에 대하여 계산되는 색깔은 무엇일까? 이 그림에서 숫자는 거리에 대한 비율을 나타내고 있음.
- (d) 그림 9(b)는 확대 필터는 GL_NEAREST, 그리고 축소 필터는 GL_LINEAR_MIPMAP_LINEAR를 사용한 경우의 렌더링 결과이다. 과연 이 그림의 상황은 축소 상황이 발생한 것인지, 아니면 확대 상황이 발생한 것인지 구체적으로 기술하라.
- (e) 그림 9(c)는 축소 상황이 발생한 경우이다. 서로 다른 레벨의 mip맵 텍스처에 대해서 서로 다른 색깔의 텍스처 이미지를 사용하였는데, 과연 이 그림은 삼선형 필터(trilinear)

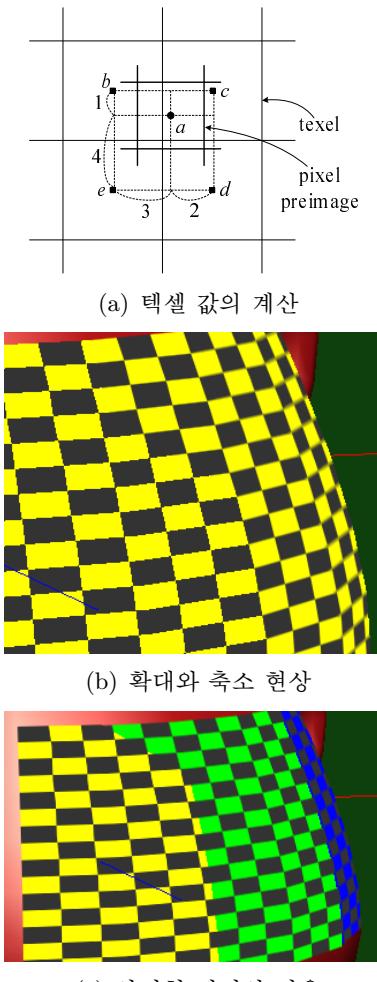


Figure 9: 텍스춰 필터링

interpolation filter)를 사용한 결과라 할 수 있는가? 예/아니오로 답하고 그렇게 답한 이유를 밝혀라.

9. 본 시험 문제지의 마지막 쪽에는 OpenGL Shading Language (GLSL)를 사용하여 가급적 그림 7의 풍의 조명 모델 식에 기반을 두어 풍 쉐이딩 방법을 구현한 버텍스 쉐이더와 프래그먼트 쉐이더 프로그램이 주어져 있다.

- (a) 보편적인 렌더링 상황에서, 어떠한 조건을 만족할 경우 5번 문장의 행렬의 왼쪽-위 3행 3열 행렬과 6번 문장 행렬의 내용이 동일할까?
 (b) 125번 문장에서 `corrected_normal_EC` 벡터를 정규화를 해주고 있는데, (정확한 계

산을 위해서) 이 과정이 필요한 이유는 무엇일까?

- (c) OpenGL 렌더링 과정중 view volume clipping 계산이 수행되는 좌표계에서의 좌표값을 가지는 변수를 모두 기술하라.
- (d) GPU의 메모리 영역에 사용자가 할당한 Vertex Buffer Object의 데이터와 직접적으로 연관이 있는 변수를 모두 기술하라.
- (e) 현재 처리하고 있는 광원이 점광원인지는 어떻게 판별하고 있는가?
- (f) 점 광원을 처리할 때 광원과 쉐이딩하려는 지점 간의 거리가 계산이 되는 문장의 번호를 기술하라.
- (g) 평행 광원을 처리할 때 빛의 감쇠 효과는 어떻게 처리가 될까?
- (h) 그림 7(c)의 i 번째 광원에 해당하는 k_{2i} 값을 저장하고 있는 변수 (또는 식)를 GLSL 문법에 맞게 정확히 기술하라.
- (i) 100번 문장 시점에서 그림 7(a)의 조명 공식 (1)에서 사용한 벡터 V 값을 이 쉐이더 코드의 문맥에서 GLSL 문법에 맞게 정확히 기술하라.
- (j) 92번 문장의 (가) 부분에 들어갈 내용을 GLSL 문법에 맞게 정확히 기술하라.
- (k) 이 쉐이더 코드에서는 halfway vector를 사용하여 쉐이딩을 하고 있다. 101번 문장의 (나) 부분에 들어갈 수식을 GLSL 문법에 맞게 정확히 기술하라.
- (l) (위 문제에 이어서) 102 문장의 (다) 부분에 들어갈 수식을 GLSL 문법에 맞게 정확히 기술하라.
- (m) 이 쉐이더 코드의 68번 문장에서 93번 문장의 내용 중 명확히 틀린 부분이 있다. 문장 번호와 함께 틀린 부분을 명확히 밝히고 어떻게 고쳐야며 할지 설명하라.
- (n) 108 문장의 (라) 부분에 들어갈 변수나 수식을 GLSL 문법에 맞게 정확히 기술하라.
- (o) 그림 7(b)의 조명 식의 변수 f_i 의 사용 목적이 이 쉐이더에서는 어떻게 달성이 되는지 관련 문장 번호를 지칭하면서 정확히 설명하라.

```

1 /*----- Vertex Shader -----*/
2 #version 430
3
4 uniform mat4 u_ModelViewProjectionMatrix;
5 uniform mat4 u_ModelViewMatrix;
6 uniform mat3 u_ModelViewMatrixInvTrans;
7
8 layout(location = 0) in vec3 a_position;
9 layout(location = 1) in vec3 a_normal;
10 layout(location = 2) in vec2 a_tex_coord;
11 out vec3 v_position_EC;
12 out vec3 v_normal_EC;
13 out vec2 v_tex_coord;
14
15 void main(void) {
16     v_position_EC = vec3(u_ModelViewMatrix*vec4(a_position, 1.0f));
17     v_normal_EC = normalize(u_ModelViewMatrixInvTrans*a_normal);
18     v_tex_coord = a_tex_coord;
19
20     gl_Position = u_ModelViewProjectionMatrix*vec4(a_position, 1.0f);
21 }
22
23 /*----- Fragment Shader -----*/
24 #version 430
25
26 struct LIGHT {
27     vec4 position;
28     vec4 ambient_color, diffuse_color, specular_color;
29     vec4 light_attenuation_factors;
30     vec3 spot_direction;
31     float spot_exponent;
32     float spot_cutoff_angle;
33     bool light_on;
34 };
35
36 struct MATERIAL {
37     vec4 ambient_color;
38     vec4 diffuse_color;
39     vec4 specular_color;
40     vec4 emissive_color;
41     float specular_exponent;
42 };
43
44 uniform vec4 u_global_ambient_color;
45 #define NUMBER_OF_LIGHTS_SUPPORTED 4
46 uniform LIGHT u_light[NUMBER_OF_LIGHTS_SUPPORTED];
47 uniform MATERIAL u_material;
48 uniform sampler2D u_base_texture;
49 uniform bool u_flag_texture_mapping = true;
50
51 const float zero_f = 0.0f;
52 const float one_f = 1.0f;
53
54 in vec3 v_position_EC;
55 in vec3 v_normal_EC;
56 in vec2 v_tex_coord;
57 layout(location = 0) out vec4 final_color;
58
59 vec4 lighting_equation_textured(in vec3 P_EC, in vec3 N_EC, in vec4 base_color) {
60     vec4 color_sum;
61     float local_scale_factor, tmp_float;
62     vec3 L_EC;
63
64     color_sum = u_material.emissive_color + u_global_ambient_color * base_color;
65     for (int i = 0; i < NUMBER_OF_LIGHTS_SUPPORTED; i++) {
66         if (!u_light[i].light_on) continue;
67         local_scale_factor = one_f;
68
69         if (u_light[i].position.w != zero_f) {
70             L_EC = u_light[i].position.xyz - P_EC.xyz;
71             if (u_light[i].light_attenuation_factors.w != zero_f) {
72                 vec4 tmp_vec4;
73                 tmp_vec4.x = one_f;
74                 tmp_vec4.z = dot(L_EC, L_EC);
75                 tmp_vec4.y = sqrt(tmp_vec4.z);
76                 tmp_vec4.w = zero_f;
77                 local_scale_factor = one_f / dot(tmp_vec4, u_light[i].light_attenuation_factors);
78             }
79             L_EC = normalize(L_EC);
80             if (u_light[i].spot_cutoff_angle < 180.0f) { // [0.0f, 90.0f] or 180.0f
81                 float spot_cutoff_angle = clamp(u_light[i].spot_cutoff_angle, zero_f, 90.0f);
82                 vec3 spot_dir = normalize(u_light[i].spot_direction);
83                 tmp_float = dot(-L_EC, spot_dir);
84                 if (tmp_float < cos(radians(u_light[i].spot_cutoff_angle))) {
85                     tmp_float = pow(tmp_float, u_light[i].spot_exponent);
86                 } else {
87                     tmp_float = zero_f;
88                     local_scale_factor *= tmp_float;
89                 }
90             } else {
91                 L_EC = normalize( (가) );
92             }
93             if (local_scale_factor > zero_f) {
94                 vec4 local_color_sum = u_light[i].ambient_color * u_material.ambient_color;
95
96                 tmp_float = dot(N_EC, L_EC);
97                 if (tmp_float > zero_f) {
98                     local_color_sum += u_light[i].diffuse_color*base_color*tmp_float;
99
100                vec3 H_EC = normalize( (나) );
101                tmp_float = dot( (나) );
102                if (tmp_float > zero_f) {
103                    local_color_sum += u_light[i].specular_color *
104                        u_material.specular_color*pow(tmp_float, u_material.specular_exponent);
105                }
106            }
107            color_sum += (나)*local_color_sum;
108        }
109    }
110 }
111 return color_sum;
112 }
113
114 void main(void) {
115     vec3 corrected_normal_EC;
116     vec4 base_color;
117
118     if (gl_FrontFacing) corrected_normal_EC = v_normal_EC;
119     else corrected_normal_EC = -v_normal_EC;
120
121     if (u_flag_texture_mapping)
122         base_color = texture(u_base_texture, v_tex_coord);
123     else
124         base_color = u_material.diffuse_color;
125     final_color = lighting_equation_textured(v_position_EC, normalize(corrected_normal_EC), base_color);
126 }

```

[CSE4170: 기초 컴퓨터 그래픽스]

기말고사

(담당교수: 임인성)

- 답은 연습지가 아니라 답안지에 기술할 것.
 - 답안지 공간이 부족할 경우, 답안지 뒷면에 기술하고, 해당 답안지 칸에 그 사실을 명기할 것.
1. 다음 쪽의 그림 1(a)에는 풍의 조명 모델 식의 두 예가 기술되어 있고, (b)-(d)에는 OpenGL 시스템에서 사용하는 기본 조명 공식이 주어져 있다.
- (a) 문맥상 (a)의 공식에서 잘못된 곳이 있으면 모두 찾아 바르게 고쳐라(없을 경우 ‘없음’이라고 답할 것).
 - (b) (a)의 공식에서 H_i 를 이 공식의 다른 기호를 사용하여 정확히 표현하라.
 - (c) (a)의 공식에서 Lambert's cosine law를 표현해주는 부분을 정확히 기술하라.
 - (d) (a)의 공식에서 만약 평행 광원과 평행 투영을 사용할 경우 물체의 위치와 방향이 바뀌었을 때 영향을 받는 변수를 모두 나열하라 (d_i 는 제외).
 - (e) (a)의 공식에서 카메라에서 바라보는 방향에 직접적으로 영향을 받는 변수를 모두 나열하라.
 - (f) (a)의 공식에서 만약 점 광원을 사용할 경우 광원의 위치의 변화에 영향을 받는 변수를 모두 나열하라.
 - (g) (a)의 공식에서 무한 관찰자 (infinite viewer) 기능과 관련한 인자 값을 변경할 때 직접적으로 영향을 받는 변수를 기술하라.
 - (h) (a)의 공식에서 현재 풍의 조명 모델 식이 적용되고 있는 물체 표면 지점의 좌표를 P 라고 하자. (b)의 공식에서 이 P 에 해당하는 변수(또는 수식)를 기술하라.
 - (i) OpenGL fixed-function pipeline에서 (b)의 공식은 WC, NDC, CC, WdC, MC, EC 중 어떤 좌표계에서 적용되는가?
 - (j) (b)의 공식에서 (a)의 공식의 d_i 에 해당하는 변수(또는 수식)를 기술하라.

- (k) (b)의 공식에서 \odot 은 주어진 두 벡터 \mathbf{u} 와 \mathbf{v} 에 대하여 어떠한 값을 계산해주는 연산자인지 정확히 기술하라.
- (l) (b)의 공식에서 $\mathbf{0}$ 벡터가 아닌 임의의 벡터 \mathbf{v} 에 대해 $\hat{\mathbf{v}}$ 는 무엇을 의미하는가?
- (m) 정반사 방향(reflection direction), 즉 정반사 물질이 입사 광선을 가장 강하게 반사시키는 방향을 (b) 공식의 변수들을 사용하여 표현하라. $\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli}$ 은 양수가 가정함.
- (n) (b)의 공식에서 만약 무한 관찰자 기능을 사용하지 않을 경우 카메라의 위치가 바뀌었을 때 이 공식에서 직접적으로 영향을 받는 변수(또는 식)를 모두 나열하라.
- (o) (b)의 공식에서 정반사로 인한 하이라이트 영역의 크기를 조절할 수 있는 변수를 기술하라.
- (p) (b)의 공식에서 간접 조명(indirect illumination) 효과를 (근사적이나마) 생성하는데 필요한 간접적으로 들어오는 빛, 즉 간접적인 입사 광선과 직접적인 연관성이 있는 변수들을 모두 기술하라.
- (q) 아래의 식은 \mathbf{h}_i 를 계산하는 과정을 기술하고 있다.

$$\mathbf{h}_i = \begin{cases} \overrightarrow{\mathbf{VP}}_{pli} + \overrightarrow{\mathbf{P}}_e, & v_{bs} = \text{TRUE}, \\ \overrightarrow{\mathbf{VP}}_{pli} + (a \ b \ c \ d)^t, & v_{bs} = \text{FALSE} \end{cases}$$

여기서 \mathbf{P}_e 의 동차좌표를 기술하라.

- (r) 만약 무한 관찰자(infinite viewer) 기능을 사용한다고 할 때, 위 문제의 식에서 a , b , c , 그리고 d 변수들의 값은 무엇이 될까?
- (s) (b)의 공식에서 다음과 같은 함수 호출에 ‘직접적인’ 영향을 받는 변수를 기술하라.

```
glLightfv(GL_LIGHT0, GL_POSITION,
           light_position);
```

- (t) (c)의 공식의 조건식 \mathbf{P}_{pli} 's $w \neq 0$ 이 만족되는 상황은 어떤 경우인지 정확히 기술하라.

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (R \circ V)^n\} \quad (1)$$

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (N \circ H_i)^n\} \quad (2)$$

(a) 풍의 조명 모델 식의 두 예

$$\mathbf{c} = \mathbf{e}_{cm} + \mathbf{a}_{cm} * \mathbf{a}_{cs} + \sum_{i=0}^{n-1} (att_i)(spot_i)[\mathbf{a}_{cm} * \mathbf{a}_{cli} + (\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli}) \mathbf{d}_{cm} * \mathbf{d}_{cli} + (f_i)(\mathbf{n} \odot \hat{\mathbf{h}}_i)^{s_{rm}} \mathbf{s}_{cm} * \mathbf{s}_{cli}]$$

(b) OpenGL 시스템의 조명 공식

$$att_i = \begin{cases} \frac{1}{k_{0i} + k_{1i}\|\overrightarrow{\mathbf{VP}}_{pli}\| + k_{2i}\|\overrightarrow{\mathbf{VP}}_{pli}\|^2}, & \mathbf{P}_{pli}'s w \neq 0, \\ 1.0, & \text{otherwise} \end{cases}$$

(c) 빛의 감쇠 효과의 계산

$$spot_i = \begin{cases} (\overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{dl_i})^{s_{rl_i}}, & c_{rl_i} \neq 180.0 \& \overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{dl_i} < \cos c_{rl_i}, \\ 0.0, & c_{rl_i} \neq 180.0 \& \overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{dl_i} \geq \cos c_{rl_i}, \\ 1.0, & c_{rl_i} = 180.0 \end{cases}$$

(d) 스폷 광원 효과의 계산

Figure 1: 1번 문제 공식

- (u) (d)의 공식은 스폷 광원으로 인한 입사 광선의 밝기를 계산하기 위한 식이다. 여기서 두 벡터 $\overrightarrow{\mathbf{P}}_{pli} \overrightarrow{\mathbf{V}}$ 와 $\hat{\mathbf{s}}_{dl_i}$ 가 의미하는 방향을 정확히 기술하라.
- (v) (d)의 공식에서 잘못된 것을 모두 찾아 바로 잡아라.

2. 다음은 카메라 및 광원의 설정에 관한 문제이다.

- (a) 광원을 카메라의 기준점(즉 눈 좌표계의 원점)에서 위로 3.0 만큼 떨어진 곳에 위치시키려 한다. 이를 위하여 아래의 코드에서 `glLightfv(GL_LIGHT0, GL_POSITION, li_pos);` 함수를 정확히 어느 지점에서 호출해야하는지 문장 번호를 사용하여 기술하고, 그때 (A), (B), (C)에 들어갈 내용은 무엇인지 기술하라.

```
GLfloat li_pos[4] = { (A), (B),
                      (C), 1.0 }; // (1)
```

:

```
glMatrixMode(GL_MODELVIEW); // (2)
```

```
glLoadIdentity(); // (3)
gluLookAt(v[0], v[1], v[2],
           c[0], c[1], c[2],
           0.0, 1.0, 0.0); // (4)
```

- (b) 초기 상태의 카메라 프레임이 세상 좌표계의 좌표축과 일치되어 있는 상태에서, 카메라 프레임에 대하여 $M_1 \rightarrow M_2 \rightarrow M_3$ 순서로 변환을 하였다면, 그 경우 뷰잉 변환 행렬 M_V 는 무엇인지 기술하라.
- (c) 초기 상태 $C_0 = (e, u, v, n) = ((0\ 0\ 0), (1\ 0\ 0), (0\ 1\ 0), (0\ 0\ 1))$ 인 카메라를 사용자의 조작을 통하여 $C_1 = ((0\ 10\ 10), (0\ 0\ 1), (0\ 1\ 0), (-1\ 0\ 0))$ 와 같은 상태로 변환하는 상황을 아래와 같은 뷰잉 변환 코드로 구현하려 한다.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMultMatrixf(cam_mat);
glTranslatef( (A), (B), (C));
이때 (A), (B), (C)에 들어갈 내용을 기술
```

하라.

- (d) (위 문제에 이어) 올바른 변환을 위하여 float cam_mat[16]; 과 같이 정의된 배열 cam_mat이 저장해야 할 16개 원소의 값을 순서대로 기술하라.

3. 다음은 배열 p[*][*]에 저장된, y_w 축에 수직인 평면 상의 경로를 따라 움직이는 자동차에서 바라본 세상을 렌더링해주는 코드의 일부이다.

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

    v[0] = ...; v[1] = ...; v[2] = ...; // (1)

    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);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    (2)
    draw_axes(); draw_path();

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

- (a) (1)번 문장에서 v 벡터의 값 v[0], v[1], v[2]의 값을 설정하고 있는데, 각 원소 값은 문맥상 어떤 값으로 설정되어야 할까? 세상 좌표계와 자동차 몸체의 모델링 좌표계 모두 위쪽이 y 축 방향에 해당함.

- (b) 문맥 상 set_up_rot_mat() 함수 안에서 배열 m과 minv의 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.

- (c) 그림 2는 자동차 몸체의 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 m과 minv가 나타내는 M_m 과 M_{minv} 등의 곱으로 표현하라.

- (d) 지금 운전석에 배치한 카메라를 사용하여 렌더링한다고 가정할 때, display() 함수의 (2)에 들어갈 뷰잉 변환 코드를 OpenGL 함수들을 적절히 사용하여 C언어 문법에 맞게 정확히 기술하라 (힌트: 카메라 프레임은 자동차 몸체를 세상 좌표계에 배치할 때 같이 움직임).

4. 그림 3의 OpenGL ES 렌더링 파이프라인을 보고 답하라. 이 문제의 질문은 모두 ‘정상적인 렌더링 과정’을 가정한다. (상자를 지칭할 때에는 해당 상자에 대하여 X와 같이 기술할 것)

- (a) 계층적인 구조를 가지는 복잡한 물체를 효과적으로 렌더링 해주는 hierarchical modeling과 가장 관련이 높은 상자의 기호는?
- (b) 물체가 투영되는 픽셀에 칠해질 최종 색깔을 계산해주는 상자의 기호는?
- (c) glEnable(GL_DEPTH_TEST); 문장과 가장 관련이 있는 상자의 기호는?
- (d) glCullFace(GL_BACK); 문장이 가장 직접적으로 영향을 미치는 상자의 기호는?
- (e) ‘정규화된 3차원 필름’이 정의되는 좌표계는 어느 상자와 어느 상자 사이의 지점에 해당할까?
- (f) 사용자가 설정한 방식대로 투영변환 행렬이 곱해지는 상자의 기호는?
- (g) 사실적인 렌더링을 위하여 바나나 물체의 표면에 사실적인 질감을 입히는 부분의 상자의 기호는?
- (h) Polygonal shading model 중의 하나인 Phong shading 기법을 구현할 때, 실제 Phong의 illumination model 식이 적용되는 상자의 기호는?

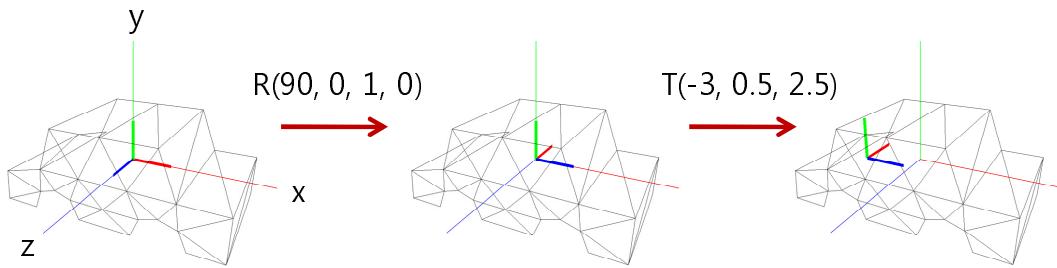


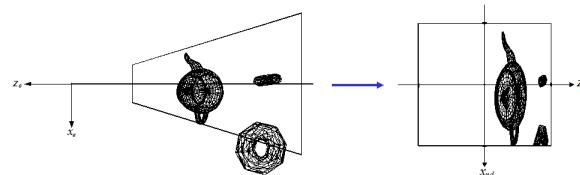
Figure 2: 운전석 카메라의 배치

- (i) 다음 기하변환 행렬과 가장 관련이 높은 상자의 번호는?

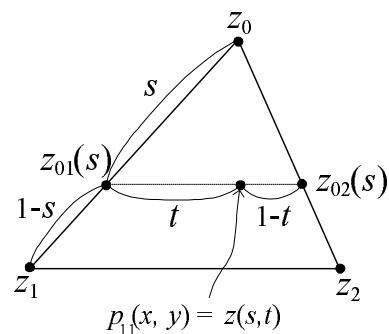
$$\begin{bmatrix} \frac{w}{2} & 0 & 0 & o_x \\ 0 & \frac{h}{2} & 0 & o_y \\ 0 & 0 & \frac{f-n}{2} & \frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- (j) 한 기하 물체가 다른 물체로 인하여 가려지게 되는 부분을 찾아내어 제거하는 상자의 기호는?
 (k) 꼭지점의 속성이 픽셀의 속성으로 전달이 되는 상자의 기호는?
 (l) 확률적으로 `void glRotatef(angle, x, y, z);` 함수가 의미하는 계산이 가장 빈번하게 일어나는 상자의 기호는?
 (m) 사진 촬영 시 현상한 필름을 인화지의 특정 영역에 인화를 해주는 과정과 가장 관련이 높은 상자의 기호는?
 (n) Near clipping plane보다 앞 쪽에 있는 기하 물체가 제거되는 상자의 기호는?
 (o) `gluPerspective(fovy, aspect, zNear, zFar);` 함수를 통하여 설정한 내용이 직접적으로 영향을 미치는 상자를 모두 기술하라.
 (p) Polygonal shading model 중의 하나인 Gouraud shading 기법을 구현할 때, 실제 Phong의 illumination model 식이 적용되는 상자의 기호는?
 (q) 사용자가 설정한 카메라의 위치와 방향에 대한 처리를 해주는 상자의 기호는?
 (r) 원근투영을 적용할 때, 실제 원근감이 생성 과정과 가장 직접적인 관련이 있는 상자의 기호는?
 (s) 일반적인 렌더링 과정에서 꼭지점의 homogenous coordinate의 W 좌표값이 1이 아닐 가능성이 높은 상자의 기호를 모두 기술하라.

- (t) `glDrawArrays(GL_TRIANGLES, 0, 333);` 문장에서 상수 `GL_TRIANGLES`이 직접적으로 영향을 미치는 상자의 기호는?
 (u) 꼭지점이 나열된 방향에 따라 삼각형을 제거할 수 있는 상자의 기호는?
 (v) 비로서 CC (Clip Coordinate)의 꼭지점을 NDC (Normalized Device Coordinate)의 꼭지점으로 변환해주는 상자의 기호는?
 (w) 어느 두 상자 사이에서 꼭지점의 좌표값이 항상 -1과 1사이의 값만 가질지 그 두 상자 기호를 기술하라.
 (x) 주어진 꼭지점에 대해 사용자가 화면에 띄운 윈도우를 기준으로 하는 좌표값을 직접적으로 계산해주는 상자의 기호는?
 (y) 다음 그림이 암시하는 렌더링 계산과 직접적으로 관련이 있는 두 상자의 기호를 기술하라.



- (z) 다음 그림이 암시하는 계산과 가장 직접적인 관련이 있는 상자의 기호는?



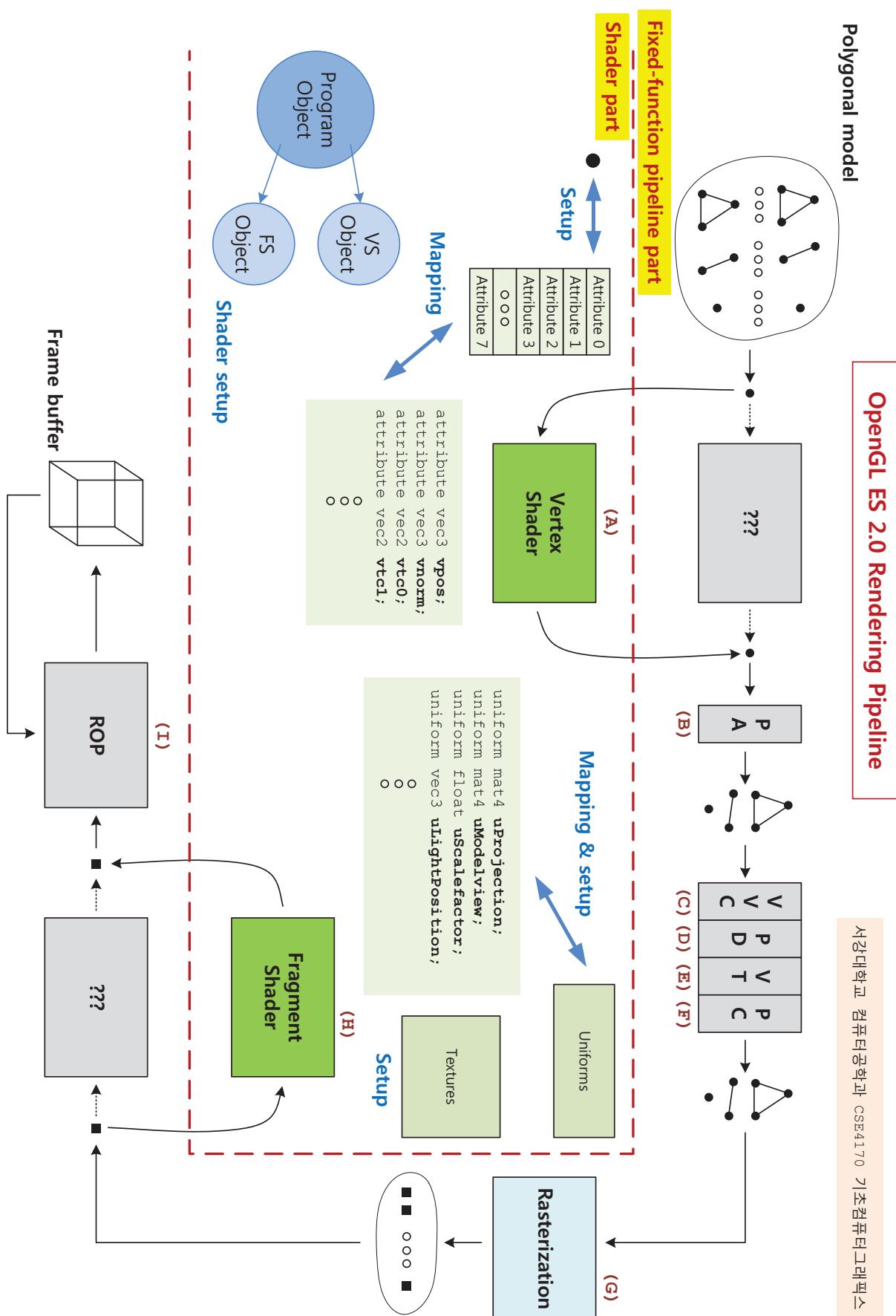


Figure 3: OpenGL ES 렌더링 파이프라인

5. 본 시험 문제지의 마지막 두 쪽에는 OpenGL ES 2.0 Shading Language를 사용하여 풍 쉐이딩 방법을 구현한 버텍스 쉐이더와 프래그먼트 쉐이더 프로그램이 주어져 있다(이하 OpenGL ES 쉐이더 코드).

(a) 3차원 공간의 점 p 를 4행 4열 행렬 M 이 나타내는 아핀 변환을 통해 기하 변환한다고 할 때, p 에서의 법선 벡터 (normal vector) n 에 대해 동일한 아핀 변환을 하려면 n 에 어떤 행렬을 곱해야 할지 M 을 사용하여 표현하라.

(b) 다음은 Direct3D의 쉐이더인 HLSL Shader에 대한 설명의 일부이다.

As a minimum, a vertex shader must output ??? in homogeneous clip space.

OpenGL ES 쉐이더 코드에서 이 설명과 가장 밀접한 연관이 있는 변수는 무엇인가?

(c) OpenGL ES 쉐이더 코드에서 기존 방식에서 흔히 사용되었던 `glVertex3f(x, y, z);`; 함수와 가장 밀접한 연관이 있는 변수은 무엇인가?

(d) 실제로 풍의 조명 모델 공식이 적용되는 물체의 각 지점에서 표면에 수직인 방향이 최종적으로 결정이 되는 문장의 번호를 기술하라.

(e) 71번 문장의 조건문에서 이 조건이 성립하지 않는다는 것은 무엇을 의미하는가?

(f) OpenGL ES 쉐이더 코드에서 그림 1(b)의 조명 공식의 d_{cli} 에 해당하는 변수의 이름은 무엇인가?

(g) 문맥상 그림 1(b)의 조명 공식에서 103번 문장의 `max(c_zero, dot(N_eye, L))` 값에 해당하는 변수(또는 식)를 정확히 기술하라.

(h) 74번부터 78번 문장까지는 그림 1(c)의 식을 계산해주고 있다. 문맥상 76번 문장의 `sqrt()` 함수의 인자를 OpenGL ES 쉐이딩 언어 문법에 맞게 정확히 기술하라.

(i) 문맥상 그림 1(c)의 k_{0i} 값을 저장하고 있는 변수(또는 식)를 OpenGL ES 쉐이딩 언어 문법에 맞게 정확히 기술하라

(j) 106번 문장에서 사용되고 있는 `-normalize(v_position).xyz` 벡터는 ‘어떤 지점’에서 ‘어떤 방향’을 가리키고 있는지 정확히 기술하라.

(k) 문맥상 84번 문장의 "here" 부분에 들어갈 내용을 OpenGL ES 쉐이딩 언어 문법에 맞게 정확히 기술하라.

(l) 문맥상 106번 문장의 "there 1"과 "there 2" 부분에 들어갈 내용을 OpenGL ES 쉐이딩 언어 문법에 맞게 정확히 기술하라.

(m) 다음은 NVIDIA사에서 개발한 Cg(C for Graphics)라는 쉐이딩 언어를 사용하여 구현한 버텍스 쉐이더 및 퍼셀 쉐이더 코드이다 (이하 Cg 쉐이더 코드). 참고로 이 예에서는 원근 투영과 점광원을 사용하고 있음.

```
struct _output {
    float4 X: TEXCOORD0;
    float3 Y: TEXCOORD1; // (0)
    float4 position: POSITION;
};

_output main(float4 A: POSITION,
            float4 B: NORMAL,
            uniform float4x4 ModelViewProj:
                state.matrix.mvp,
            uniform float4x4 ModelView:
                state.matrix.modelview,
            _output OUT;

    OUT.X = mul(ModelView, A);
    OUT.Y = mul(ModelView, B).xyz; // (1)
    OUT.position = mul(ModelViewProj, A); // (2)
    return OUT;
}
```

```
struct _output { float3 color: COLOR; };
_output main(float4 X: TEXCOORD0,
            float3 Y: TEXCOORD1,
            uniform float4 ga:
                state.lightmodel.ambient,
            uniform float4 ppp: ???, // (3)
            uniform float4 la:
                state.light[0].ambient,
            uniform float4 ld:
                state.light[0].diffuse,
            uniform float4 ls:
                state.light[0].specular,
            uniform float4 ma:
                state.material.ambient,
            uniform float4 md:
                state.material.diffuse,
            uniform float4 ms:
                state.material.specular,
            uniform float4 me:
                state.material.shininess) {
    _output OUT;
    OUT.color = ga * ma;
}
```

```

float3 N = normalize(Y);
float3 L = normalize(ppp - X);
float NdotL = dot(N, L);
OUT.color += ma * la;
if(NdotL >= 0.0) { // (4)
    float3 What1 = normalize(______); // (5)
    float3 What2 = normalize(L + What1); //
(6)
    float NdotWhat2 = dot(N, What2);
    float4 li = lit(NdotL, NdotWhat2, me);
    float3 D = md * li.y;
    float3 S = ms * li.z;
    OUT.color += (8);
}
return OUT;
}

```

이때 문맥상 Cg 쉐이더 코드의 (8)번에 들어갈 내용을 Cg 언어 문법에 맞게 정확히 기술하라.

— 문제 끝 —

이 Cg 쉐이더 코드의 (1) 번 문장에서 꼭지 점의 법선 벡터 (normal vector)를 OpenGL ES 쉐이더에서와는 다른 방식으로 변환을 하고 있다. 이 프로그램은 기하 물체에 적용이 되는 “어떤 기하 변환”이 “어떤 성질을 가지고 있을 때” 이 프로그램이 정상적으로 작동을 할까?

- (n) Cg 쉐이더 코드의 (2) 번 문장의 변수 OUT.position에 대응되는 OpenGL ES 쉐이더 변수를 기술하라.
- (o) 문맥 상 Cg 쉐이더 코드의 (3) 번 문장이 정의하는 변수 ppp는 정확히 OpenGL의 “어떤 좌표계”에서 “어떤 좌표값”을 저장하고 있어야 하나?
- (p) 문맥 상 Cg 쉐이더 코드의 (5) 번 문장의 normalize() 함수의 인자를 Cg 언어 문법에 맞게 정확히 기술하라.
- (q) 문맥상 그림 1(b)의 조명 공식에서 Cg 쉐이더 코드의 (6) 번 문장의 What2 변수에 저장되는 값에 대응되는 변수 (또는 수식)를 기술하라.
- (r) Cg 쉐이더 코드의 (6) 번 문장의 What2 변수가 저장하는 벡터를 통칭 무엇이라고 하는가?
- (s) Cg 쉐이더 코드에서 lit() 함수는 다음과 같은 내용의 계산을 수행한다.

```

float4 lit(float NdotL, float
NdotH, float m) {
    float specular = (NdotL > 0) ?
        pow(max(0.0, NdotH), m): 0.0;
    return float4(1.0, max(0.0,

```

```

1 // Begin of Vertex Shader
2 attribute vec4 a_position;
3 attribute vec3 a_normal;
4 attribute vec2 a_texture;
5
6 uniform mat4 mvp_matrix;
7 uniform mat4 modelview_matrix;
8 uniform mat4 invtrans_modelview_matrix;
9
10 varying vec4 v_position;
11 varying vec2 v_texture;
12 varying vec3 v_normal;
13
14 const float c_zero = 0.0;
15 const float c_one = 1.0;
16
17 void main() {
18     v_position = modelview_matrix * a_position;
19     v_normal = normalize(invtrans_modelview_matrix
20                           * vec4(a_normal, c_zero)).xyz;
21     v_texture = a_texture;
22
23     gl_Position = mvp_matrix * a_position;
24 }
25 // End of Vertex Shader
26
27 /////////////////////////////////
28
29 // Begin of Fragment Shader
30 precision mediump float;
31
32 varying vec4 v_position;
33 varying vec2 v_texture;
34 varying vec3 v_normal;
35
36 uniform vec4 material_ambient;
37 uniform vec4 material_diffuse;
38 uniform vec4 material_specular;
39 uniform float material_specular_exponent;
40 uniform vec4 light_position;
41 uniform vec4 light_ambient;
42 uniform vec4 light_diffuse;
43 uniform vec4 light_specular;
44 uniform vec3 spot_direction;
45 uniform float spot_exponent;
46 uniform float spot_cutoff_angle;
47 uniform vec4 global_ambient;
48 uniform vec3 distance_attenuation_factors;
49 uniform int distance_attenuation_flag;
50 uniform sampler2D u_Texture;
51 uniform int tex_flag;
52
53 const float c_zero = 0.0;
54 const float c_one = 1.0;
55
56 vec3 N_eye;
57
58 vec4 lighting_equation() {
59     vec4 computed_color = vec4(c_zero, c_zero, c_zero, c_zero);
60     vec3 H;
61     float ndotl;
62     float ndoth;
63     float att_factor;
64     float spot_factor;
65     vec3 att_dist;
66     vec3 L;

```

```

67 att_factor = c_one;
68 computed_color += material_ambient * global_ambient;
69
70 if (light_position.w != c_zero) {
71     L = light_position.xyz - v_position.xyz;
72     if (distance_attenuation_flag == 1) {
73         att_dist.x = c_one;
74         att_dist.z = dot(L, L);
75         att_dist.y = sqrt( );
76         att_factor = c_one/dot(att_dist ,
77                                 distance_attenuation_factors);
78     }
79     L = normalize(L);
80
81     if (spot_cutoff_angle < 180.0) {
82         vec3 spot_dir = normalize(spot_direction);
83         spot_factor = "here";
84         if (spot_factor >= cos(radians(spot_cutoff_angle))) {
85             spot_factor = pow(spot_factor, spot_exponent);
86         }
87         else {
88             spot_factor = c_zero;
89         }
90         att_factor *= spot_factor;
91     } // if (spot_cutoff_angle < 180.0)
92 } // if (light_position.w != c_zero)
93 else {
94     L = light_position.xyz;
95     L = normalize(L);
96 }
97
98 if (att_factor > c_zero) {
99     computed_color += light_ambient * material_ambient;
100
101     ndotL = max(c_zero, dot(N_eye, L));
102     computed_color += (ndotL*light_diffuse*material_diffuse);
103
104     H = normalize("there 1" - normalize("there 2").xyz);
105     ndoth = max(c_zero, dot(N_eye, H));
106     if (ndoth > c_zero) {
107         computed_color += (pow(ndoth, material_specular_exponent)
108                             * material_specular * light_specular);
109     }
110     computed_color *= att_factor;
111 }
112
113 return computed_color;
114 }
115 }
116
117 void main() {
118     vec4 color;
119
120     N_eye = normalize(v_normal);
121
122     color = lighting_equation();
123
124     if (tex_flag == 1) {
125         vec4 decalcolor = texture2D(u_Texture, v_texture.xy);
126         gl_FragColor = color * decalcolor;
127     }
128     else
129         gl_FragColor = color;
130
131 }
132 // End of Fragment Shader

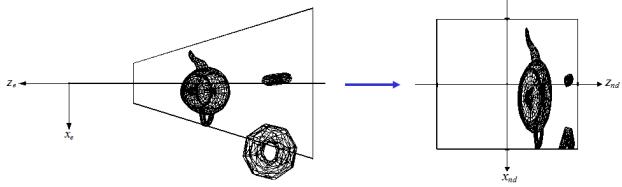
```

[CSE4170: 기초 컴퓨터 그래픽스]

기말고사

(담당교수: 임인성)

- 답은 연습지가 아니라 답안지에 기술할 것.
 - 답안지 공간이 부족할 경우, 답안지 뒷면에 기술하고, 해당 답안지 칸에 그 사실을 명기할 것.
1. 다음은 OpenGL fixed-function 렌더링 파이프라인에 관한 문제이다. 각 문제에 대하여 그림 5를 보면서, 해당하는 상자 이름 ((A), (B), (C), ...) 을 사용하여 정확하게 답하라.
- 삼각형의 세 꼭지점의 회전 방향에 따라 그 삼각형이 눈에 보이는 앞면에 해당하는지 아닌지를 결정하는 지점은?
 - 다음 그림이 암시하는 렌더링 계산은 어느 지점에서 어느 지점까지에 해당하는가?



- (c) 정상적인 렌더링 과정에서 다음과 같은 형태의 행렬이 빈번히 저장되는 지점은?

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

- 화면에 띠운 윈도우를 기준으로 한 좌표값은 어느 지점에서 비로소 생성될까?
- 정상적인 렌더링 과정 중 어떤 지점의 계산 결과 동차 좌표계의 네 번째 좌표 값, 즉 w 값이 1이 아닌 값이 처음으로 나타날 수 있을까?
- (위 문제에 이어) 어떤 지점의 계산 결과 w 좌표가 다시 1 값을 가지게 될까?
- 빛의 감쇠 효과가 생성되는 지점은?

(h) 정상적인 렌더링 과정에서 움직이는 자동차의 운전석에 카메라를 배치하는 기하 변환이 적용되는 지점은?

- 어느 두 지점 사이에서 꼭지점의 좌표값이 항상 -1과 1사이의 값만 가질까?
- 일종의 눈속임 기법인 무한 관찰자(infinite viewer) 기능이 적용되는 지점은?
- 어느 지점 직후부터 꼭지점의 좌표값이 카메라를 기준으로 한 값으로 바뀔까?
- 버텍스 쉐이더는 이 렌더링 파이프라인의 시작 지점부터 어느 지점까지의 계산을 대치할까?

2. 그림 1(a)에 주어진 OpenGL 조명 공식을 보면서 답하라.

- OpenGL fixed-function 렌더링 파이프라인에서는 이 공식이 WC, NDC, CC, WdC, MC, EC 중 어떤 좌표계에서 적용되는가?
- 쉐이딩하려는 물체를 구성하는 물질의 빛의 반사 성질을 표현하는 변수를 모두 기술하라.
- 는 두 벡터간의 변형된 내적 연산자를 의미하는데, 일반적인 내적 계산과 어떤 차이가 있는가?
- 이 공식에서 **0** 벡터가 아닌 임의의 벡터 **v**에 대해 \hat{v} 는 무엇을 의미하는가?
- 정반사 방향, 즉 정반사 물질이 입사 광선을 가장 강하게 반사시키는 방향을 이 공식의 변수들을 사용하여 표현하라.
- (문제 삭제)
- 램버트의 코사인 법칙을 표현해주는 수식을 정확히 기술하라.
- 다음 그림(네 개의 물주전자)의 렌더링 효과는 이 공식의 어떤 변수를 사용하여 조절한 것일까?

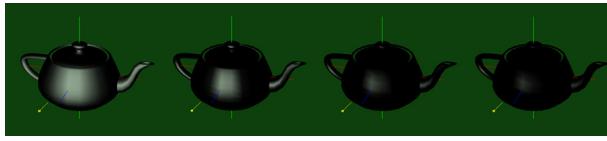
$$\mathbf{c} = \mathbf{e}_{cm} + \mathbf{a}_{cm} * \mathbf{a}_{cs} + \sum_{i=0}^{n-1} (att_i)(spot_i)[\mathbf{a}_{cm} * \mathbf{a}_{cli} + (\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli})\mathbf{d}_{cm} * \mathbf{d}_{cli} + (f_i)(\mathbf{n} \odot \hat{\mathbf{h}}_i)^{s_{rm}}\mathbf{s}_{cm} * \mathbf{s}_{cli}]$$

(a) OpenGL 조명 공식

$$spot_i = \begin{cases} (\overrightarrow{\mathbf{P}_{pli}\mathbf{V}} \odot \hat{\mathbf{s}}_{cli})^{s_{rl}}, & \text{if } c_{rl} \neq 180.0 \& \overrightarrow{\mathbf{P}_{pli}\mathbf{V}} \odot \hat{\mathbf{s}}_{cli} \clubsuit, \\ \alpha, & \text{if } c_{rl} \neq 180.0 \& \overrightarrow{\mathbf{P}_{pli}\mathbf{V}} \odot \hat{\mathbf{s}}_{cli} \spadesuit, \\ \beta, & \text{if } c_{rl} = 180.0 \end{cases}$$

(b) 스폿 광원 효과

Figure 1: OpenGL 조명 공식 관련 수식



- (i) 아래 식은 해프웨이 벡터(halfway vector)를 계산하는 과정을 기술하고 있다.

$$\mathbf{h}_i = \begin{cases} \overrightarrow{\mathbf{VP}_{pli}} + \overrightarrow{\mathbf{VP}}_e, & v_{bs} = \text{TRUE}, \\ \overrightarrow{\mathbf{VP}_{pli}} + (a \ b \ c \ d)^t, & v_{bs} = \text{FALSE} \end{cases}$$

여기서 \mathbf{P}_e 의 동차좌표를 기술하라.

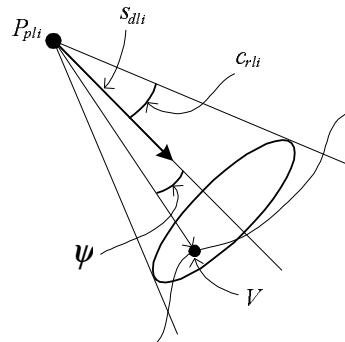
- (j) (위 문제에 이어) 지금 쉐이딩하려는 점 \mathbf{V} 의 동차좌표가 $(0 \ 0 \ -100 \ 1)^t$ 이라 할 때, 벡터 $\overrightarrow{\mathbf{VP}}_e$ 의 동차좌표를 기술하라.
- (k) (위 문제에 이어) 무한 관찰자(infinite viewer) 기능을 사용한다고 할 때, a, b, c , 그리고 d 값을 기술하라.
- (l) (위 문제에 이어) 계산효율 관점에서 볼 때, 해프웨이 벡터의 사용이 효과를 보려면 우선 무한 관찰자 기능을 사용(또는 직교 투영을 사용)해야 한다. 이와 함께 만족해야 할 추가적인 조건은 무엇인지 기술하라.
- (m) 다음 그림은 직접 조명에 의한 빛의 반사 효과를 생성하는 과정을 보여주고 있다. OpenGL 조명 공식에서 가장 왼쪽 그림의



렌더링 효과에 해당하는 수식을 정확히 기술하라(광원은 n 개가 있다고 가정하며, 필요하다면 Σ 기호외에 $\mathbf{e}_{cm}, (att_i), (spot_i)$, 그리고 (f_i) 와 같은 변수들을 사용할 것).

(n) (위 문제에 이어) 이 공식에서 (근사적으로 나마) 간접 조명 효과를 생성해주는 수식을 정확히 기술하라(광원은 n 개가 있다고 가정하며, 필요하다면 Σ 기호외에 $\mathbf{e}_{cm}, (att_i), (spot_i)$, 그리고 (f_i) 와 같은 변수들을 사용할 것).

(o) 그림 1(b)에는 스폿 광원 효과 생성을 위한 공식이 주어져 있는데, 다음 그림을 보면서 답하라.



OpenGL 시스템에서는 절단 각도 c_{rl} 는 디폴트 값으로 180.0이 설정되어 있으며, 이는 사방으로 빛을 방출하는 일반적인 점 광원을 사용한다는 것을 의미한다. 위 식이 올바르게 스폿 광원 효과를 생성할 수 있도록, \clubsuit 와 \spadesuit 에 들어갈 수식을 정확히 표현하라.

- (p) (위 문제에 이어) 이 공식에서 α 와 β 값을 기술하라.
- (q) (위 문제에 이어) 이 공식에서 $\cos \psi$ 값과 가장 관련이 깊은 수식을 기술하라.
- (r) (위 문제에 이어) 이 공식에서 s_{rl} 를 통하여 어떤 광원 효과를 조절할 수 있을까?

3. 다음은 광원의 기하 성질의 설정에 관한 문제이다. 그림 2에는 세 가지 서로 다른 형태의 광원이 사용되고 있다. A 광원은 세상 좌표계에서 임의로 돌아다니고 있는 빨간 소의 눈에서 나오는 레이저 형태의 광원이고, B 광원은, 역시 세상 좌표계에서 임의로 돌아다니고 있는, 관찰자, 즉 카메라의 바로 위에 고정된 스포트 광원이며, 마지막으로 C 광원은 세상 좌표계의 한 지점에 고정된 가로등에 설치된 조명이다.

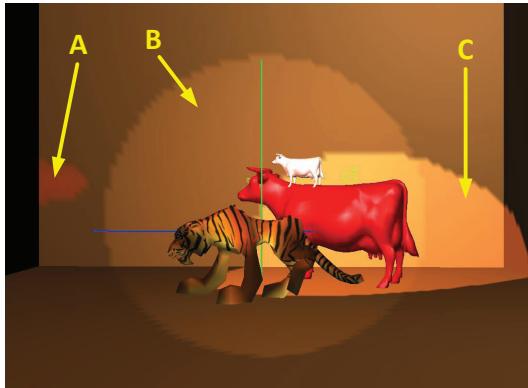


Figure 2: 광원 기하 성질의 설정

- (a) OpenGL fixed-function 렌더링 파이프라인을 통한 정상적인 렌더링 상황에서 A 광원의 기하 성질을 설정하는 시점에 모델뷰 행렬 스택의 탑에 있는 기하변환 행렬은 (WC, NDC, CC, WdC, MC, EC 등의 좌표계를 고려할 때) 어떤 좌표계에서 어떤 좌표계로 보내주는 행렬이어야 할까?

- (b) C 광원에 대해 위 문제를 풀어라.
 (c) 아래 코드는 그림 2를 그려주는 디스플레이 컬백 함수의 일부를 보여주고 있다. 정상적인 렌더링 상황에서 이 함수에서 정의되는 세 개의 광원, 즉 GL_LIGHT0, GL_LIGHT1, 그리고 GL_LIGHT2는 위의 세 광원과 어떻게 자연스럽게 대응이 될까?

```
void display(void) {
    glClear(GL_COLOR_BUFFER_BIT |
        GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glLightfv(GL_LIGHT0, GL_POSITION,
        light0_pos);
    :
    glMultMatrixf(cam.mat);
    glTranslatef(-cam.pos[X],
        -cam.pos[Y], -cam.pos[Z]);
    glLightfv(GL_LIGHT1, GL_POSITION,
```

```
        light1_pos);
    :
    glPushMatrix();
    glRotatef(angle, 0.0, 1.0, 0.0);
    glTranslatef(0.0, 0.0, 5.0);
    glScalef(6.5, 6.5, 6.5);
    :
    glLightfv(GL_LIGHT2, GL_POSITION,
        light2_pos);
    :
    glPopMatrix();
    glutSwapBuffers();
}
```

4. 다음은 카메라의 설정에 관한 문제이다. 그림 3(b)에는 운전석의 바로 앞에서 자동차 안을 바라보도록 카메라를 배치한 모습을 보여주고 있다(영화에서 앞 좌석의 두 주인공을 정면으로 들여다보는 상황을 상상할 것). 이를 위하여 자동차 몸체의 모델링 좌표계의 좌표축과 일치되어 있던 카메라 프레임을 y 축 둘레로 -90.0도 회전시킨 후, x 축 방향으로 -6.0만큼, 그리고 y 축 방향으로 1.2만큼 이동시켜 카메라를 배치하였다. 또한 그림 3(a)는 자동차 몸체(그리고 카메라 또한)를 y 축 둘레로 153.0도 회전시킨 후, x 축 방향으로 8.20만큼, y 축 방향으로 4.89만큼, 그리고 z 축 방향으로 1.95만큼 이동시켜 자동차와 카메라를 세상 좌표계에 배치한 모습을 도시하고 있다(본 과목의 관례대로 좌표축의 뺄강, 초록, 파랑 세 가지 색깔은 각각 x , y , z 축을 나타냄).

- (a) 다음 함수는 그림 3(c)를 그려주는 디스플레이 컬백 함수이다. 정상적인 렌더링 상황에서 (A) 부분에 들어갈 내용을 OpenGL API 함수 호출을 통하여 기술하라(이에 필요한 OpenGL 함수의 정확한 문법은 본 시험지에서 파악할 수 있으므로, C 언어 문법에 맞게 정확하게 기술할 것).

```
void display(void) {
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    (B)
    draw_fences(); draw_ground();
    draw_axes();
    glPushMatrix();
    (A)
    draw_car(); // Draw car in MC.
    glPopMatrix();
}
```

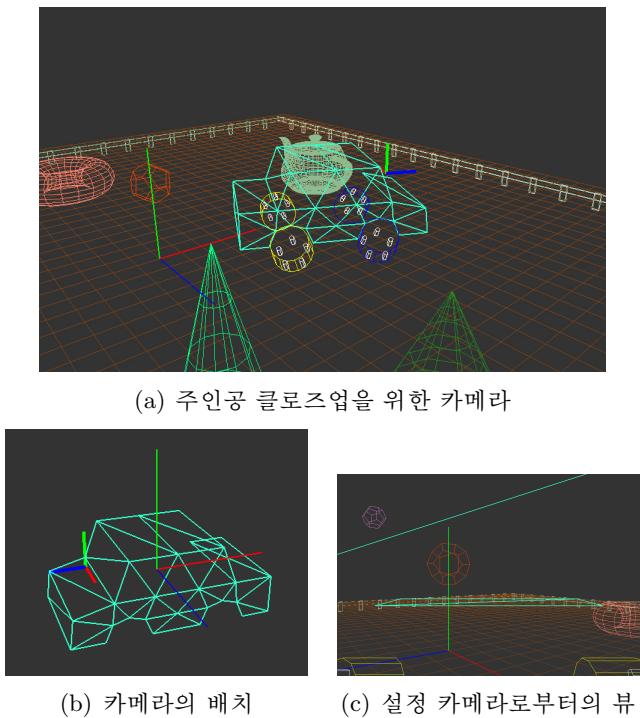


Figure 3: 카메라의 설정

- (b) (B) 부분에 대하여 위 문제를 풀어라.
5. 그림 4에는 OpenGL ES 2.0 Shading Language를 사용하여 풍 쇼이딩 방법을 구현한 버텍스 쉐이더와 프래그먼트 쉐이더의 예가 주어져 있는데, 이 예제 프로그램에서는 쇼이딩 계산에 필요한 기하 정보들을 EC로 변환하여 풍의 조명 모델 공식을 적용하고 있다.
- (a) 꼭지점에 적용된 모델링 변환, 뷰잉 변환, 그리고 투영 변환의 행렬을 각각 M_M , M_V , 그리고 M_P 라 하자. 문맥 상 버텍스 쉐이더에서 사용하는 uniform 탑입의 변수 $u_{Matrix0}$ 에는 어떤 행렬 값이 지정되어 있을지를 위 세 행렬을 사용하여 수학적으로 정확히 표현하라.
 - (b) $u_{Matrix1}$ 에 대하여 위 문제를 풀어라.
 - (c) $u_{Matrix2}$ 에 대하여 위 문제를 풀어라.
 - (d) 그림 5에서 버텍스 쉐이더의 varying 탑입의 변수들에 저장된 정보가 프래그먼트 쉐이더의 varying 탑입의 변수들에 전달되는 지점은?
 - (e) Line (a) 문장에서 v_N 벡터의 길이를 1로 만들어 사용하고 있는데, 이러한 연산을 해야하는 근본적인 이유는 무엇일까?
 - (f) 만약 Line (a) 문장을 제거하면 렌더링 결과 영상이 크게 바뀔까? ‘예/아니오’로 답하고 그 이유를 설명하라.

- (g) 문맥 상 uniform 탑입의 변수 $u_{PointLightPos}$ 의 값은 WC, NDC, CC, WdC, MC, 그리고 EC 중 어떤 좌표계를 기준으로 주어질까?
- (h) 프래그먼트 쉐이더에서 조명 계산 시 간접 조명을 극사적이나마 표현해주려 하는 문장을 정확히 기술하라.
- (i) Line (b)와 (c) 문장에서 잘못된 것이 있으면 모두 올바르게 고쳐라(문장 전체를 기술하고, 오류가 없다면 없다고 답할 것).
- (j) (오류가 있을 경우 수정하였다는 가정하에) Line (d) 문장을 `reflect()` 함수를 사용치 않고 적절한 변수를 사용하여 작성하라.
- (k) 프래그먼트 쉐이더의 `if (NdotL > c_zero)` 문장의 조건이 거짓이 되는 경우는 어떤 상황을 의미하는지 기술하라.
- (l) (오류가 있을 경우 수정하였다는 가정하에) 지금 정반사 방향 대신 해프웨이 벡터를 사용하여 쇼이딩을 하려한다. 이를 위해 이 프래그먼트 쉐이더의 어느 부분을 고쳐야 할지를 명시하고, OpenGL ES 2.0 Shadning Language 문법에 맞게 수정하라(반드시 해프웨이 벡터의 이름을 H로 할 것).

6. OpenGL ES 렌더링 파이프라인에서 삼각형으로 구성된 기하 모델에 대한 풍 쇼이딩 기법을 구현하는데 있어 다음 각 단계에서 수행되는(또는 수행되어야 하는) 중요한 계산을 기술하라.
- (a) 버텍스 쇼이딩 단계에 프로그래머가 해주어야 할 계산 두 가지
 - (b) 프래그먼트 쇼이딩 단계에서 프로그래머가 해주어야 할 계산
 - (c) 래스터화 단계에서 수행되는 계산 두 가지
7. OpenGL ES 렌더링 파이프라인에서 다음 두 단계에서 보편적으로 수행되는 전형적인 3D 실시간 렌더링 계산에 대하여 정확히 기술하라.
- (a) 래스터화 이전 부분
 - i. 쉐이더 동작 부분에서 반드시 수행되어야 하는 계산
 - ii. 그 이후의 주요 계산 모두
 - (b) 래스터화 이후 부분
 - i. 쉐이더 동작 부분에서 반드시 수행되어야 하는 계산
 - ii. 그 이후의 주요 계산 한 가지

```

uniform mat4 u_Matrix0;
uniform mat4 u_Matrix1;
uniform mat4 u_Matrix2;
attribute vec4 a_Vertex;
attribute vec3 a_Normal;
attribute vec2 a_TexCoord;
varying vec4 v_P;
varying vec3 v_N;
varying vec2 v_TexCoord;

void main(void) {
    v_P = u_Matrix0 * a_Vertex;
    v_N = normalize(mat3(u_Matrix1) * a_Normal);
    v_TexCoord = a_TexCoord;
    gl_Position = u_Matrix2 * a_Vertex;
}

```

(a) 베텍스 쉐이더

```

precision mediump float;
uniform vec4 u_PointLightPos;
uniform vec4 u_PointLightCol;
uniform vec4 u_AmbientLightCol;
uniform vec4 u_AmbientMat, u_DiffuseMat, u_SpecularMat;
uniform float u_SpecularPow; // Must be positive
uniform bool u_isTextureOn;
uniform sampler2D u_Texture;
varying vec4 v_P;
varying vec3 v_N;
varying vec2 v_TexCoord;
const float c_zero = 0.0;

void main(void) {
    vec3 N = normalize(v_N); // Line (a)
    vec3 L = normalize(u_PointLightPos.xyz); // Line (b)
    vec3 V = normalize(-v_P.xyz); // Line (c)
    vec3 R = reflect(-L, N); // Line (d)
    float NdotL = max(dot(N, L), c_zero);
    vec4 tmpCol = vec4(c_zero, c_zero, c_zero, c_zero);
    if (NdotL > c_zero) {
        tmpCol = u_DiffuseMat * NdotL;
        tmpCol += u_SpecularMat * pow(max(dot(R, V), c_zero), u_SpecularPow);
        tmpCol *= u_PointLightCol;
    }
    tmpCol += u_AmbientLightCol*u_AmbientMat;
    if (u_isTextureOn) {
        vec4 TextureCol = texture2D(u_Texture, v_TexCoord.xy);
        gl_FragColor = TextureCol*tmpCol;
    }
    else gl_FragColor = tmpCol;
}

```

(b) 프래그먼트 쉐이더

Figure 4: OpenGL ES 쉐이더를 통한 다면체 모델의 셰이딩

- 2014년 6월 16일(월) 오후 7:00 (AS 414) -

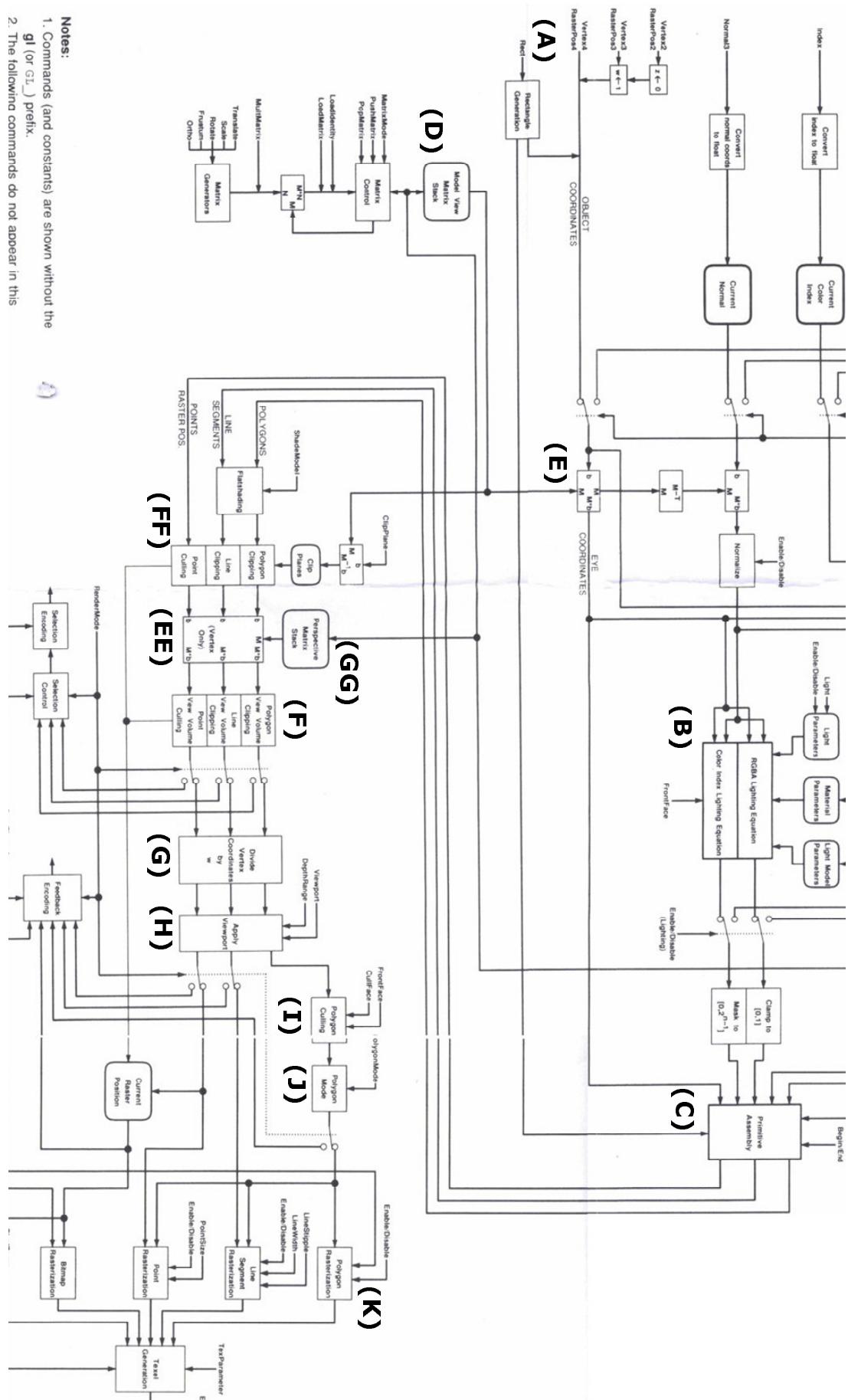


Figure 5: OpenGL fixed-function 렌더링 파이프라인의 일부

- 2014년 6월 16일(월) 오후 7:00 (AS 414) -

[CSE4170: 기초 컴퓨터 그래픽스]

기 말 고 사

(담당교수: 임 인 성)

- 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.
- OpenGL 시스템의 각 좌표계에 대한 약어는 다음과 같으며, 답을 기술할 때 필요할 경우 적절히 약어를 사용하라.

- WdC: Window Coordinates
- EC: Eye Coordinates
- NDC: Normalized Device Coordinates
- OC: Object Coordinates
- MC: Modeling Coordinates
- CC: Clip Coordinates
- WC: World Coordinates

1. 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();
glPushMatrix();
glRotatef(angle, 0.0, 1.0, 0.0);
glTranslatef(0.0, 0.0, 5.0);
glScalef(6.5, 6.5, 6.5);
// Line (d)
draw_cow();
:
glPopMatrix();
glFlush();
}
```

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

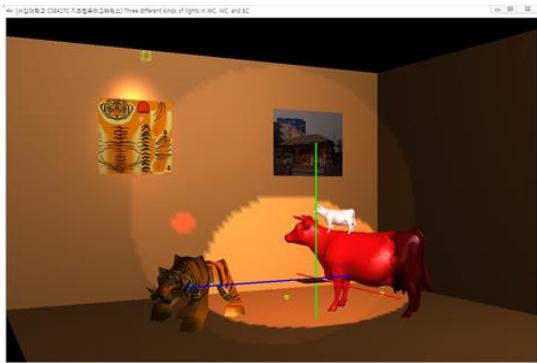


Figure 1: 광원의 기하 성질 설정

```

glVertex3f(150.0, 0.0, 0.0);
:
glEnd();
glLineWidth(1.0);
}

```

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

- (c) 그림 1에서 가장 큰 원으로 보이는 광원 효과는 카메라에 고정된 스폿 광원을 사용하여 생성한 것이다. 이 광원에 대한 기하 성질들은 Line (a), Line (b), Line (d) 중 어느 시점에서 설정하는 것이 가장 자연스러운가?
- (d) 그림 1에서 가장 작은 (빨간색) 원으로 보이는 광원 효과는 큰 소의 눈에 고정된 스폿 광원을 사용하여 생성한 것이다. 이 광원에 대한 기하 성질들은 Line (a), Line (b), Line (d) 중 어느 시점에서 설정하는 것이 가장 자연스러운가?
- (e) 그림 1에서 벽면 아래 쪽에 반원처럼 보이는 광원 효과는 세상 좌표계의 가로등에 고정된 스폿 광원을 사용하여 생성한 것이다. 이 광원에 대한 기하 성질들은 Line (a), Line (b), Line (d) 중 어느 시점에서 설정하는 것이 가장 자연스러운가?
- (f) 문맥상 `set_rotate_mat(cam.mat);` 문장은 `cam.mat[]` 배열에 어떤 값을 어떻게 넣어주는 역할을 할지 정확히 기술하라.
- (g) 그림 2는 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.

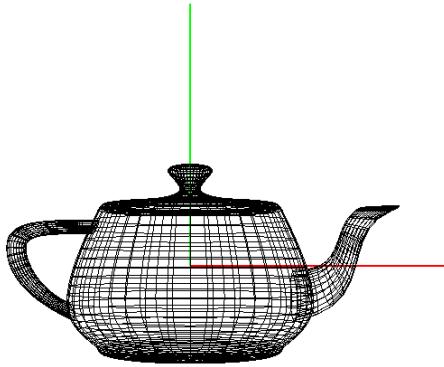


Figure 2: 물주전자 그리기

- (h) 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.

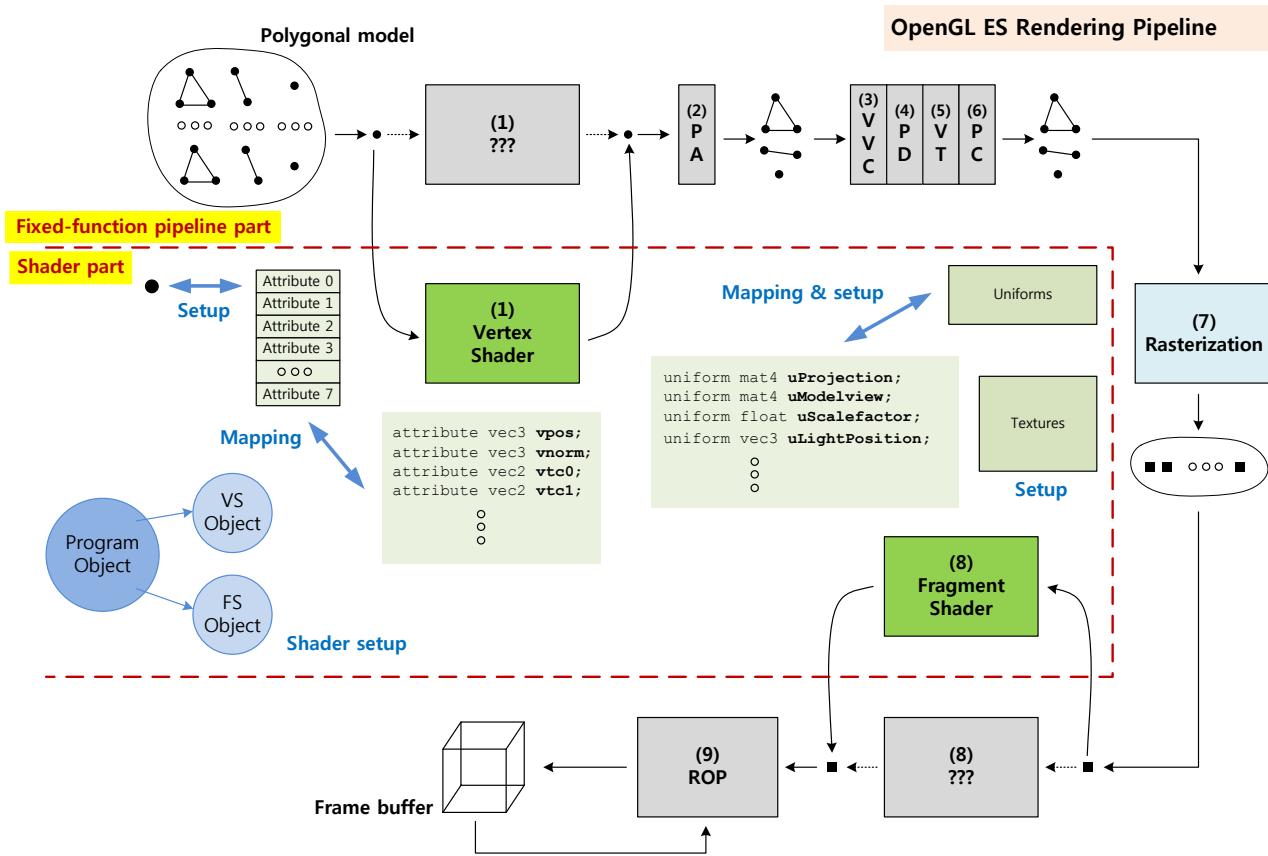


Figure 3: OpenGL ES 2.0 렌더링 파이프라인

2. 그림 3에 도시한 OpenGL ES 2.0 렌더링 파이프라인을 보면서 아래 질문에 답하라.
- 은면 제거를 위한 깊이 버퍼링 (depth buffering) 계산과 가장 관련이 깊은 단계의 번호는?
 - OC, MC, WC, EC, CC, NDC, WdC 등의 OpenGL 좌표계 중 (2)번 시점은 어떤 좌표계에 해당할까?
 - 삼선형 보간 (trilinear interpolation)과 가장 관련이 깊은 단계의 번호는?
 - 꼭지점의 나열 방향을 통하여 뒤면 (backface)이 보이는 삼각형을 제거할 수 있는데, 실제로 이러한 계산이 일어나는 단계의 번호는?
 - 3차원 퀘리움의 내용이 인화지에 인화가 되는 과정과 가장 관련이 많은 단계의 번호는?
 - 정상적인 렌더링 과정에서 모델링 변환과 뷔잉 변환이 일어나는 단계의 번호는?

- 이 렌더링 파이프라인에서 NDC는 몇 번 단계와 몇 번 단계 사이에 존재하는가?
- 다면체 모델에 대한 쉐이딩 방법 중의 하나인 풍 쉐이딩 방법을 구현하려 할 때, 가장 자연스럽게 풍의 조명 모델 공식이 적용될 수 있는 단계의 번호는?
- 정상적인 렌더링 과정에서 투영 변환이 일어나는 단계의 번호는?
- OpenGL ES 2.0 시스템의 `gl_Position` 변수와 가장 관련이 많은 단계의 번호는?
- 삼각형의 꼭지점에 설정된 여러 속성들이 퍽 셀들에 대한 속성으로 바뀌는 단계의 번호는?
- 원근 투영을 사용할 경우 실제로 원근감이 생성되는 과정과 가장 관련이 깊은 단계의 번호는?

$$\mathbf{c} = \mathbf{e}_{cm} + \mathbf{a}_{cm} * \mathbf{a}_{cs} + \sum_{i=0}^{n-1} (att_i)(spot_i)[\mathbf{a}_{cm} * \mathbf{a}_{cli} + (\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli})\mathbf{d}_{cm} * \mathbf{d}_{cli} + (f_i)(\mathbf{n} \odot \hat{\mathbf{h}}_i)^{s_{rm}}\mathbf{s}_{cm} * \mathbf{s}_{cli}]$$

(a) 기본 조명 공식

$$att_i = \begin{cases} \frac{1}{k_{0i} + k_{1i}\|\overrightarrow{\mathbf{VP}}_{pli}\| + k_{2i}\|\overrightarrow{\mathbf{VP}}_{pli}\|^2}, & \mathbf{P}_{pli}'s w \neq 0, \\ 1.0, & \text{otherwise} \end{cases}$$

(b) 빛의 감쇠 효과의 계산

$$spot_i = \begin{cases} (\overrightarrow{\mathbf{P}}_{pli}\overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{cli})^{s_{rl}}, & c_{rl} \neq 180.0 \& \overrightarrow{\mathbf{P}}_{pli}\overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{cli} \geq \cos c_{rl}, \\ 0.0, & c_{rl} \neq 180.0 \& \overrightarrow{\mathbf{P}}_{pli}\overrightarrow{\mathbf{V}} \odot \hat{\mathbf{s}}_{cli} < \cos c_{rl}, \\ 1.0, & c_{rl} = 180.0 \end{cases}$$

(c) 스폷 광원 효과의 계산

Figure 4: OpenGL 시스템의 조명 공식

3. 그림 4에는 OpenGL 시스템에서 사용하는 기본 조명 공식이 주어져 있다.

- (a) 이 공식에서 \odot 은 주어진 두 벡터에 대하여 어떠한 값을 계산해주는 연산자인지 정확히 기술하라.
- (b) 만약 무한 관찰자 (infinite viewer) 기능을 사용할 경우 이 공식의 $\hat{\mathbf{h}}_i$ 변수가 어떻게 정의가 될지 이 공식의 변수나 상수식 등을 사용하여 정확히 기술하라 (^ 기호의 의미를 잘 생각해볼것).
- (c) 만약 무한 관찰자 기능을 사용할 경우 추가적으로 어떤 조건 하에서 이 조명 공식에 기반을 둔 쉐이딩 계산을 어떻게 최적화할 수 있을까?
- (d) 그림 4(a)의 식에서 무한 관찰자 기능을 사용하지 않을 경우 카메라의 위치가 바뀌었을 때 직접적으로 영향을 받는 변수 (또는 식)를 모두 나열하라.
- (e) 그림 4(a)의 식에서 점 광원을 사용할 경우 광원의 위치가 바뀌었을 때 직접적으로 영향을 받을 수 있는 변수 (또는 식)들을 모두 나열하라.
- (f) 그림 4(a)의 식에서 렌더링 하고자 하는 물체의 위치와 방향이 모두 바뀌었을 때 직접적으로 영향을 받을 수 있는 변수 (또는 식)



들을 모두 나열하라.

- (g) 위 그림은 하이라이트 효과의 조절 모습을 도시하고 있는데, 그림 4(a)의 식에서 어떤 변수를 통하여 조절할 수 있을까?
- (h) 그림 4(a)의 식에서 램버트의 코사인 법칙 (Lambert's cosine law)을 표현해주는 부분을 정확히 기술하라.
- (i) 기본적으로 풍의 조명 모델은 광원으로부터 직접 입사하여 물체 표면에서 반사되는 빛의 색깔을 어떻게 계산할지를 기술하고 있다. 이 경우 다른 물체에 반사된 후 간접적으로 들어오는 빛을 무시함으로써 발생하는 문제를 조금이라도 줄이기 위하여 간접 반사를 고려하고 있다. 그림 4(a)의 식에서 간접적으로 들어오는 빛과 직접적인 연관성이 있는 변수 (또는 식)들을 모두 기술하라.
- (j) 그림 4(a)의 식에서 OpenGL API 함수인 `glLightfv` (`*`, `*`, `*`) 함수로 설정할 수 있는 변수 (또는 식)들을 모두 기술하라.

- (k) 그림 4(b)의 식에서 $\|VP_{pli}\|$ 는 어떤 기하 정보에 해당하는지 정확히 기술하라.
- (l) 그림 4(b)의 식에서 조건식 $P_{pli}'s \cdot w \neq 0$ 이 만족되는 상황은 어떤 경우인지 정확히 기술하라.
- (m) 그림 4(c)의 식에서 두 벡터 $\overrightarrow{P_{pli}V}$ 와 \hat{s}_{dli} 가 의미하는 방향을 정확히 기술하라.
- (n) 그림 4(c)의 식에서 조건식 $\overrightarrow{P_{pli}V} \odot \hat{s}_{dli} \geq \cos c_{rli}$ 이 만족되는 상황은 어떤 경우인지 정확히 기술하라.
- (o) 다면체 모델의 쉐이딩 기법인 고우러드 쉐이딩 방법을 적용한다고 할 때, 보편적으로 이 조명 공식은 그림 3의 어느 부분에서 적용이 될지, 가장 관련이 깊은 단계의 번호를 기술하라.
4. 그림 5는 Pixar의 RenderMan Shading Language로 작성한 쉐이더와 이를 통한 렌더링 결과를 도시하고 있다.
- (a) 문맥상 이 쉐이더 프로그램에서 물체의 뼈대의 굵기를 조절해주는 변수는 어떤 것일까?
 - (b) 문맥상 이 쉐이더 프로그램에서 가로-세로 방향의 뼈대의 개수를 조절해주는 변수는 어떤 것일까?
 - (c) OpenGL ES 2.0 Shading Language에서 제공하는 built-in name들 중 (C_i , O_i)와 가장 밀접한 연관이 있는 변수의 이름은 무엇일까?
5. 본 시험 문제지의 마지막 두 쪽에는 OpenGL ES 2.0 Shading Language를 사용하여 풍 쉐이딩 방법을 구현한 버텍스 쉐이더와 프래그먼트 쉐이더의 예가 주어져 있는데, 여기서는 쉐이딩 계산에 필요한 기하 정보들을 EC로 변환하여 풍의 조명 모델 공식을 적용하고 있다.
- (a) 이 버텍스 쉐이더의 계산 결과로 산출되는 결과 정보를 저장해주는 변수들을 모두 기술하라.
 - (b) 이 두 쉐이더의 변수들 중 일반적인 렌더링 과정에서 항상 반드시 계산을 해주어야 하는 변수가 있다. 그 변수 이름을 기술하고 (한 개 이상이면 모두), 정상적인 상황에서 그 변수는 어떤 정보를 표현해주어야 할까?
- (c) 문맥상 물체의 조명 모델 공식이 적용되는 물체 지점의 좌표를 저장하고 있는 프래그먼트 쉐이더의 변수 이름을 기술하라.
- (d) 6번 문장의 uniform 변수 `mvp_matrix`의 기하변환 행렬은 OpenGL 용어를 사용할 경우 어느 좌표계에서 어느 좌표계로의 변환에 해당하는 행렬을 저장하고 있어야 할까?
- (e) 만약 7번 문장의 uniform 변수 `modelview_matrix`의 기하변환 행렬이 항상 강체 변환에 해당한다면 8번 문장의 uniform 변수를 제거할 수 있다. 이때 버텍스 쉐이더의 몇 번 문장의 어느 부분을 어떻게 수정을 해야할까?
- (f) 10번 문장의 varying 변수 `v_position`의 저장하고 있는 정보가 32번 문장의 varying 변수 `v_position`으로 전달이 되어야 한다. OpenGL 렌더링 파이프라인에서 어떤 “중요한” 계산 과정의 수행 시 어떠한 방식으로 전달이 되는지 설명하라.
- (g) 프래그먼트 쉐이더의 71번 문장의 조건문에서 이 조건이 성립한다는 것은 무엇을 뜻할까?
- (h) 이 쉐이더 프로그램에서 문맥상 그림 4(a)의 조명 공식의 `acs` 변수에 해당하는 쉐이더 변수 이름을 기술하라.
- (i) 문맥상 그림 4(a)의 조명 공식에서 100번 문장 수행 시 `att_factor` 변수가 저장하고 있는 값과 가장 밀접한 관련이 있는 변수 (또는 식)를 정확히 기술하라.
- (j) 만약 평행 광원을 사용할 경우 `att_factor` 변수는 이 프로그램에서 어떤 값을 갖게 될까?
- (k) 문맥상 그림 4(a)의 조명 공식에서 103번 문장의 `max(c_zero, dot(N_eye, L))` 값에 해당하는 변수 (또는 식)를 정확히 기술하라.
- (l) EC에서 카메라의 기준점인 눈에 해당하는 점의 좌표는?
- (m) 106번 문장에서 사용되고 있는 `-normalize(v_position).xyz` 벡터는 어떤 방향을 가리키고 있는지 정확히 기술하라.

```

surface screen(float Ka = 1, Kd = 0.75, Ks = 0.4, AAA = 0.1;
              color BBB = 1; float CCC = 0.25, DDD = 20;) {
    point Nf;

    if (mod(s*DDD, 1) < CCC || mod(t*DDD, 1) < CCC) {
        Oi = 1;
        Nf = faceforward(normalize(N), I);
        Ci = Os*(Cs*(Ka*ambient() + Kd*diffuse(Nf))
                  + BBB*Ks*specular(Nf,-normalize(I), AAA));
    }
    else {
        Oi = 0;
        Ci = 0;
    }
}

```

(a) Screen 쉐이더



(b) 렌더링 결과

Figure 5: RenderMan 쉐이더 예

(n) 문맥상 84번 문장의 "here" 부분에 들어갈 내용을 위의 쉐이딩 언어 문법에 맞게 정확히 기술하라.

(o) 문맥상 74번부터 78번 문장까지는 그림 4(b)의 식을 구현하고 있다. 문맥상 75번 문장의 "there" 부분에 들어갈 내용을 위의 쉐이딩 언어 문법에 맞게 정확히 기술하라.

(p) (제거해도 큰 문제는 없지만) 120번 문장에서 normalize(v_normal) 함수 호출을 통하여 v_normal 벡터의 길이를 1로 만들어 주는 이유는?

(q) 문맥상 그림 4(a)의 조명 공식에서 120번 문장이 수행된 후의 N_eye 변수 값에 해당하는 변수 (또는 식)를 정확히 기술하라.

<한 학기 동안 수고 많았습니다!>

```

1 // Begin of Vertex Shader
2 attribute vec4 a_position;
3 attribute vec3 a_normal;
4 attribute vec2 a_texture;
5
6 uniform mat4 mvp_matrix;
7 uniform mat4 modelview_matrix;
8 uniform mat4 invtrans_modelview_matrix;
9
10 varying vec4 v_position;
11 varying vec2 v_texture;
12 varying vec3 v_normal;
13
14 const float c_zero = 0.0;
15 const float c_one = 1.0;
16
17 void main() {
18     v_position = modelview_matrix * a_position;
19     v_normal = normalize(invtrans_modelview_matrix
20                           * vec4(a_normal, c_zero)).xyz;
21     v_texture = a_texture;
22
23     gl_Position = mvp_matrix * a_position;
24 }
25 // End of Vertex Shader
26
27 /////////////////////////////////
28
29 // Begin of Fragment Shader
30 precision mediump float;
31
32 varying vec4 v_position;
33 varying vec2 v_texture;
34 varying vec3 v_normal;
35
36 uniform vec4 material_ambient;
37 uniform vec4 material_diffuse;
38 uniform vec4 material_specular;
39 uniform float material_specular_exponent;
40 uniform vec4 light_position;
41 uniform vec4 light_ambient;
42 uniform vec4 light_diffuse;
43 uniform vec4 light_specular;
44 uniform vec3 spot_direction;
45 uniform float spot_exponent;
46 uniform float spot_cutoff_angle;
47 uniform vec4 global_ambient;
48 uniform vec3 distance_attenuation_factors;
49 uniform int distance_attenuation_flag;
50 uniform sampler2D u_Texture;
51 uniform int tex_flag;
52
53 const float c_zero = 0.0;
54 const float c_one = 1.0;
55
56 vec3 N_eye;
57
58 vec4 lighting_equation() {
59     vec4 computed_color = vec4(c_zero, c_zero, c_zero, c_zero);
60     vec3 H;
61     float ndotl;
62     float ndoth;
63     float att_factor;
64     float spot_factor;
65     vec3 att_dist;
66     vec3 L;
67
68     att_factor = c_one;
69     computed_color += material_ambient * global_ambient;
70
71     if (light_position.w != c_zero) {

```

```

72     L = light_position.xyz - v_position.xyz;
73     if (distance_attenuation_flag == 1) {
74         att_dist.x = c_one;
75         att_dist.z = "there";
76         att_dist.y = sqrt(att_dist.z);
77         att_factor = c_one/dot(att_dist,
78                                 distance_attenuation_factors);
79     }
80     L = normalize(L);
81
82     if (spot_cutoff_angle < 180.0) {
83         vec3 spot_dir = normalize(spot_direction);
84         spot_factor = "here";
85         if (spot_factor >= cos(radians(spot_cutoff_angle))) {
86             spot_factor = pow(spot_factor, spot_exponent);
87         }
88         else {
89             spot_factor = c_zero;
90         }
91         att_factor *= spot_factor;
92     } // if (spot_cutoff_angle < 180.0)
93 } // if (light_position.w != c_zero)
94 else {
95     L = light_position.xyz;
96     L = normalize(L);
97 }
98
99     if (att_factor > c_zero) {
100        computed_color += light_ambient * material_ambient;
101
102        ndotL = max(c_zero, dot(N_eye, L));
103        computed_color += (ndotL*light_diffuse*material_diffuse);
104
105        H = normalize(L - normalize(v_position).xyz);
106        ndoth = max(c_zero, dot(N_eye, H));
107        if (ndoth > c_zero) {
108            computed_color += (pow(ndoth, material_specular_exponent)
109                                * material_specular * light_specular);
110        }
111        computed_color *= att_factor;
112    }
113 }
114 return computed_color;
115 }
116
117 void main() {
118     vec4 color;
119
120     N_eye = normalize(v_normal);
121
122     color = lighting_equation();
123
124     if (tex_flag == 1) {
125         vec4 decalcolor = texture2D(u_Texture, v_texture.xy);
126         gl_FragColor = color * decalcolor;
127     }
128     else
129         gl_FragColor = color;
130 }
131 }
132 // End of Fragment Shader

```

[CSE4170] 기초 컴퓨터 그래픽스 기 말 고 사

담당 교수: 임 인 성

- 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.
- OpenGL 시스템의 각 좌표계에 대한 약어는 다음과 같으며, 답을 기술할 때 필요할 경우 적절히 약어를 사용하라.

WdC: Window Coordinates

EC: Eye Coordinates

NDC: Normalized Device Coordinates

OC: Object Coordinates

MC: Modeling Coordinates

CC: Clip Coordinates

WC: World Coordinates

- 아래의 그림 1은 본 시험 전반에 걸쳐 언급될 수 있으니 해당 문제에 대하여 적절히 답하라.

(1) vertex stream → (2) Vertex Shader → (3) vertex stream → (4) Primitive Assembly → (5) primitive stream → (6) Geometry Shader → (7) primitive stream → (8) Clipping & Setup → (9) Rasterization → (10) pixel stream → (11) Pixel Shader → (12) pixel stream → (13) Raster Operation → (14) Framebuffer

Figure 1: 실시간 렌더링 파이프라인 예

1. 그림 1에 주어진 실시간 렌더링 파이프라인의 계산 과정에 대한 질문에 답하라.

- (3)번 지점은 OpenGL 좌표계 중 어떤 좌표계에 해당할까?
- 보편적인 상황에서 모델링 변환과 뷰잉 변환이 일어나는 단계의 번호는?
- 전통적인 OpenGL 파이프라인에서 은면 제거, 즉 안보이는 면이 제거되는 계산과 가장 관련이 깊은 단계의 번호는?

(d) 꼭지점의 좌표에 투영 변환 행렬이 곱해지는 단계의 번호는?

(e) 꼭지점의 나열 순서에 따라 다각형을 제거할 수도 있는 단계의 번호는?

(f) 베텍스 쉐이더의 속성 데이터가 대응되는 프래그먼트 쉐이더의 속성 데이터로 변환되는 단계의 번호는?

(g) 투영 변환 행렬이 암시하는 뷰 볼륨의 밖에 존재하는 기하 프리미티브들이 제거가 되는 단계의 번호는?

(h) 다면체 모델에 대한 쉐이딩 방법 중의 하나인 풍 쉐이딩 방법을 구현하려 할 때, 실제로 풍의 조명 모델 공식의 계산이 가장 자연스럽게 수행될 수 있는 단계의 번호는?

(i) 원근 나눗셈과 가장 관련이 깊은 단계의 번호는?

2. 다음 쪽 윗 부분의 그림 2(a)에는 OpenGL 시스템에서 사용하는 기본 조명 공식이 주어져 있다.

(a) 이 공식에서 ○은 주어진 두 벡터에 대하여 어떠한 값을 계산해주는 연산자인지 정확히 기술하라.

(b) 이 공식에서 Lambert의 코사인 법칙을 표현해주는 부분을 정확히 기술하라.

(c) 만약 무한 관찰자 (infinite viewer) 기능을 사용할 경우 이 공식의 \hat{h}_i 변수가 어떻게 정의 될지 이 공식의 변수나 상수식 등을 사용하여 정확히 기술하라 (^ 기호의 의미를 상기할 것).

(d) 만약 무한 관찰자 기능을 사용할 경우 추가적으로 어떤 조건 하에서 이 조명 공식에 기반을 둔 쉐이딩 계산을 어떻게 최적화할 수 있을까?

(e) 무한 관찰자 기능을 사용하지 않을 경우 카메라의 위치가 바뀌었을 때 이 공식에서 직접적으로 영향을 받는 변수 (또는 식)를 모두 나열하라.

$$\mathbf{c} = \mathbf{e}_{cm} + \mathbf{a}_{cm} * \mathbf{a}_{cs} + \sum_{i=0}^{n-1} (att_i)(spot_i)[\mathbf{a}_{cm} * \mathbf{a}_{cli} + (\mathbf{n} \odot \overrightarrow{\mathbf{VP}}_{pli}) \mathbf{d}_{cm} * \mathbf{d}_{cli} + (f_i)(\mathbf{n} \odot \hat{\mathbf{h}}_i)^{s_{rm}} \mathbf{s}_{cm} * \mathbf{s}_{cli}]$$

(a) OpenGL 시스템의 조명 공식

$$att_i = \begin{cases} \frac{1}{k_{0i} + k_{1i}\|\overrightarrow{\mathbf{VP}}_{pli}\| + k_{2i}\|\overrightarrow{\mathbf{VP}}_{pli}\|^2}, & \mathbf{P}_{pli}'s w \neq 0, \\ 1.0, & \text{otherwise} \end{cases}$$

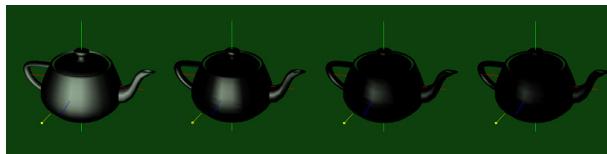
(b) 빛의 감쇠 효과의 계산

$$spot_i = \begin{cases} (\overrightarrow{\mathbf{P}_{pli}\mathbf{V}} \odot \hat{\mathbf{s}}_{cli})^{s_{rl}}, & c_{rl} \neq 180.0 \& \overrightarrow{\mathbf{P}_{pli}\mathbf{V}} \odot \hat{\mathbf{s}}_{cli} \geq \cos c_{rl}, \\ 0.0, & c_{rl} \neq 180.0 \& \overrightarrow{\mathbf{P}_{pli}\mathbf{V}} \odot \hat{\mathbf{s}}_{cli} < \cos c_{rl}, \\ 1.0, & c_{rl} = 180.0 \end{cases}$$

(c) 스폷 광원 효과의 계산

Figure 2: 2번 문제 공식

- (f) 점 광원을 사용할 경우 광원의 위치가 바뀌었을 때 이 공식에서 직접적으로 영향을 받을 수 있는 변수 (또는 식)들을 모두 나열하라.
- (g) 렌더링 하고자 하는 물체의 위치와 방향이 모두 바뀌었을 때 이 공식에서 직접적으로 영향을 받을 수 있는 변수 (또는 식)들을 모두 나열하라.
- (h) 아래의 그림은 하이라이트 효과의 조절 모습을 도시하고 있는데, 이 공식의 어떤 변수를 통하여 조절할 수 있을까?



- (i) 기본적으로 풍의 조명 모델은 광원으로부터 직접 입사하여 물체 표면에서 반사되는 빛의 색깔을 어떻게 계산할지를 기술하고 있다. 이 경우 다른 물체에 반사된 후 간접적으로 들어오는 빛을 무시함으로써 발생하는 문제를 조금이라도 줄이기 위하여 간접 반사를 고려하고 있다. 이 공식에서 간접적으로 들어오는 빛과 직접적인 연관성이 있는 변수 (또는 식)들을 모두 기술하라.
- (j) 이 공식에서 OpenGL API 함수인

`glMaterial{if}[v](*, *, *)` 함수로 설정할 수 있는 변수 (또는 식)를 모두 기술하라.

- (k) 그림 2(b)의 식에서 $\|\overrightarrow{\mathbf{VP}}_{pli}\|$ 는 어떤 기하 정보에 해당하는지 정확히 기술하라.
- (l) 그림 2(b)의 식에서 조건식 $\mathbf{P}_{pli}'s w \neq 0$ 이 만족되는 상황은 어떤 경우인지 정확히 기술하라.
- (m) 그림 2(c)의 식에서 두 벡터 $\overrightarrow{\mathbf{P}_{pli}\mathbf{V}}$ 와 $\hat{\mathbf{s}}_{cli}$ 가 의미하는 방향을 정확히 기술하라.
- (n) 그림 2(c)의 식에서 조건식 $\overrightarrow{\mathbf{P}_{pli}\mathbf{V}} \odot \hat{\mathbf{s}}_{cli} \geq \cos c_{rl}$ 이 만족되는 상황은 어떤 경우인지 정확히 기술하라.
- (o) 다면체 모델의 쉐이딩 기법인 Gouraud 쉐이딩 방법을 적용한다고 할 때, 보편적으로 이 조명 공식은 그림 1의 어느 부분에서 적용이 될지, 가장 관련이 깊은 단계의 번호를 기술하라.
- (p) 다면체 모델의 쉐이딩 기법인 Phong 쉐이딩 방법을 적용한다고 할 때, 보편적으로 이 조명 공식은 그림 1의 어느 부분에서 적용이 될지, 가장 관련이 깊은 단계의 번호를 기술하라.

3. 다음은 색깔 혼합에 관한 문제이다. 두 장의 이미지 S와 D의 합성에 대하여 생각하자. $(c_S \alpha_S)$ 와 $(c_D \alpha_D)$ 를 두 이미지의 대응되는 화소의 미리 곱한 색깔(pre-multiplied color)이라 할 때,

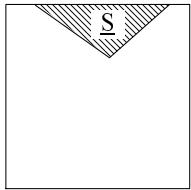


Figure 3: 색깔 합성 예 1

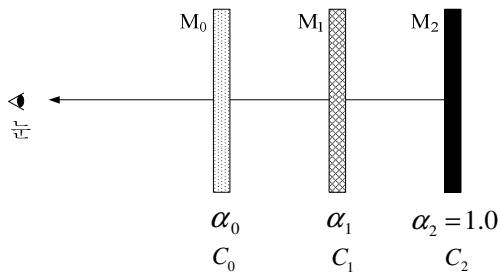


Figure 4: 색깔 합성 예 2

두 색깔의 합성을 통하여 생성한 결과 색깔 ($c_O \alpha_O$)은 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 $(0.5, 0.5, 0.5, 0.5)$ 인 미리 곱한 색깔로 어떤 화소를 칠한다고 할 때, 이는 그 화소를 어떻게 칠하는 것인지 정확히 기술하라.
 - (b) 그림 3에 도시한 합성 연산의 경우 F_S 와 F_D 의 값은 각각 얼마인가?
 - (c) 만약 S over D 연산을 적용한다면, 합성 후 α_O 는 어떤 값을 가질지를 S와 D의 관련 값을 사용하여 정확히 기술하라.
 - (d) 그림 4와 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 비율을 만큼만 통과시키고, 제일 오른쪽에 있는 M_2 는 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.
 - (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 5(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) {
    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((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 5(b)와 같은 결과를 얻으려면, (A)와 (B)에 어떤 인자값이 설정되어야 할지, 다음 값들 중 적절한 인자를 선택하라. (여

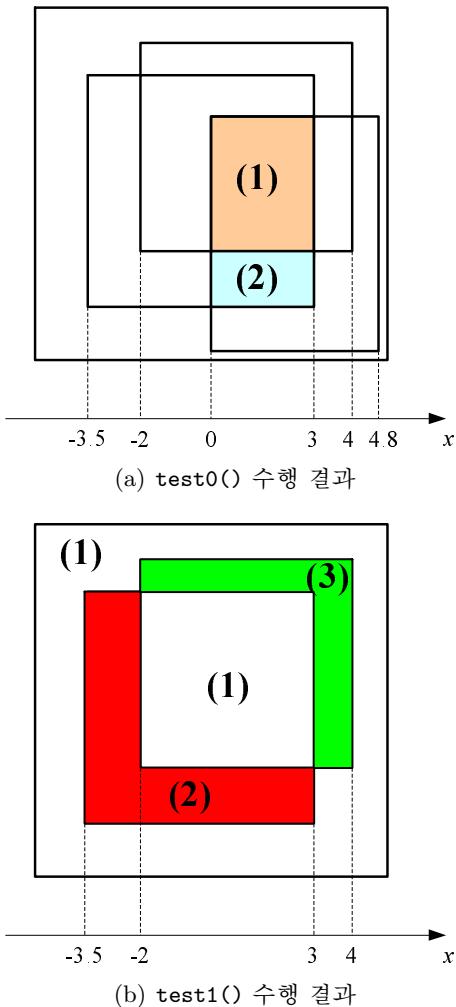


Figure 5: 색깔의 혼합

기서 (1), (2), (3)번 영역의 색깔은 각각 (0, 0, 0), (1, 0, 0), (0, 1, 0)임)

```
GL_ZERO, GL_ONE,
GL_SRC_ALPHA,
GL_DST_ALPHA,
GL_ONE_MINUS_SRC_ALPHA,
GL_ONE_MINUS_DST_ALPHA
```

4. 다음은 Binary Space Partitioning Tree (BSPT)에 관한 문제이다. 다음 쪽의 그림 6의 코드를 보면서 아래 질문에 답하라.

- (a) 함수 `display_bspt_front_to_back()`은 시점의 위치 `viewer[3]`가 주어졌을 때, `bspt`의 다각형들을 앞에서 뒤로 가면서 (front-to-back) 하나씩 나열하면서 그려주는 함수이다. 문맥상 (A)에 들어갈 내용을 C 언어 문법에 맞게 기술하라.

- (b) 문맥상 (B)와 (C)에 들어갈 내용을 C 언어 문법에 맞게 기술하라.
- (c) 문맥상 (D)와 (E)에 들어갈 내용을 C 언어 문법에 맞게 기술하라.
- (d) 다음 `build_bspt()` 함수는 `plist`가 가리키는 `npoly`개로 구성된 다각형 리스트를 받아들여 BSPT를 구성하여 리턴해주는 함수이다. 여기서 `partition plist()` 함수는 `plist`에 존재하는 다각형들을 `bspt->poly`에 의해 정의되는 평면의 앞쪽과 뒷쪽에 있는 다각형들을 분류하여 그에 해당하는 두 리스트 `fplist`와 `bplist`를 구성해주는 역할을 한다. 이때 문맥상 (H)에 들어갈 내용을 C 언어 문법에 맞게 기술하라.
- (e) 다음 `los()` 함수는 3차원 공간의 점을 나타내는 `start`와 `end`에 의해 정의되는 선분이 `bspt`가 저장하고 있는 임의의 다각형과 교차하는지 아닌지를 결정해주는 함수로서, 3차원 공간에서의 네비게이션에 유용하게 활용할 수 있는 함수이다. 이 함수는 문맥상 언제 1 값을, 그리고 언제 0 값을 리턴하는지 정확히 기술하라.
- (f) 문맥상 `los()` 함수 내의 `SOGANG()` 함수는 특정 다각형 `bspt->poly`와 두 지점의 위치 `start`와 `end`에 대한 정보가 주어졌을 때, 정확히 어떤 경우에 NULL이 아닌 값을 리턴해주는 함수일지 정확히 기술하라.
- (g) 문맥상 `los()` 함수에서 (F)와 (G)에 들어갈 내용을 C 언어 문법에 맞게 기술하라.
- (h) 문맥상 `los()` 함수에서 내용적으로 크게 잘못된 부분을 지적하고 옳게 고쳐라.
5. 다음은 OpenGL ES 2.0 Shading Language를 사용한 Gouraud 쉐이딩 방법의 구현에 관한 문제이다. 본 시험 문제의 마지막 두 쪽에는 해당 버텍스 쉐이더의 구현 예가 주어져 있는데, 쉐이딩 계산을 위하여 버텍스 쉐이더에 입력된 꼭지점과 법선 벡터 정보를 EC로 변환하여 풍의 조명 모델 공식을 적용하고 있다.

- (a) 문맥상 EC로 변환된 꼭지점의 좌표를 저장하고 있는 변수 이름을 기술하라.
- (b) 이 버텍스 쉐이더의 계산 결과로 산출되는 결과 정보를 저장해주는 변수들을 모두 기술하라.

```

typedef struct _bspt { // data structure for bspt
    Polygon *poly;
    struct _bspt *fchild; // for the front side
    struct _bspt *bchild; // for the back side
} BSPT;

BSPT *build_bspt(int npoly, PLIST *plist) {
    int fnpoly, bnpoly; PLIST *fplist, *bplist; BSPT *bspt;

    if (npoly == 0) return NULL;
    bspt = (BSPT *) malloc(sizeof(BSPT));
    bspt->poly = pick_poly(&npoly, &plist);
    totalpolygon++;
    fnpoly = bnpoly = 0; fplist = bplist = NULL;
    partition plist(bspt->poly, npoly, plist, &fnpoly, &fplist, &bnpoly, &bplist);
    (H)
    return bspt;
}

void display_bspt_front_to_back(BSPT *bspt, float *viewer) {
    int viewer_side;

    if ((A)) return;
    viewer_side = check_side(viewer, bspt->poly);
    if (viewer_side == BSPT_FRONT) {
        display_bspt_front_to_back((B), viewer);
        draw_bspt_poly(bspt->poly);
        display_bspt_front_to_back((C), viewer);
    }
    else {
        display_bspt_front_to_back((D), viewer);
        draw_bspt_poly(bspt->poly);
        display_bspt_front_to_back((E), viewer);
    }
}

int los(BSPT *bspt, float *start, float *end) {
    int cs, ce, side0, side1;

    if (!bspt) return 1;
    cs = check_side(start, bspt->poly);
    ce = check_side(end, bspt->poly);
    if ((cs == BSPT_FRONT) && (ce == BSPT_FRONT)) {
        return los(bspt->fchild, start, end);
    }
    else if ((cs == BSPT_BACK) && (ce == BSPT_BACK)) {
        return los(bspt->bchild, start, end);
    }
    else {
        if (SOGANG(bspt->poly, start, end)) return 0;
        side0 = los((F), start, end);
        side1 = los((G), start, end);
        return side0 || side1;
    }
}

```

Figure 6: BSPT 관련 함수
- 2012년 6월 22일(금) 오전 11:00 -

- (c) 다음은 Direct3D의 쉐이더인 HLSL Shader에 대한 설명의 일부이다.

As a minimum, a vertex shader must output vertex position in homogeneous clip space.

이 베텍스 쉐이더 코드에서 이 설명과 가장 밀접한 연관이 있는 한 문장의 번호를 기술하라.

- (d) 37)번의 조건문 문장에서 이 조건이 성립 한다는 것은 무엇을 뜻 할까?

- (e) 문맥상 그림 2(a)의 조명 공식에서 57) 번 문장의 `max(c_zero, dot(N, L))` 함수 값에 해당하는 변수 (또는 식)을 정확히 기술하라.

- (f) EC에서 카메라의 기준점인 눈에 해당하는 점의 좌표는?

- (g) 59)번 문장의 `-normalize(p_eye).xyz` 변수는 EC에서 어느 지점에서 어느 지점을 향한 벡터의 길이를 1로 만들어준 단위 벡터를 나타낸다. 과연 어느 지점에서 어느 지점일까?

- (h) 문맥상 59)번 문장의 X 변수 값에 저장되는 내용을 통칭 무엇이라고 부르는가?

- (i) 문맥상 그림 2(a)의 조명 공식에서 55)번 문장 수행 시 `att_factor` 변수가 저장하고 있는 값과 가장 밀접한 관련이 있는 변수 (또는 식)을 정확히 기술하라.

6. 다음은 카메라의 조작에 관한 문제이다.

- (a) 초기 상태의 카메라 프레임이 눈 좌표계의 좌표축과 일치되어 있는 상태에서, 카메라 프레임에 대하여 $M_1 \rightarrow M_2 \rightarrow M_3$ 순서로 변환을 하였다면, 그 경우 뷰잉 변환 행렬 M_V 는 무엇인지 기술하라.

- (b) 초기 상태 $C_0 = (e, u, v, n) = ((0\ 0\ 0), (1\ 0\ 0), (0\ 1\ 0), (0\ 0\ 1))$ 인 카메라를 사용자의 조작을 통하여 $C_1 = ((0\ 10\ 10), (0\ 0\ 1), (0\ 1\ 0), (-1\ 0\ 0))$ 와 같은 상태로 변환하는 상황을 아래와 같은 뷰잉 변환 코드로 구현한다고 가정하자.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMultMatrixf(cam.mat);
glTranslatef(_(A)_ , _(B)_ ,
_(C)_ );
```

- i. 이때 (A), (B), (C)에 들어갈 내용을 기술하라.

- ii. 올바른 변환을 위하여 `cam.mat`이 저장 해야할 4행 4열 변환 행렬의 내용을 정확히 기술하라.

- (c) 그림 1에서 본 카메라 조작 과정과 가장 관련이 있는 단계의 번호를 기술하라.

– 수고 많았습니다 –

vs_gs_simplelight_wc_prob.txt

```
1) attribute vec4 a_position;
2) attribute vec3 a_normal;
3) attribute vec2 a_Texture;

4) uniform mat4 mvp_matrix;
5) uniform mat4 modelview_matrix;
6) uniform mat4 inv_modelview_matrix;
7) uniform float rescale_normal_factor;
8) uniform vec4 material_ambient;
9) uniform vec4 material_diffuse;
10) uniform vec4 material_specular;
11) uniform float material_specular_exponent;
12) uniform vec4 light_position;
13) uniform vec4 light_ambient;
14) uniform vec4 light_diffuse;
15) uniform vec4 light_specular;
16) uniform vec3 spot_direction;
17) uniform float spot_exponent;
18) uniform float spot_cutoff_angle;
19) uniform vec4 global_ambient;

20) varying vec4 v_color;
21) varying vec2 v_Texture;

22) const float c_zero = 0.0;
23) const float c_one = 1.0;

24) vec4 p_eye;
25) vec3 N;

26) vec4 lighting_equation() {
27)     vec4 computed_color = vec4( c_zero, c_zero, c_zero, c_zero );
28)     vec3 X;
29)     float ndotl;
30)     float ndotx;
31)     float att_factor;

32)     float spot_factor;
33)     vec3 att_dist;
34)     vec3 L;

35)     att_factor = c_one;
36)     computed_color += material_ambient * global_ambient;

37)     if ( light_position.w != c_zero ) { // spot light
38)         L = light_position.xyz - p_eye.xyz;
39)         L = normalize(L);

40)         if( spot_cutoff_angle < 180.0 ) {
41)             vec3 spot_dir = normalize(spot_direction);

42)             spot_factor = dot( -L, spot_dir );
43)             if( spot_factor >= cos(radians( spot_cutoff_angle ) ) ) {
44)                 spot_factor = pow(spot_factor, spot_exponent );
45)             }
46)             else {
47)                 spot_factor = c_zero;
48)             }
49)             att_factor *= spot_factor;
50)         }
51)     else {
52)         L = light_position.xyz;
53)         L = normalize(L);
54)     }
55)     if ( att_factor > c_zero ) {
56)         computed_color += light_ambient * material_ambient;
57)         ndotl = max(c_zero,dot(N, L ));
```

```
vs_gs_simplelight_wc_prob.txt
58)     computed_color += ( ndotl * light_diffuse * material_diffuse );
59)     X = normalize( L - normalize(p_eye).xyz);
60)     ndotx = dot( N, X );
61)     if ( ndotx > c_zero )  {
62)         computed_color += (pow ( ndotx, material_specular_exponent)
63)                               * material_specular *
64)         light_specular );
65)     }
66)     computed_color *= att_factor;
67) }
68) void main() {
69)     p_eye = modelview_matrix * a_position;
70)     N = normalize( inv_modelview_matrix * vec4( a_normal, c_zero) ).xyz;
71)     v_color = lighting_equation();
72)     v_Texture = a_Texture;
73)     gl_Position = mvp_matrix * a_position;
74) }
```

기초 컴퓨터 그래픽스 기말고사 (CSE4170)

담당 교수: 임 인 성

- 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 “BSP TREE FREQUENTLY ASKED QUESTIONS (FAQ)”에 기술되어 있는 Binary Space Partitioning Tree (BSPT)에 관한 프로그램의 일부이다 (수업 시간에 다른 내용과 약간 다를 수 있음).

```

struct BSP_tree {
    plane partition; list polygons;
    BSP_tree *front, *back;
};

void Build_BSP_Tree(BSP_tree *tree,
                    list polygons) {
    polygon *root = polygons.Get_From_List();
    tree->partition = root->Get_Plane();
    tree->polygons.Add_To_List(root);

    list front_list, back_list;
    polygon *poly;
    while ((poly = polygons.Get_From_List())
           != 0) {
        int result
        = tree->partition.AAA(poly); // (1)
        switch (result) {
            case COINCIDENT:
                tree->polygons.Add_To_List(poly); break;
            case IN_BACK_OF:
                backlist.Add_To_List(poly); break;
            case IN_FRONT_OF:
                frontlist.Add_To_List(poly); break;
            case SPANNING:
                polygon *front_piece, *back_piece;
                BBB(poly, tree->partition, // (2)
                     front_piece, back_piece);
                backlist.Add_To_List(back_piece);
                frontlist.Add_To_List(front_piece);
                break;
        }
    }
    if (!front_list.IsEmpty_List()) {
        tree->front = new BSP_tree;
        Build_BSP_Tree(_, _); // (3-a)
    }
}

```

```

    }
    if (!back_list.IsEmpty_List()) {
        tree->back = new BSP_tree;
        Build_BSP_Tree(_, _); // (3-b)
    }
}

void Draw_BSP_Tree(BSP_tree *tree, point eye) {
    float result
        = tree->partition.CCC(eye);
    if (result > 0) {
        Draw_BSP_Tree(tree->back, eye);
        tree->polygons.Draw_Polygon_List();
        Draw_BSP_Tree(tree->front, eye);
    }
    else if (result < 0) {
        Draw_BSP_Tree(_, eye); // (4-a)
        tree->polygons.Draw_Polygon_List();
        Draw_BSP_Tree(_, eye); // (4-b)
    }
    else {
        Draw_BSP_Tree(tree->back, eye);
        Draw_BSP_Tree(tree->front, eye);
    }
}

```

- (a) 문맥 상 struct BSP_tree의 partition 은 네 개의 float 타입의 값으로 구성된 구조체 타입의 변수라 할 수 있다. 이 네 개의 값은 무엇을 나타낼까?
- (b) 위의 Build_BSP_Tree()는 다각형 리스트 polygons를 넘겨 받아 적절히 BSPT *tree를 구축해주는 함수이다. 문맥 상 (1)번 라인의 AAA() 함수는 주어진 tree->partition과 poly에 대해 어떤 일을 해주는지 정확히 기술하라.
- (c) 문맥 상 (2)번 라인의 BBB()는 어떤 일을 해주는 함수인지, 어떤 정보를 받아들여 어떤 정보를 넘겨주는지 정확히 기술하라.
- (d) 문맥 상 (3-a)와 (3-b)번 라인의 빈 곳에 들어갈 내용을 정확히 기술하라.
- (e) 위의 Draw_BSP_Tree()는 *tree에 저장되어 있는 다각형들을 시점 eye에 대해 뒤에서 앞으로 가면서 (back-to-front order)

나열해 주는 함수이다. 첫 문장에서 호출되는 CCC() 함수는 문맥 상 주어진 변수 tree->partition과 점 eye에 대해 어떤 경우에 각각 양수, 음수, 그리고 0을 리턴해줄까?

- (f) 문맥 상 (4-a)와 (4-b) 라인의 빈곳에 들어갈 내용을 정확히 기술하라.
 - (g) 위의 코드에서 문맥 상 잘못된 부분을 지적하고 올바르게 고쳐라.
2. 다음도 BSPT의 응용에 관한 문제이다. 지금 가상의 3차원 실내 공간을 자유롭게 돌아다니려 하는데, 사용자가 벽면을 뚫고 지나가지 못하도록 BSPT를 사용하여 충돌검사를 하려한다. 지금 모든 벽면이 사각형으로 모델링이 되어 있는데, 이 사각형들을 사용하여 구성한 BSPT TREE에 대해 아래와 같은 함수를 호출하려한다.

LOS(TREE, A, B);

이때 원하는 것은 3차원 공간의 두 점 A와 B에 의해 정의된 선분이 실내 공간의 임의의 벽면에 해당하는 사각형과 교차를 하는지 알아내는 것인데, 이 작업은 아래의 코드로 수행할 수 있다. 참고로 이 문제에서 BSPT 타입은 다음과 같이 정의되어 있음.

```
typedef struct _BSPT {
    Polygon *poly;
    struct _BSPT *fchild;
    struct _BSPT *bchild;
} BSPT;

int LOS(BSPT *T, float *start, float *end) {
    int side_s, side_e, side_f, side_b;

    if (!T) return 1;
    side_s = check(start, T->poly);
    side_e = check(end, T->poly);

    if ((side_s == BSPT_FRONT)
        && (side_e == BSPT_FRONT)) {
        return LOS( (A) );
    }
    else if ((side_s == BSPT_BACK)
        && (side_e == BSPT_BACK)) {
        return LOS( (B) );
    }
    else {
        if (intersect(T->poly, start, end))
            return 0;
```

```
        else {
            side_f = LOS(T->fchild, start, end);
            side_b = LOS(T->bchild, start, end);
            return (C) ;
        }
    }
```

- (a) 이 함수는 0 또는 1 값을 리턴하도록 되어 있는데, 문맥 상 언제 0 값을, 그리고 언제 1 값을 리턴하는지 정확히 기술하라.
 - (b) 문맥 상 check() 함수가 하는 일이 무엇인지 정확히 기술하라.
 - (c) 문맥 상 (A)와 (B)에 들어갈 내용을 C 언어 문법에 맞게 정확히 기술하라.
 - (d) 문맥 상 (C)에 들어갈 내용을 C 언어 문법에 맞게 정확히 기술하라.
3. 다음은 OpenGL ES 2.0 Shading Language를 사용한 풍 쉐이딩 (Phong Shading)의 구현에 관한 문제이다. 아래에 주어진 버텍스 쉐이더 (vertex shader) 및 프래그먼트 쉐이더 (fragment shader) 코드를 보면서 답하라. 참고로 이 문제에서는 원근투영과 점광원을 사용하고 있음.

```
static const char* VertexShader = STRINGIFY(
attribute vec4 Position;
attribute vec3 Normal;

uniform mat4 Projection;
uniform mat4 Modelview;
uniform float Alpha;

varying vec3 X;
varying vec3 Y;

void main(void) {
    X = (Modelview * Position).xyz;
    Y = normalize((Modelview * vec4(Normal, 0.0)
                    * Alpha).xyz);
    gl_Position = Projection * Modelview
                  * Position;
}
);

-----
static const char* FragmentShader = STRINGIFY(
precision mediump float;

varying vec3 X;
varying vec3 Y;
```

```

uniform vec4 AmbientGlobal;
uniform vec4 DiffuseMaterial;
uniform vec4 AmbientMaterial;
uniform vec4 SpecularMaterial;
uniform float Shininess;
uniform vec4 LightPosition;
uniform vec4 AmbientLight;
uniform vec4 DiffSpecLight;

void main(void) {
    vec4 color = AmbientGlobal * AmbientMaterial;

    vec3 N = normalize(Y); // (1)
    vec3 L = normalize(LightPosition.xyz - X);

    float NdotL = dot(N, L);
    if( NdotL > 0.0 ) { // (2)
        color += AmbientMaterial * AmbientLight;
        vec3 What1 = normalize(-X); // (3)
        vec3 What2 = normalize(L + What1); // (4)
        float NdotWhat2 = max(dot(N, What2), 0.0);
        vec4 diffuse = DiffuseMaterial * NdotL;
        vec4 specular = SpecularMaterial
            * pow(NdotWhat2, Shininess);
        gl_FragColor = color + DiffSpecLight *(5);
    }
    else {
        gl_FragColor = color;
    }
}
);

```

- (a) 위 쉐이더를 사용하면 렌더링하고자 하는 물체 표면의 어떤 지점들에 대해 쉐이딩 계산이 일어난다고 할 수 있을까?
- (b) 또한 문맥상 OpenGL의 어떤 좌표계에서 쉐이딩 계산이 수행될까? OC, MC, WdC, CC, NDC, EC, WC 등의 기호를 사용하여 답하라.
- (c) 버텍스 쉐이더의 출력 변수인 varying 변수 X와 Y에 저장된 값이 렌더링 파이프라인의 어느 시점에서, 그리고 어떻게 프래그먼트 쉐이더의 입력 변수인 varying 변수 X와 Y 값으로 변환이 되는지 정확히 기술하라.
- (d) 한 점 p 를 4행 4열 행렬 M 이 나타내는 아핀 변환을 통해 기하 변환을 한다고 하자. p 에서의 법선 벡터 (normal vector) n 에 대해 동일한 아핀 변환을 가하려면, n 에 어떤 행렬을 곱해야할지 M 또는 M 의 내용을 적절히 사용하여 표현하라 (힌트:

본 시험 문제에서 $A_{3 \times 3}$ 은 4행 4열 행렬 A 의 왼쪽-위 3행 3열 부행렬을 나타냄).

- (e) 이 문제의 예제 프로그램에서 사용하는 모델링 변환과 뷰잉 변환은 이동변환 T , T_1 , 회전변환 R , R_1 , 그리고 각 축 방향으로 s 배 확대해주는 크기 변환 $S(s)$ 에 대해 다음과 같다.

$$M_M = R * S(s) * R_1 * T_1,$$

$$M_V = T$$

모델뷰 행렬 $M_{MV} = M_V * M_M$ 에 대해 $((M_{MV})_{3 \times 3})^{-1}$ 는 다음과 같이 세 개의 행렬의 곱으로 표현할 수 있는데, 이때 행렬 B 와 C 의 내용을 정확히 기술하라.

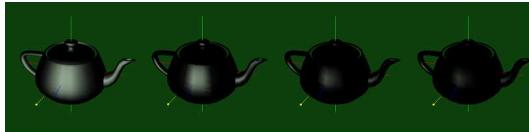
- $((M_{MV})_{3 \times 3})^{-1} = B * S(\frac{1}{s}) * C$
- (f) $((M_{MV})_{3 \times 3})^{-1}$ 을 크기 변환 인자 값 s 와 M_{MV} 의 내용을 적절히 사용하여 표현하라.
- (g) 위 버텍스 쉐이더에서 문맥 상 uniform 변수인 Alpha에는 어떤 값이 담겨져 올까?
- (h) 다음은 Direct3D의 쉐이더인 HLSL Shader에 대한 설명의 일부이다.

As a minimum, a vertex shader must output vertex position in homogeneous clip space.

위의 쉐이더 프로그램에서 이 설명과 가장 밀접한 연관이 있는 한 문장을 기술하라.

- (i) 위 버텍스 쉐이더는 약간 비효율적으로 작성되어 있다. 이 쉐이더를 어떻게 고치면 좀 더 효율적으로 렌더링 계산을 수행할 수 있을까?
- (j) (1)번 문장은 Y 벡터의 길이를 1로 만들어 주고 있다. 만약 이 문장을 제거하면 렌더링 결과에 많은 차이가 있는지, ‘예/아니오’로 답하고 그 이유를 정확히 설명하라.
- (k) (2)번 문장의 if 문은 어떠한 경우에 지역 조명 모델식을 적용하지 않겠다는 것인지 정확히 기술하라.
- (l) 만약 (2)번 문장의 if 문을 제거해도 제거 전과 동일하게 쉐이딩이 될지, ‘예/아니오’로 답하고 그 이유를 정확히 기술하라.
- (m) (3)번 문장의 What1 변수에는 정확히 어떤 지점에서 어느 방향을 가리키는 벡터가 저장이 될까?
- (n) (4)번 문장의 What2 변수가 저장하는 벡터를 통칭 무엇이라고 하는가?

- (o) 문맥상 (5)번에 들어갈 내용을 정확한 쉐이더 문법을 사용하여 기술하라.
- (p) 아래 그림은 위의 쉐이더 프로그램에서 어떤 특정 변수 값을 바꾸어가며 조절한 렌더링 효과를 보여주고 있다. 과연 어떤 변수인지 정확히 기술하라.

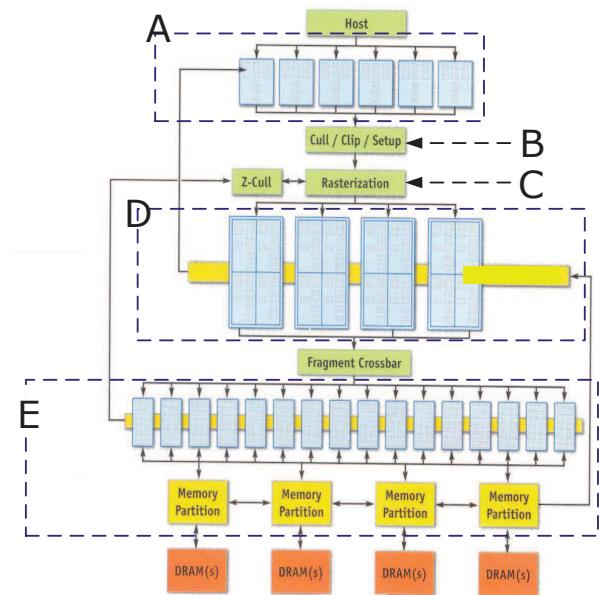


4. 다음의 풍의 조명 모델 (Phong's illumination model) 식을 보고 답하라.

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (N \circ H_i)^n\}$$

- (a) 변수 $I_{a\lambda}$, $k_{a\lambda}$, d_i , $I_{l_i\lambda}$, $k_{d\lambda}$, N , L_i , $k_{s\lambda}$, H_i , n 중 카메라의 위치에 영향을 받는 변수를 모두 나열하라.
- (b) 물체의 위치와 방향이 바뀌었을 때 영향을 받는 변수를 모두 나열하라.
- (c) Lambert의 코사인 법칙을 표현해주는 부분을 정확히 기술하라.
- (d) 광원의 위치 변화에 영향을 받는 변수를 모두 나열하라.
- (e) H_i 는 어떤 상황에서 특히 효율적으로 사용이 되는가? 두 가지 조건을 명시하라.
- (f) 플랫 쉐이딩 (flat shading), 풍 쉐이딩, 그리고 고우러드 쉐이딩 (Gouraud shading) 등 세 가지 방법을 계산량이 적은 것부터 순서대로 나열하라.
- (g) $f_{att}(d_i)$ 에서 d_i 는 i 번째 광원에 대해 어떤 값을 나타내는 변수인지, 그리고 이 함수는 어떤 효과를 나타내기 위한 것인지 정확히 기술하라.
- (h) 위 문제에 주어진 fragment shader에서 $I_{l_i\lambda}$ 값과 가장 밀접한 관계가 있는 uniform 변수 값을 기술하라 ($f_{att}(d_i)$ 부분 제외).
- (i) 앞 문제에 주어진 프래그먼트 쉐이더는 한 개의 광원만 (즉 $m = 1$) 고려하고 있는데, 이 문제에 주어진 식과 이 쉐이더에서 사용하는 조명 모델의 가장 큰 차이를 기술하라 ($f_{att}(d_i)$ 부분 제외).

5. 다음 그림은 몇 년전에 발표된 어떤 GPU (Graphics Processing Unit)의 렌더링 계산 구조를 도시하고 있다. 각 문제에 대해 가장 연관성이 높은 파이프라인 부분의 이름을 A, B, C, D, 그리고 E 중의 하나로 기술하라.



- (a) 쉐이더를 통하여 풍 쉐이딩을 구현할 때, 풍의 조명 모델이 적용되는 부분은?
- (b) 꼭지점의 나열 순서에 따라 다각형을 제거할 수도 있는 부분은?
- (c) 필터링 방법 중의 하나인 mipmapping과 가장 밀접한 관계가 있는 부분은?
- (d) OpenGL의 glBlendFunc() 함수가 직접적으로 영향을 미치는 부분은?
- (e) 실제로 은면 제거 (hidden surface elimination) 계산이 수행되는 부분은?
- (f) 카메라를 기준으로 하는 좌표계가 존재하는 부분은?
- (g) 일반적으로 hierarchical modeling과 가장 관계가 깊은 부분은?
- (h) 버텍스 쉐이더의 속성 데이터가 프래그먼트 쉐이더의 속성 데이터로 변환되는 부분은?
- (i) 원근 나눗셈 연산이 수행되는 부분은?
- (j) OpenGL의 gluPerspective() 함수로 설정한 뷰 볼륨의 밖에 존재하는 기하 프리미티브 (geometric primitive)들이 제거가 되는 부분은?

- (k) 정상적인 원근 투영 변환을 할 때, 꼭지점의 동차 좌표의 W좌표 값이 1이 아닌 값이 나타날 수 있는 부분은?
- (l) 뷔퍼 변환과 가장 밀접한 관련이 있는 부분은?
- (m) 꼭지점들이 기하 프리미티브들로 조립되는 부분은?

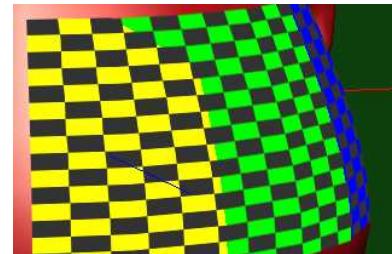
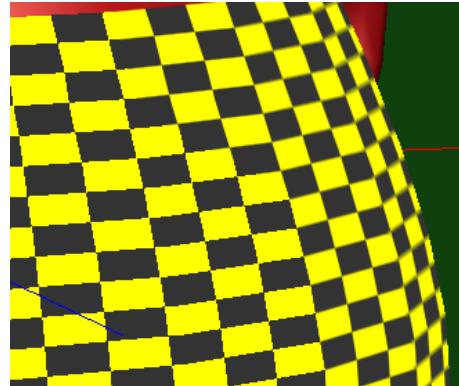
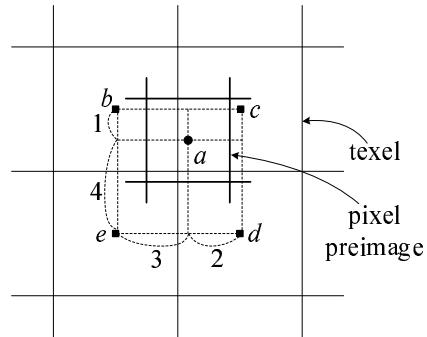
6. 다음은 광원의 설정에 관한 문제이다.

- (a) 지금 광부들이 사용하는 헤드 라이트와 같이 카메라에 고정된 점 광원을 사용하려 한다. 만약 광원을 항상 카메라의 기준점에서 위로 5.0만큼 떨어진 곳에 위치시키려하면, 아래의 코드에서 `glLightfv(GL_LIGHT0, GL_POSITION, li_pos);` 문장을 정확히 몇 번 라인과 몇 번 라인 사이에 삽입해야 하는지, 그리고 그때 (A), (B), (C), (D)에 들어갈 내용은 무엇인지 기술하라.

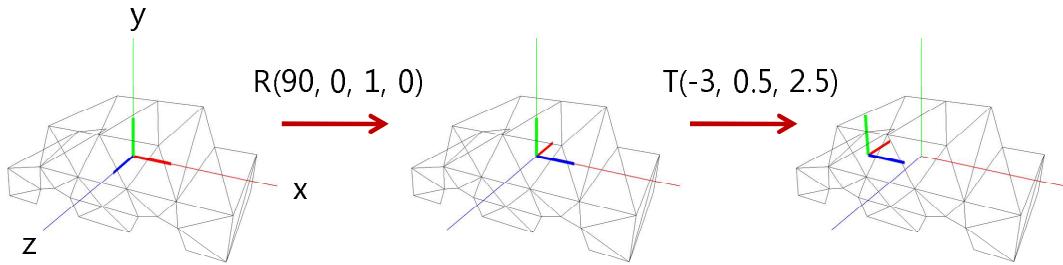
```
GLfloat li_pos[4] = { (A) , (B) ,
(C) , (D) }; // (1)
:
glMatrixMode(GL_MODELVIEW); // (2)
glLoadIdentity(); // (3)
// Do viewing transform.
gluLookAt(v[0], v[1], v[2],
           c[0], c[1], c[2],
           0.0, 1.0, 0.0); // (4)
glPushMatrix(); // (5)
// Do modeling transform.
glRotatef(45.0,
          0.0, 1.0, 0.0); // (6)
glTranslatef(-2.5,
              0.5, -9.5); // (7)
draw_object_in_MC(); // (8)
glPopMatrix(); // (9)
```

- (b) 만약 광원을 세상 좌표계의 (3.0, 0.5, -1.5) 인 지점에 고정시킨 후 렌더링을 하려할 경우에 대해 위 문제를 풀어라.
- (c) 만약 광원을 (8)번 라인에서 그리고 있는 물체의 모델링 좌표계의 (3.0, 0.5, -1.5) 지점에 고정시킨 후, 렌더링을 하려할 경우에 대해 위 문제를 풀어라.

7. 다음은 OpenGL 환경에서의 텍스춰 필터링에 관한 문제이다. 아래 그림을 보면서 답하라.



- (a) 레벨 0 (최고 해상도)의 mipmap 텍스춰의 한 텍셀이 나타내는 영역의 면적은 레벨 3의 mipmap 텍스춰의 한 텍셀이 나타내는 영역의 면적의 몇 배일까?
- (b) 첫 번째 그림에서 a 는 픽셀에 대한 원상 (pre-image)의 중점을 가리키고, b 부터 e 까지는 a 를 포함하는 주변 텍셀의 중심점을 나타낸다. b , c , d , 그리고 e 지점의 텍셀 색깔이 각각 (1, 1, 1), (1, 0, 0), (1, 1, 0), 그리고 (0, 1, 1)이라고 하자. 이때 선형 필터 (GL_LINEAR)를 사용할 경우 a 지점에 대해 계산되는 색깔은 무엇일까? 반드시 계산 과정을 밝혀라 (이 그림에서 숫자는 거리에 대한 비율을 나타냄).
- (c) 두 번째 그림은 축소 필터로 GL_LINEAR를, 확대 필터로는 GL_NEAREST를 사용한 경우의 렌더링 결과이다. 지금 축소 및 확대 두 필터가 동시에 적용이 되고 있는데, 과연 이러한 상황에는 왼쪽과 오른쪽 부분에 각각 어떤 필터가 적용되고 있을까? 그리고 그 이유는 무엇일까?



- (d) 세 번째 그림은 축소 상황이 발생한 경우이다. 서로 다른 레벨의 mip맵 텍스춰에 대해서 서로 다른 색깔의 텍스춰 이미지를 로드한 후, 축소 필터로 `GL_LINEAR_MIPMAP_NEAREST`를 사용한 결과인데, 만약 이 축소 필터 대신에 `GL_LINEAR_MIPMAP_LINEAR`를 사용할 경우 렌더링 결과가 어떻게 달라질까?
8. 다음은 배열 `p[*][*]`에 저장된, y_w 축에 수직인 평면 상의 경로를 따라 움직이는 자동차에서 바라본 세상을 렌더링해주는 코드의 일부이다.

```

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

    v[0] = ...; // (1)

    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);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    (2)
    draw_axes(); draw_path();
}

```

```

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

```

- (a) (1)번 문장에서 `v` 벡터의 값 `v[0]`, `v[1]`, `v[2]`의 값을 설정하고 있는데, 각 원소 값은 문맥상 어떤 값으로 설정되어야 할까? 세상 좌표계와 자동차 몸체의 모델링 좌표계 모두 위쪽이 y 축 방향에 해당함.
- (b) 문맥상 `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.
- (c) 이 쪽의 위에 있는 그림은 자동차 몸체의 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.
- (d) 지금 운전석에 배치한 카메라를 사용하여 렌더링한다고 가정할 때, `display()` 함수의 (2)에 들어갈 뷰잉 코드를 OpenGL 함수들을 적절히 사용하여 C언어 문법에 맞게 정확히 기술하라 (힌트: 카메라 프레임은 자동차 몸체를 세상 좌표계에 배치할 때 같이 움직임).

- 끝 -

기초 컴퓨터 그래픽스 기말고사 (CSE4170)

담당 교수: 임 인 성

- 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.

- 아래 그림 1에 주어진 실시간 3D 렌더링 파이프라인의 계산 과정에 대한 질문에 답하라.

(1) vertex stream → (2) Vertex Shader → (3) vertex stream → (4) Primitive Assembly → (5) primitive stream → (6) Geometry Shader → (7) primitive stream → (8) Clipping & Setup → (9) Rasterization → (10) pixel stream → (11) Pixel Shader → (12) pixel stream → (13) Raster Operation → (14) Framebuffer

그림 1: 실시간 렌더링 계산 과정 예

- OpenGL 파이프라인에서의 색깔 혼합 (color blending) 계산과 가장 관련이 깊은 단계의 번호는?
- OC, MC, WC, EC, CC, NDC, WdC 등의 OpenGL 좌표계 중 (3)번 시점은 어떤 좌표계에 해당할까?
- 뷰포트 변환과 가장 밀접한 관련이 있는 단계의 번호는?
- 모델링 변환과 뷰잉 변환이 일어나는 단계에 해당하는 번호는?
- “전통적인 OpenGL 파이프라인”에서 은 면제거, 즉 안보이는 면이 제거되는 계산과 가장 관련이 깊은 단계의 번호는?

(f) 다면체 모델에 대한 쉐이딩 방법 중의 하나인 풍 쉐이딩 방법을 구현하려할 때, 실제로 풍의 조명 모델 식의 계산이 가장 자연스럽게 수행될 수 있는 단계에 해당하는 번호는?

(g) 꼭지점의 좌표에 투영 변환 행렬이 곱해지는 단계에 해당하는 번호는?

(h) 삼각형의 꼭지점에 설정된 여러 속성들이 픽셀들에 대한 속성으로 바뀌는 단계에 해당하는 번호는?

- 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);
    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) &&
```

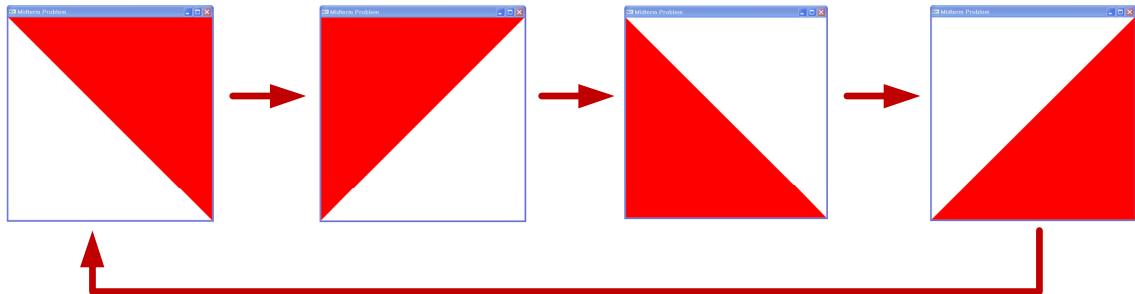


그림 2: OpenGL을 사용한 2차원 기하 변환

```

        (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) 사용자가 왼쪽 마우스 버튼을 누를 때마다 화면의 그림이 어떤 식으로 순환하는지, 그림 2와 같은 방식으로 그려라. 도형의 모양과 위치를 가급적 정확히 표시하고 어떤 부분이 적색 영역인지 분명히 밝힐 것.
- (b) 만약 그림 2에 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.
- (c) 원래의 프로그램 상태에서 university() 함수의 glOrtho() 함수 문장을 아래와 같이 수정할 경우 화면의 내용이 어떻게 반복될지, 그림 2와 같은 방식으로 그려라.

```

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

```
- (d) 원래의 프로그램 상태에서 university() 함수의 glOrtho() 함수 문장 직전에 아래와 같은 문장을 삽입할 경우 화면의 내용이 어떻게 반복될지, 그림 2와 같은 방식으로 그려라.

```

glScalef(0.5, 0.5, 1.0);

```

(힌트: 앞 문제의 직교 투영 변환을 잘 생각해보고, 이 크기 변환이 직교 투영에 어

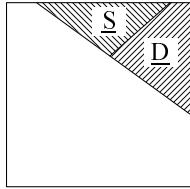


그림 3: 색깔 합성 예 1

- 면 영향을 미치는지를 생각해볼 것)
3. 다음은 색깔 혼합에 관한 문제이다. 두 장의 이미지 S와 D의 합성에 대하여 생각하자. ($c_S \alpha_S$)와 ($c_D \alpha_D$)를 두 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 $(0.5, 0.5, 0.5, 0.5)$ 인 미리 곱한 색깔로 어떤 화소를 칠한다고 할 때, 이는 그 화소를 어떻게 칠하는 것인지 정확히 기술하라.
- (b) 그림 3에 도시한 합성 연산의 경우 F_S 와 F_D 의 값은 각각 얼마인가?
- (c) 만약 S over D 연산을 적용한다면, 합성 후 α_O 는 어떤 값을 가질지를 S와 D의 관련 값을 사용하여 정확히 기술하라.
- (d) 그림 4와 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 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.

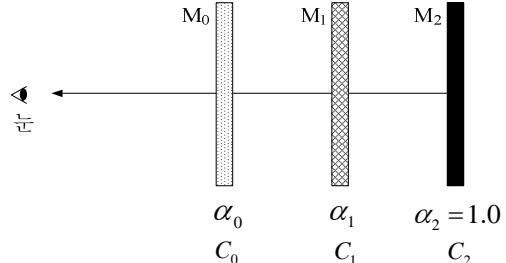


그림 4: 색깔 합성 예 2

- (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 5(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) {
    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((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 5(b)와 같은 결과를 얻으려면, (A)와 (B)에 어떤 인자값이 설정되어야 할지, 다음 값들 중 적절한 인자를 선택하라. (여기서 (1), (2), (3)번 영역의 색깔은 각각 (0, 0, 0), (1, 0, 0), (0, 1, 0)임)

```

    GL_ZERO, GL_ONE,
    GL_SRC_ALPHA,
    GL_DST_ALPHA,
    GL_ONE_MINUS_SRC_ALPHA,
    GL_ONE_MINUS_DST_ALPHA

```

4. 지금 가상의 3차원 실내 공간을 자유롭게 네비게이션 하려하는데, 사용자가 벽면을 뚫고 지나가지 못하도록 Binary Space Partitioning Tree (BSPT)를 사용하여 충돌 검사를 하려

한다. 지금 모든 벽면이 사각형으로 모델링이 되어 있는데, 이 사각형들을 사용하여 구성한 BSPT TREE에 대해 아래와 같은 함수를 호출하여한다.

```
LOS(TREE, A, B);
```

이때 원하는 것은 3차원 공간의 두 점 A와 B에 의해 정의된 선분이 실내 공간의 임의의 벽면에 해당하는 사각형과 교차를 하는지 알아내는 것인데, 이 작업은 아래의 코드로 수행할 수 있다. 참고로 BSPT 타입은 다음과 같이 정의되어 있음.

```

typedef struct _BSPT {
    Polygon *poly;
    struct _BSPT *fchild;
    struct _BSPT *bchild;
} BSPT;

```

```

int LOS(BSPT *T, float *start, float *end) {
    int side_s, side_e, side_f, side_b;

    if (!T) return 1;
    side_s = check(start, T->poly);
    side_e = check(end, T->poly);

    if ((side_s == BSPT_FRONT)
        && (side_e == BSPT_FRONT)) {
        return LOS( (A) , start, end);
    }
    else if ((side_s == BSPT_BACK)
        && (side_e == BSPT_BACK)) {
        return LOS( (B) , start, end);
    }
    else {
        if (intersect(T->poly, start, end))
            return 0;
        else {
            side_f = LOS(T->fchild, start, end);
            side_b = LOS(T->bchild, start, end);
            return (C) ;
        }
    }
}

```

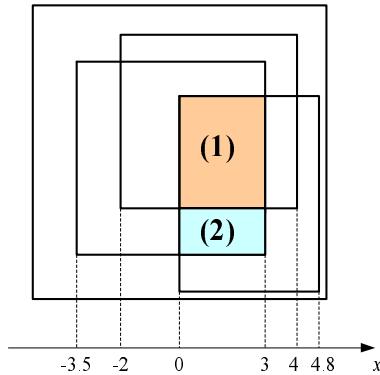
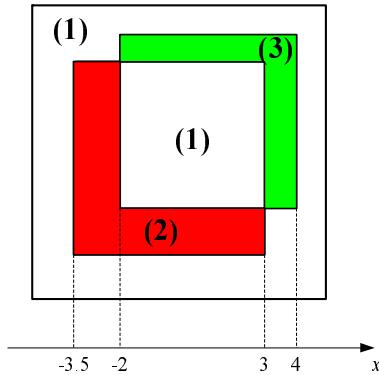
(a) `test0()` 수행 결과(b) `test1()` 수행 결과

그림 5: 색깔의 혼합

- ```

 }
}
}

(a) 이 함수는 0 또는 1 값을 리턴하도록 되어
 있는데, 문맥 상 언제 0 값을, 그리고 언제
 1 값을 리턴하는지 정확히 기술하라.

(b) 문맥 상 check() 함수가 하는 일이 무엇
 인지 정확히 기술하라.

(c) 문맥 상 (A)와 (B)에 들어갈 내용을 C 언
 어 문법에 맞게 정확히 기술하라.

(d) 문맥 상 (C)에 들어갈 내용을 C 언어 문
 법에 맞게 정확히 기술하라.

```

5. 이 문제도 Binary Space Partitioning Tree에 관한 것이다. 그림 6의 함수 `display_bspt_back_to_front()`는 시점의 위치 `viewer[3]`가 주어졌을 때, 뒤에서 앞으로 (back-to-front) 가면서 bspt의 다각형들을 그려주는 함수이다.

- 이 예제 프로그램의 문맥상 (A)에 들어갈 가장 적절한 내용의 C 코드를 작성하라. 이 함수는 결과로 BSPT\_FRONT 또는 BSPT\_BACK을 리턴해야 한다.
- (B), (C), (D), (E)에 들어갈 내용을 정확한 C 코드 형태로 기술하라.

(c) `bspt`가 나타내는 물체를 OpenGL의 블렌딩 기능을 사용하여 투명하게 그려주려한다. (F)에 들어갈 내용을 정확한 C 코드 형태로 기술하라.

(d) 위의 문제를 모두 제대로 풀었다는 가정하에, 만약 뒤에서 앞으로 (back-to-front)가 아니라 앞에서 뒤로 가면서 (front-to-back) 다각형들을 그려 주려면 이 프로그램의 어느 부분을 어떻게 고쳐야 할지, 문장 번호를 언급하며 정확히 기술하라.

6. 다음은 라이팅과 쉐이딩에 관한 문제이다. 아래의  $m$ 개의 광원에 대한 풍의 조명 모델 식을 보고 답하라.

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (N \circ H_i)^n\}$$

- 변수  $I_{a\lambda}$ ,  $k_{a\lambda}$ ,  $d_i$ ,  $I_{l_i\lambda}$ ,  $k_{d\lambda}$ ,  $N$ ,  $L_i$ ,  $k_{s\lambda}$ ,  $H_i$ ,  $n$  중 카메라의 위치에 영향을 받는 변수를 모두 나열하라.
- 물체의 위치와 방향이 바뀌었을 때 영향을 받는 변수를 모두 나열하라.
- 광원의 위치 변화에 영향을 받는 변수를 모두 나열하라.

```

typedef struct {
 int nv; // number of vertices
 float *vertex; // pointer to vertex data
 float *normal; // pointer to normal data
 float plane[4]; // plane equation
} Polygon;

typedef struct _bspt { // data structure for bspt
 Polygon *poly;
 struct _bspt *fchild; // for the front side
 struct _bspt *bchild; // for the back side
} BSPT;

int check_side(float *pos, Polygon *poly) {
 (A)
}

BSPT *bspt; // object in BSPT
float viewer[3]; // camera position
:
void display_bspt_back_to_front(BSPT *bspt, float *viewer) {
 int viewer_side; // (1)
 if (bspt == NULL) return; // (2)
 viewer_side = check_side(viewer, bspt->poly); // (3)
 if (viewer_side == BSPT_BACK) { // (4)
 display_bspt_back_to_front((B), viewer); // (5)
 draw_bspt_poly(bspt->poly); // (6)
 display_bspt_back_to_front((C), viewer); // (7)
 }
 else { /* viewer_side == BSPT_FRONT */ // (8)
 display_bspt_back_to_front((D), viewer); // (9)
 draw_bspt_poly(bspt->poly); // (10)
 display_bspt_back_to_front((E), viewer); // (11)
 }
}

void display(void) {
 glBlendFunc((F)); // (12)
 glEnable(GL_BLEND); // (13)
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // (14)
 display_bspt_back_to_front(bspt, viewer); // (15)
 glutSwapBuffers();
}

```

그림 6: BSPT 관련 함수

- (d)  $H_i$ 는 어떤 상황에서 특히 효율적으로 사용이 되는가? 두 가지 조건을 명시하라.
- (e) 플랫 쉐이딩, 풍 쉐이딩, 그리고 고우러드 쉐이딩 등 세 가지 방법을 계산량이 적은 것부터 순서대로 나열하라.
7. 다음은 Cg API를 통한 쉐이더 프로그래밍과 풍의 조명 모델에 관한 문제이다. 아래에 주어진 버텍스 쉐이더 및 텍셀 쉐이더를 보고 답하라. 참고로 이 예에서는 원근 투영과 점광원을 사용하고 있음.

```
struct _output {
 float4 X: TEXCOORD0;
 float3 Y: TEXCOORD1; // (0)
 float4 position: POSITION;
};

_output main(float4 A: POSITION,
 float4 B: NORMAL,
 uniform float4x4 ModelViewProj:
 state.matrix.mvp,
 uniform float4x4 ModelView:
 state.matrix.modelview,
 uniform float4x4 ModelViewIT:
 state.matrix.(4).invtrans) {
 _output OUT;
 OUT.X = mul(ModelView, A); // (1)
 OUT.Y = mul(ModelViewIT, B).xyz; // (2)
 OUT.position = mul(ModelViewProj, A); // (3)
 return OUT;
}
```

---

```
struct _output { float3 color: COLOR; };

_output main(float3 X: TEXCOORD0,
 float3 Y: TEXCOORD1,
 uniform float4 g_a:
 state.lightmodel.ambient,
 uniform float4 lp:
 state.light[0].position,
 uniform float4 la:
```

```
state.light[0].ambient,
uniform float4 ld:
 state.light[0].diffuse,
uniform float4 ls:
 state.light[0].specular,
uniform float4 ma:
 state.material.ambient,
uniform float4 md:
 state.material.diffuse,
uniform float4 ms:
 state.material.specular,
uniform float me:
 state.material.shininess) {
 _output OUT;

 OUT.color = g_a * ma;
 float3 N = normalize(Y);
 float3 L = normalize(lp - X);
 float NdotL = dot(N, L);
 OUT.color += ma * la;
 if(NdotL >= 0.0) { // (5)
 float3 What1 = normalize(-X); // (6)
 float3 What2 = normalize(L + What1); // (7)
 float NdotWhat2 = dot(N, What2);
 float4 li = lit(NdotL, NdotWhat2, me);
 float3 D = md * li.y;
 float3 S = ms * li.z;
 OUT.color += (8);
 }
 return OUT;
}
```

- (a) 이 프로그램은 각 텍셀을 통해서 보이는 물체의 각 지점에서 풍의 조명 모델을 적용하여 쉐이딩 계산을 하는 풍 쉐이딩 계산을 위한 쉐이더 코드이다. 이 프로그램에서는 OpenGL의 어떤 좌표계에서 쉐이딩 계산을 하고 있을까? OC, MC, WdC, CC, NDC, EC, WC 등의 기호를 사용하여 답하라.

(b) 주어진 한 점  $p$ 를 4행 4열 행렬  $M$ 이 나타내는 아핀 변환을 통해 기하 변환한다고 할 때,  $p$ 에서의 법선 벡터 (normal vector)  $n$ 에 대해 동일한 아핀 변환을 가하려면  $n$ 에 어떤 행렬을 곱해야할지  $M$ 을 사용하여 표현하라.

(c) 만약 (0) 문장의

```
float3 Y: TEXCOORD1;를
```

```
float3 Y: COLOR1;과
```

같이 변경하였을 때, 이 쉐이더들이 제대로 작동하려면 약간의 수정이 필요한데, 그 내용을 C/Cg 문법에 맞게 정확히 기술하라.

(d) 보편적인 렌더링 관련 쉐이더 코드 작성 시 버텍스 쉐이더에서는 꼭 해주어야 하는 작업이 하나 있다. 위의 버텍스 쉐이더에서 이 작업과 가장 관련이 깊은 문장의 번호는 (1), (2), (3) 중 어떤 것이며, 이 작업은 구체적으로 무엇을 의미하는지 정확히 기술하라.

(e) 문맥상 버텍스 쉐이더 코드의 (4) 부분에 들어갈 내용을 정확히 기술하라. 대소문자를 구별할 것.

(f) (5) 번의 if 문장은 어떠한 경우에 지역 조명 모델식을 계산하지 않겠다는 것인지 그 상황을 설명하라.

(g) 만약 (5) 번의 if 문장을 제거해도 제거전과 동일하게 쉐이딩이 될까? 예/아니오로 답하고 그 이유를 정확히 기술하라.

(h) (6) 번 문장의 `What1` 변수에는 정확히 어떤 지점에서 어느 방향을 가리키는 벡터가 저장이 될까?

(i) (7) 번 문장의 `What2` 변수가 저장하는 벡터를 통칭 무엇이라고 하는가?

(j) 문맥상 (8) 번에 들어갈 내용을 C 언어 문맥에 맞게 정확히 기술하라.

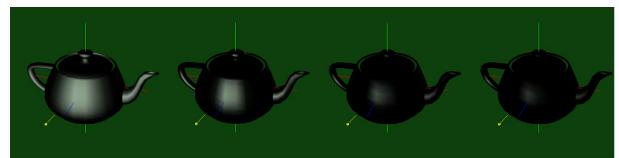


그림 7: 조명 효과의 조절

(k) 그림 7은 위의 쉐이더 프로그램에서 어떤 특정 변수 값을 바꾸어가며 조절한 렌더링 효과를 보여주고 있다. 과연 어떤 변수인지 정확히 기술하라.

## [CSE4170: 기초 컴퓨터 그래픽스]

### 기말고사 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.

1. 그림 1에는 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).

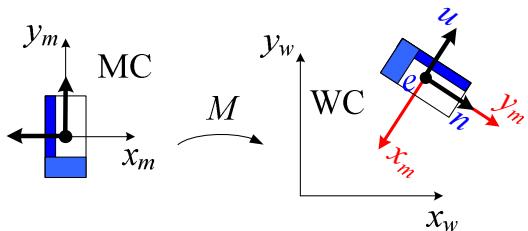


그림 1: 모델링 변환과 뷔잉 변환

- (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 등의 기본 변환 행렬의 곱으로 적절히 표현하라 (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 행

렬은 계산할 필요가 없음).

- (b)  $C = (e, (u, v, n))$  정보를 사용하여 카메라를 배치하려 할 때 적용되는 뷔잉 변환  $M_V$ 를 위의 문제에서 사용한 기본 변환의 행렬 또는 그의 역행렬을 가급적 많이 사용하여 표현하라.
- (c) 마지막 쪽의 그림 9에서 이 문제와 가장 관련이 깊은 박스의 기호를 기술하라.
2. 다음은 색깔 혼합 (color blending)에 관한 문제이다.  $(c_A \alpha_A)$ 와  $(c_B \alpha_B)$ 를 두 장의 이미지 A와 B의 대응되는 화소의 미리 곱한 색깔 (pre-multiplied color)이라 하자.  $F_A$ 와  $F_B$ 를 각각 A와 B의 화소에서  $\alpha_A$ 와  $\alpha_B$ 의 비율로 존재하는 미립자들 중 결과 이미지 O에 살아 남는 미립자의 비율이라 하면, 합성의 결과로 생성되는 미리 곱한 색깔은 다음과 같다.

$$\begin{pmatrix} c_O \\ \alpha_O \end{pmatrix} = F_A \begin{pmatrix} c_A \\ \alpha_A \end{pmatrix} + F_B \begin{pmatrix} c_B \\ \alpha_B \end{pmatrix}$$

- (a) 투명한 물체를 렌더링하거나, 안개 등의 기상 효과, 텍스처의 혼합, 앤티앨리어싱 기법들을 구현하는데 유용하게 사용이 되는 A over B 연산의 경우  $F_A$ 와  $F_B$ 는 각각 얼마일까?
- (b) 그림 2의 아래 쪽 그림은 시선을 따라 샘플링한 색깔들을 카메라를 기준으로 앞에서 뒤로 가면서 (front-to-back) 점진적으로 합성해주는 상황을 도시하고 있다. 이

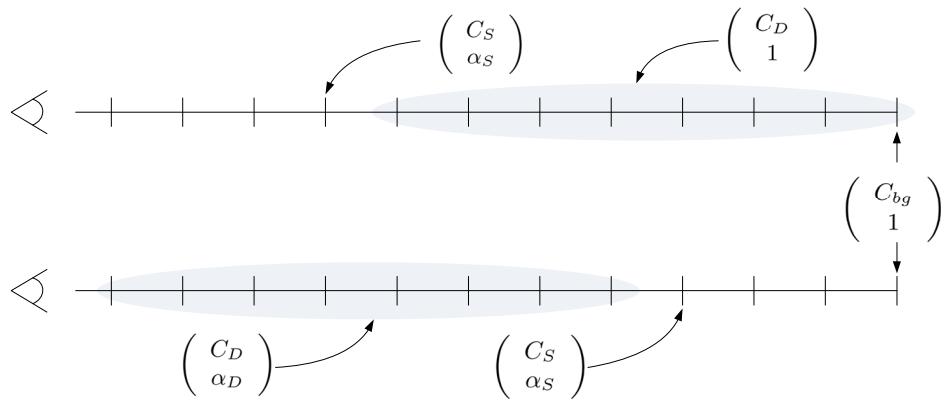


그림 2: Back-to-front and front-to-back compositions

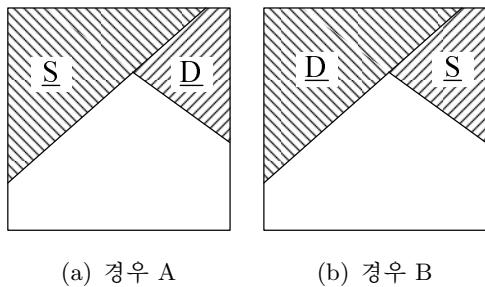


그림 3: 두 가지 경우

를 OpenGL API를 사용하여 구현하려면, 현재까지 합성한 색깔을 미리 곱한 색깔 ( $c_D \alpha_D$ ) = ( $\alpha_D C_D \alpha_D$ ) 형태로 RGBA 타입의 색깔 버퍼 (color buffer)에 저장한 후, 지금 추가적으로 합성하려는 색깔을 렌더링 파이프라인을 타고 들어오는 미리 곱한 색깔 형태의 프래그먼트 색깔 ( $c_S \alpha_S$ ) = ( $\alpha_S C_S \alpha_S$ )로 하여 합성을 해주면 된다. 이때 그림 3(a)와 (b)에 도시된 두 가지 경우 중 이 문제에 더 적합한 경우는 A와 B 중 어떤 것일까?

- (c) 위 문제에 연속하여, 합성의 결과 새로이 생성되는 미리 곱한 색깔을 ( $c_O \alpha_O$ ) = ( $\alpha_O C_O \alpha_O$ )라 할 때,  $\alpha_O$ 의 식을 정확히 기술하라.
- (d) 이때 합성된 화소에 칠해줄 실제 색깔 (미리 곱한 색깔이 아닌)  $C_O$ 에 대한 식을  $\alpha_D, \alpha_S, C_D, C_S$  등으로 표현하라.

(e) 위 세 문제에서의 혼합 연산을 OpenGL API의 색깔 혼합 기능을 사용하여 구현하려 할 경우, 원시 인자(source factor)와 목적 인자(destination factor)로 어떤 값을 사용해야 할까? 다음 OpenGL 상수값들 중 적절한 인자를 선택하라.

`GL_ZERO, GL_ONE`

`GL_SRC_ALPHA`

`GL_DST_ALPHA`

`GL_ONE_MINUS_SRC_ALPHA`

`GL_ONE_MINUS_DST_ALPHA`

(f) 만약 그림 2의 위쪽 그림에서와 같이 뒤에서 앞으로 오면서 (back-to-front) 점진적으로 합성하려 할 경우, 과연 색깔 버퍼의 알파 채널이 필요할까? ‘예/아니오’로 답하고 그 이유를 분명히 답하라.

(g) 만약 OpenGL API를 사용하여 뒤에서 앞으로 오면서 (back-to-front) 점진적으로 합성하려 할 경우, 원시 인자(source factor)와 목적 인자(destination factor)로 어떤 값을 사용해야 할까? 위의 값들 중 적절한 인자를 선택하라.

(h) 그림 4(a)와 (b)에는 두 개의 입력 이미지가 주어져 있다. GIRL 이미지와 NAGEL 이미지 모두 RGBA 타입의 이미지로서, GIRL 이미지의 경우 모든 A 채널 값이 1.0으로 저장되어 있다. 반면 NAGEL 이



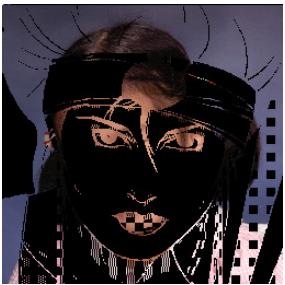
(a) GIRL 이미지



(b) NAGEL 이미지



(c) 합성 예 1



(d) 합성 예 2

그림 4: 색깔 혼합 예

미지의 경우 흰색 부분의 화소에 대해서는 A 채널 값이 0.0으로 저장되어 있고, 흰색을 제외한 나머지 영역의 화소에 대해서는 1.0으로 지정이 되어 있다. 이때 아래와 같은 코드를 사용하여 NAGEL 이미지에서 흰색이 있는 영역을 제외한 나머지 영역 (즉 A 채널 값이 1.0으로 지정되어 있는 부분)의 색깔은 자신의 색깔로, 나머지 영역은 GIRL 이미지의 색깔을 사용하여 그림 4(c) 이미지와 같은 그림을 생성하려 한다. 이때 (A) 와 (B) 에 들어갈 OpenGL 상수를 정확히 기술하라.

```

glEnable(GL_BLEND);
glBlendFunc(GL_ONE, GL_ZERO);
draw_GIRL();
glBlendFunc((A), (B));
draw_NAGEL();
glDisable(GL_BLEND);

```

- (i) 위 문제에 연속하여, 그림 4(d)의 이미지는 NAGEL 이미지에서 흰색이 있는 영

역을 제외한 나머지 영역 (즉 A 채널 값이 1.0으로 지정되어 있는 부분)은 대응되는 GIRL 이미지의 색깔로, 나머지 영역은 검정색으로 칠하도록 합성한 예이다. 이렇게 합성하기 위해서 (A) 와 (B)에 필요한 OpenGL 상수를 정확히 기술하라.

- (j) 마지막 쪽의 그림 9에서 이 문제와 가장 관련이 깊은 박스의 기호를 기술하라.

### 3. 다음은 카메라의 조작에 관한 문제이다.

(a) 지금 광부들이 사용하는 헤드 라이트의 예에서와 같이 카메라에 고정된 점 광원을 사용하려 한다. 만약 광원을 항상 카메라의 기준점에서 위로 3.0만큼 떨어진 곳에 위치시키려하면, 아래의 코드에서 glLightfv(GL\_LIGHT0, GL\_POSITION, li\_pos); 함수를 정확히 어느 지점에서 호출해야하는지 (라인 번호를 사용하여 위치 지정), 그리고 그때 (A), (B), (C), (D)에 들어갈 내용은 무엇인지 기술하라.

```
GLfloat li_pos[4] = {(A), (B),
(C), (D)}; // (1)
```

```
:
```

```
glMatrixMode(GL_MODELVIEW); // (2)
```

```
glLoadIdentity(); // (3)
```

```
gluLookAt(v[0], v[1], v[2],
```

```
c[0], c[1], c[2],
```

```
0.0, 1.0, 0.0); // (4)
```

```
glPushMatrix(); // (5)
```

```
Do modeling transform
```

```
and draw objects
```

```
glPopMatrix(); // (6)
```

- (b) 만약 광원을 세상 좌표계의 (3.0, 0.5, -1.5)인 지점에 고정시킨 후 렌더링을 하려 할 경우에 대해 위 문제를 풀어라.

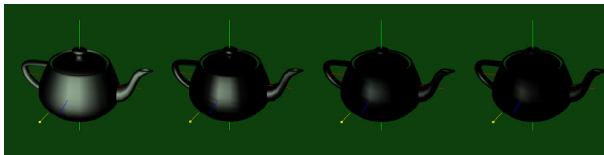


그림 6: 라이팅 계산

4. 다음은 라이팅 계산에 관한 문제이다.
- 점 광원, 평행 광원, 그리고 분산 광원을 비교할 때 계산 비용이 높은 것으로부터 낮은 순서로 나열하라.
  - 퐁 쉐이딩, 플랫 쉐이딩, 그리고 고우러드 쉐이딩을 비교할 때 계산 비용이 높은 것으로부터 낮은 순서로 나열하라.
  - 위의 세 가지 쉐이딩 방법 중 선형보간 (linear interpolation)과 가장 관련이 적은 것은?
  - 그림 5의 식에서 이상한 부분을 지적하고 옳게 고쳐라.
  - 그림 5의 식에서 Lambert의 코사인 법칙을 표현해주는 부분을 정확히 기술하라.
  - 그림 5의 식에서 카메라에서 바라보는 방향에 영향을 받는 변수를 모두 나열하라.
  - 그림 5의 식에서 물체의 위치와 방향이 바뀌었을 때 영향을 받는 변수를 모두 나열하라.
  - 그림 6은 그림 5의 식의 어떤 변수의 성질을 설명하기 위한 것인가?
  - 전통적인 fixed-function 렌더링 파이프라인을 고려할 때, 마지막 쪽의 그림 9에서 이 문제와 가장 관련이 깊은 두 개의 박스의 기호를 기술하라.
5. 다음은 Cg API를 통한 쉐이더 프로그래밍과 풍의 조명 모델에 관한 문제이다. 아래에 주어진 Cg 버텍스 쉐이더 및 텍셀 쉐이더를 보고 답하라. 참고로 이 예에서는 원근 투영과 점광원을 사용하고 있음.

```

struct _output {
 float4 X: TEXCOORD0;
 float3 Y: TEXCOORD1;
 float4 position: POSITION;
};

_output main(float4 A: POSITION,
 float4 B: NORMAL,
 uniform float4x4 ModelViewProj:
 state.matrix.mvp,
 uniform float4x4 ModelView:
 state.matrix.modelview,
 uniform float4x4 ModelViewIT:
 state.matrix.(4).invtrans) {
 _output OUT;

 OUT.X = mul(ModelView, A); // (1)
 OUT.Y = mul(ModelViewIT, B).xyz; // (2)
 OUT.position = mul(ModelViewProj, A); // (3)
 return OUT;
}

=====
struct _output { float3 color: COLOR; };

_output main(float4 X: TEXCOORD0,
 float3 Y: TEXCOORD1,
 uniform float4 g_a:
 state.lightmodel.ambient,
 uniform float4 lp:
 state.light[0].position,
 uniform float4 la:
 state.light[0].ambient,
 uniform float4 ld:
 state.light[0].diffuse,
 uniform float4 ls:
 state.light[0].specular,
 uniform float4 ma:
 state.material.ambient,
 uniform float4 md:
 state.material.diffuse,
 uniform float4 ms:
 state.material.specular,
 uniform float me:
 state.material.shininess) {

```

```

_output OUT;

OUT.color = g_a * ma;
float3 N = normalize(Y);
float3 L = normalize(lp - X);
float NdotL = dot(N, L);
if(NdotL >= 0.0) { // (5)
 OUT.color += ma * la;
 float3 What1 = normalize(-X); // (6)
 float3 What2 = normalize(L + What1); // (7)
 float NdotWhat2 = dot(N, What2);
 float4 li = lit(NdotL, NdotWhat2, me);
 float3 D = md * li.y;
 float3 S = ms * li.z;
 OUT.color += (8);
}
return OUT;
}

```

- (a) 이 프로그램은 각 픽셀을 통해서 보이는 물체의 각 지점에서 풍의 조명 모델을 적용하여 쉐이딩 계산을 하는 풍 쉐이딩 계산을 위한 쉐이더 코드이다. 이 프로그램에서는 OpenGL의 어떤 좌표계에서 쉐이딩 계산을 하고 있을까? OC, MC, WdC, CC, NDC, EC, WC 등의 기호를 사용하여 답하라.
- (b) 보편적인 렌더링 관련 쉐이더 코드 작성 시 버텍스 쉐이더에서는 꼭 해주어야 하는 작업이 하나 있다. 위의 버텍스 쉐이더에서 이 작업과 가장 관련이 깊은 문장의 번호는 (1), (2), (3) 중 어떤 것이며, 이 작업은 구체적으로 무엇을 의미하는지 정확히 기술하라.
- (c) 문맥상 버텍스 쉐이더 코드의 (4) 부분에 들어갈 내용을 정확히 기술하라. 대소문자를 구별할 것.
- (d) (사실 없어도 문제는 없으나) (5)번의 if

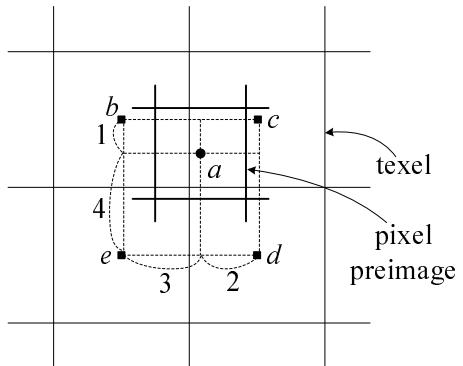


그림 7: 선형 필터

문장은 어떠한 경우에 지역 조명 모델식을 계산하지 않겠다는 것인지 그 상황을 설명하라.

- (e) (6)번 문장의 `What1` 변수에는 정확히 어떤 지점에서 어느 방향을 가리키는 벡터가 저장이 될까?
  - (f) (7)번 문장의 `What2` 변수가 저장하는 벡터를 통칭 무엇이라고 하는가?
  - (g) 문맥상 (8)번에 들어갈 내용을 C 언어 문맥에 맞게 정확히 기술하라.
  - (h) 그림 6은 위의 쉐이더 프로그램에서 어떤 특정 변수 값을 바꾸어가며 조절한 렌더링 효과를 보여주고 있다. 과연 어떤 변수인지 정확히 기술하라.
  - (i) 마지막 쪽의 그림 9에서 위의 쉐이더 코드의 두 번째에 있는 `_output main()` 함수가 수행되는 부분에 해당하는 박스의 기호를 기술하라.
6. 다음은 OpenGL 환경에서의 텍스춰 필터링에 관한 문제이다.
- (a) 레벨 3의 밀립 텍스춰의 한 텍셀이 나타내는 영역의 면적은 레벨 1의 밀립 텍스춰의 한 텍셀이 나타내는 영역의 면적의 몇 배일까? 참고로 원래 주어진 고해상도 이미지의 레벨은 0임.

- (b) 그림 7에서  $a$ 는 픽셀에 대한 원상(pre-image)의 중점을 가리키고,  $b$ 부터  $e$ 까지는  $a$ 를 포함하는 주변 텍셀의 중심점을 나타낸다.  $b, c, d$ , 그리고  $e$  지점의 텍셀 색깔이 각각  $(1, 1, 1), (1, 0, 0), (1, 1, 0)$ , 그리고  $(0, 1, 1)$ 이라고 하자. 이때 선형 필터를 사용할 경우  $a$  지점에 대하여 계산되는 색깔은 무엇일까? 반드시 계산 과정을 밝혀라 (이 그림에서 숫자는 거리에 대한 비율을 나타냄).
- (c) 그림 8(a)는 확대 필터로 `GL_NEAREST`와 `GL_LINEAR` 중의 하나를, 축소 필터로는 다음 네 가지 필터 중 하나를 사용하여 렌더링한 결과이다 (이 과정에서 서로 다른 레벨의 텍스처로 서로 다른 색깔의 이미지를 사용하였음).

```
GL_NEAREST_MIPMAP_NEAREST
GL_LINEAR_MIPMAP_NEAREST
GL_NEAREST_MIPMAP_LINEAR
GL_LINEAR_MIPMAP_LINEAR
```

렌더링 결과를 볼 때, 과연 확대 필터와 축소 필터로 각각 어떤 필터를 사용하였을까?

- (d) 그림 8(a)을 볼 때, 과연 확대 상황이 발생하고 있을까? ‘예/아니오’로 답하고, 아니라면 그 이유를, 그렇다면 구체적으로 어느 부분에서 확대 상황이 발생하고 있는지 간단한 그림을 사용하여 정확히 기술하라.
- (e) 같은 그림에서 육면체의 왼쪽 측면을 보면 서로 다른 색깔의 텍스처가 적용되고 있는데, 파란색과 녹색의 텍스처 중 어떤 색깔의 텍스처의 레벨이 더 높은지, 그리고 그 이유를 설명하라.
- (f) 그림 8(b)는 여섯 개의 사각형으로 이루어진 육면체의 각 면에 대하여 동일한 텍스처를 적용하여 렌더링한 결과이다. 이

예에서는  $[0, 1] \times [0, 1]$  영역의 텍스처 공간 전체에 대한 텍스처 이미지를 각 사각형에 적용되도록, 각 사각형의 꼭지점에 텍스처 좌표를 부여하였다. 반면에 그림 8(c)는 다른 렌더링 인자는 모두 동일한 상황에서, 텍스처 행렬 스택만 조작하여 생성한 결과이다. 즉 다음과 같은 코드를 적절히 삽입하였는데, (A) 안에 들어갈 OpenGL 문장(들)을 기술하라. 이 그림에서 텍스처가 적용된 이미지 영역의 크기가 각 방향 반으로 줄어들었고, 그 중심은 육면체 각 면의 중심에 해당한다. 여기서 텍스처 랩 (wrap) 모드는 `GL_CLAMP`임.

```
glMatrixMode(GL_TEXTURE);
glPushMatrix();
(A)
glMatrixMode(GL_MODELVIEW);
:
```

```
glMatrixMode(GL_TEXTURE);
glPopMatrix();
```

- (g) 마지막 쪽의 그림 9에서 이 문제와 가장 관련이 깊은 박스의 기호를 기술하라.

7. 다음은 임의의 광선  $\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$ 와 세 개의 꼭지점으로 구성된 삼각형  $T = (\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$ 와의 교점을 구해 (물론 교차할 경우),  $T$ 를 기준으로 하는 barycentric coordinates로 표현해주는 것에 관한 문제이다. 여기서 각 벡터의 좌표값은 다음과 같고, 편의상  $\mathbf{d}$ 의 길이는 1임.

$$\mathbf{o} = \begin{pmatrix} o_x \\ o_y \\ o_z \end{pmatrix}, \mathbf{d} = \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix}, \mathbf{p}_i = \begin{pmatrix} p_{xi} \\ p_{yi} \\ p_{zi} \end{pmatrix}$$

- (a) 이 광선과 삼각형의 교점을  $\mathbf{p}(b_1, b_2) = (1 - b_1 - b_2)\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{p}_2$ 와 같이 표

현하려한다. 지금 광선의 출발점에 해당하는  $\mathbf{o}$ 로부터의 거리에 해당하는  $t$ 와 교점의 barycentric coordinates에 해당하는  $b_1$ 과  $b_2$ 를 계산해야하는데, 이는 아래와 같이 3원 1차 연립 방정식을 풀어 해결할 수 있다. 이때 3행 3열 행렬의 9개의 원소  $m_1, m_2, \dots, m_9$ 의 각 값을 정확히 기술하라.

$$\begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{bmatrix} \begin{pmatrix} t \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} o_x - p_{x0} \\ o_y - p_{y0} \\ o_z - p_{z0} \end{pmatrix}$$

- (b)  $\mathbf{s} = \mathbf{o} - \mathbf{p}_0$ ,  $\mathbf{e}_1 = \mathbf{p}_1 - \mathbf{p}_0$ ,  $\mathbf{e}_2 = \mathbf{p}_2 - \mathbf{p}_0$ 라 할 때, 이 연립 방정식의 해는 다음과 같이 표현할 수 있다. 여기서  $\cdot$  과  $\times$ 는 각각 벡터의 내적 (inner product)과 외적 (cross product)를 나타냄.

$$\begin{bmatrix} t \\ b_1 \\ b_2 \end{bmatrix} = \frac{1}{(\mathbf{d} \times \mathbf{e}_2) \cdot \mathbf{e}_1} \begin{bmatrix} (\mathbf{s} \times \mathbf{e}_1) \cdot \mathbf{e}_2 \\ (\mathbf{d} \times \mathbf{e}_2) \cdot \mathbf{s} \\ (\mathbf{s} \times \mathbf{e}_1) \cdot \mathbf{d} \end{bmatrix}$$

이 식을 부주의하게 프로그래밍할 경우, 오버 플로우 에러가 발생할 수 있다. 과연 이런 경우는 기하학적으로 어떠한 상황에 해당하는가?

- (c) 바로 위 문제에서  $t, b_1, b_2$  값을 오버 플로우 에러 없이 계산을 하였다고 가정하자. 이 경우 과연 각 변수 값이 어떤 범위의 값을 가질 때, 광선과 삼각형이 교차한다고 말할 수 있을까?
- (d) 3차원 공간의 두 벡터  $\mathbf{a}$ 와  $\mathbf{b}$ 의 외적  $\mathbf{a} \times \mathbf{b}$ 를 계산하는데 정확히 몇 번의 (덧셈/뺄셈), 곱셈, 그리고 나눗셈 연산이 필요한지를 밝혀라. 외적의 계산식을 통하여 설명하고, 덧셈과 뺄셈 연산은 같은 연산으로 취급할 것.
- (e) 위의 식을 사용하여 한 광선과 삼각형의 교점을 효율적으로 계산하기 위하여  $t, b_1, b_2$ 를 계산하는데 소요되는 (덧셈/뺄셈),

곱셈, 그리고 나눗셈 연산의 회수를 기술하라. (주의: 일반적으로 나눗셈은 연산 비용이 높으므로, 한 벡터의 각 원소를 동일한 값으로 나누어주는 연산은 일단 그 값으로 나눗셈을 한 후 벡터의 각 원소에 곱해주는 방식을 취한다고 가정).

8. 다음은 임의의 광선  $\mathbf{p}(t) = \mathbf{o} + td$ 와 축에 정렬된 바운딩 박스 (Axis-Aligned Bounding Box, AABB)가 교차하는지를 판별해주는 프로그램의 일부이다. 이 코드를 보고 답하라.

```
float tx_min, ty_min, tz_min;
float tx_max, ty_max, tz_max;
float a = 1.0f/aa;
if (a >= 0.0) {
 tx_min = (x0 - p) * a;
 tx_max = (x1 - p) * a;
}
else {
 tx_min = (x1 - p) * a;
 tx_max = (x0 - p) * a;
}
float b = 1.0f/bb;
if (b >= 0.0) {
 ty_min = (y0 - r) * b;
 ty_max = (y1 - r) * b;
}
else {
 ty_min = (y1 - r) * b;
 ty_max = (y0 - r) * b;
}
float c = 1.0f/cc;
if (c >= 0.0) {
 tz_min = (z0 - q) * c;
 tz_max = (z1 - q) * c;
}
else {
 tz_min = (z1 - oz) * c;
}
```

```

tz_max = (z0 - oz) * c;
}

float t0, t1;

if (tx_min > ty_min) t0 = (A);
else t0 = (B);
if (tz_min > t0) t0 = (C);
if (tx_max < ty_max) t1 = tx_max;
else t1 = ty_max;
if (tz_max < t1) t1 = tz_max;
return (t0 < t1 && t1 > 0);

```

- (a) aa, bb, cc가 각각 광선의 방향 벡터  $\mathbf{d}$ 의  $x, y, z$  좌표 값을 의미한다고 할 때, 문맥상 눈의 위치에 해당하는  $\mathbf{o}$ 의  $x, y, z$  좌표를 위 코드의 변수를 사용하여 표현하라.
- (b) 문맥상 (A), (B), 그리고 (B)에 들어갈 내용을 C 언어 문법에 맞게 정확히 기술하라.
- (c) 만약 마지막 문장이 수행될 때,  $t0 < t1$ 이 거짓이라면, 이는 어떤 상황에 해당하는가?
- (d) 만약 마지막 문장이 수행될 때,  $t0 < t1$ 은 참이나  $t1 > 0$ 이 거짓이라면, 이는 어떤 상황에 해당하는가?

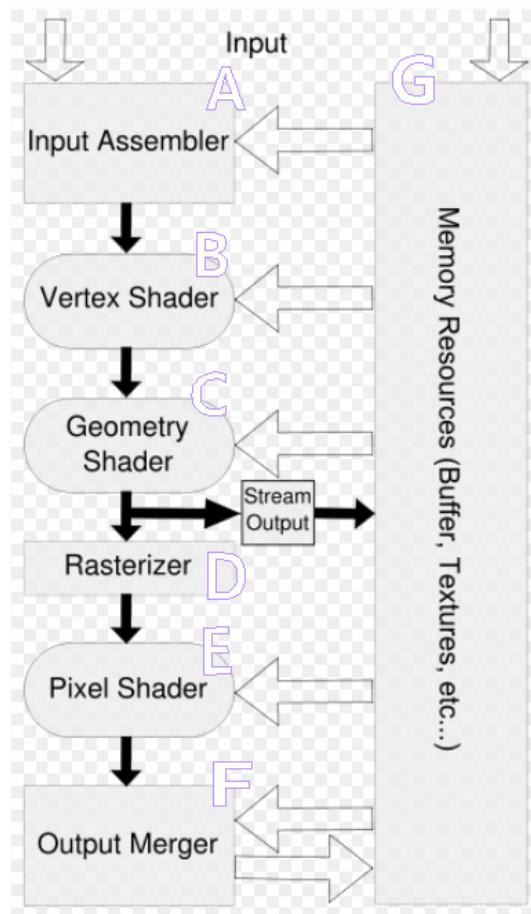
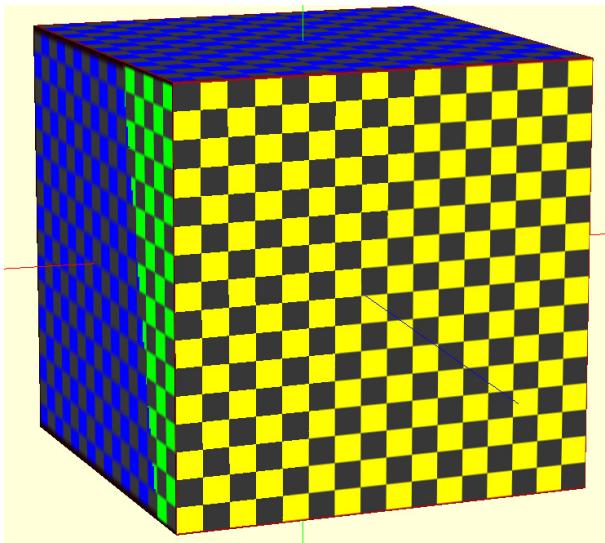


그림 9: Microsoft Direct3D 10 그래픽스 파이프라인 (from Wikipedia)

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \cdot L_i) + k_{s\lambda} \cdot (N \cdot H)^n\}$$

그림 5: 라이팅 계산 수식



(a)



(b)



(c)

그림 8: 텍스춰 필터링

[CSE4170] 기초 컴퓨터 그래픽스 기말고사

담당 교수: 임 인 성

- 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 텍스춰 이미지를 통한 간단한 애니메이션에 관한 문제이다.

- (a) 그림 1(a)는 아래와 같은 코드를 사용하여 물주전자를 그린 그림이다.

```

:
glMatrixMode(GL_MODELVIEW);
glPushMatrix();
glTranslatef(0.8, 0.7, 0.0);
glRotatef(180.0, 0.0, 1.0, 0.0);
glRotatef(-90.0, 1.0, 0.0, 0.0);
glScalef(0.95, 0.95, 0.95);
glMatrixMode(GL_TEXTURE); // (A)
glPushMatrix(); // (B)
glTranslatef(-0.5, 0.0, 0.0); // (C)
glScalef(2.0, 1.0, 1.0); // (D)
draw_object(&teapot);
glPopMatrix(); // (E)
glMatrixMode(GL_MODELVIEW); // (F)
glPopMatrix();
:

```

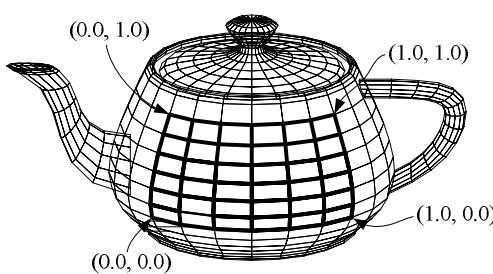


그림 2: 텍스춰 좌표 설정

텍스춰를 그릴 때 GL\_CLAMP 모드를 사용하였고, `draw_object(&teapot);` 문장을 호출할 경우 물체의 각 꼭지점에 텍스춰 좌표를 붙여

물주전자를 그리도록 하였는데, 텍스춰 좌표는 그림 2에 도시된 바와 같이 굵은 선분의 사각형 영역에는  $(s, t)$  좌표 값이 각각 0과 1 사이에 값으로 변화하도록 되어 있으며 나머지 사각형의 꼭지점에 대해서는 모두 (-1.0, -1.0) 값이 붙어 있다. 이때 만약 문장 (A)부터 (F) 까지의 여섯 문장을 제거할 경우, 물주전자 위의 굵은 선분의 사각형 영역에 텍스춰가 어떻게 그려질지 간략한 그림을 그려가며 설명하라.

- (b) 만약 위의 코드 상태에서 GL\_CLAMP 모드를 GL\_REPEAT 모드로 바꾼 후, (C)와 (D)의 문장의 순서를 서로 바꾸어 렌더링하면, 물주전자 위의 굵은 선분의 사각형 영역에 텍스춰가 어떻게 그려질지 간략한 그림을 그려가며 설명하라.
- (c) 지금 그림 1(a)의 상태에서 시작하여 매 시간 프레임마다 텍스춰가 조금씩 오른쪽으로 이동하여 (b)의 상태까지 갔다가 다시 왼쪽으로 조금씩 이동하여 (c)의 상태로 이동한 후, 계속하여 좌우로 반복하여 이동하는 텍스춰 애니메이션 효과를 내려고 하고 있다. 이를 위하여 다음과 같은 코드를 추가하였는데, 이 외에 1.(a) 문제의 코드의 어느 부분을 어떻게 수정하면 원하는 텍스춰 애니메이션 효과를 생성할 수 있을까?

```

float ttt = 0.0;
int mode = 1;
void next(int t) {
 if (mode == 1) {
 if ((ttt += 0.1) >= 1.0)
 mode = -1;
 } else {
 if ((ttt -= 0.1) <= -1.0)
 mode = 1;
 }
 glutPostRedisplay();
 glutTimerFunc(500, next, 1);
}

```



그림 1: 텍스춰 애니메이션

2. 지금 가상의 3차원 실내 공간을 자유롭게 네비게이션 하려하는데, 사용자가 벽면을 뚫고 지나가지 못하도록 Binary Space Partitioning Tree (BSPT)를 사용하여 충돌 검사를 하려한다. 지금 모든 벽면이 사각형으로 모델링이 되어 있는데, 이 사각형들을 사용하여 구성한 BSPT TREE에 대해 아래와 같은 함수를 호출하려한다.

```
LOS(TREE, A, B);
```

이때 원하는 것은 3차원 공간의 두 점 A와 B에 의해 정의된 선분이 실내 공간의 임의의 벽면에 해당하는 사각형과 교차를 하는지 알아내는 것인데, 이 작업은 아래의 코드로 수행할 수 있다. 참고로 BSPT 타입은 다음과 같이 정의되어 있음.

```
typedef struct _BSPT {
 Polygon *poly;
 struct _BSPT *fchild;
 struct _BSPT *bchild;
} BSPT;
```

```
int LOS(BSPT *T, float *start, float *end) {
 int side_s, side_e, side_f, side_b;

 if (!T) return 1;
 side_s = check(start, T->poly);
 side_e = check(end, T->poly);

 if ((side_s == BSPT_FRONT)
 && (side_e == BSPT_FRONT)) {
 return LOS((A) , start, end);
 }
 else if ((side_s == BSPT_BACK)
 && (side_e == BSPT_BACK)) {
 return LOS((B) , start, end);
 }
 else {
 if (intersect(T->poly, start, end))
```

```
 return 0;
 else {
 side_f = LOS(T->fchild, start, end);
 side_b = LOS(T->bchild, start, end);
 return (C) ;
 }
}
```

- (a) 이 함수는 0 또는 1 값을 리턴하도록 되어 있는데, 매크로 상 언제 0 값을, 그리고 언제 1 값을 리턴하는지 정확히 기술하라.  
 (b) 매크로 상 check() 함수가 하는 일이 무엇인지 정확히 기술하라.  
 (c) 매크로 상 (A)와 (B)에 들어갈 내용을 C 언어 문법에 맞게 정확히 기술하라.  
 (d) 매크로 상 (C)에 들어갈 내용을 C 언어 문법에 맞게 정확히 기술하라.

3. 다음은 programmable GPU의 렌더링 파이프라인에 관한 문제이다. 그림 3과 4를 보고 답하라.

- (a) 그림 3에서 원근 투영시 실제로 원근감이 생성되는 시점과 가장 관련이 있는 부분의 기호를 기술하라.  
 (b) 그림 3에서 프래그먼트가 생성되는 부분의 기호를 기술하라.  
 (c) 그림 3에서 은면 제거 계산과 가장 관련이 깊은 부분의 기호를 기술하라.  
 (d) 그림 3에서 OpenGL 상수 GL\_SRC\_ALPHA가 결정적으로 영향을 미치는 부분의 기호를 기술하라.  
 (e) 그림 3에서 OpenGL 상수 GL\_CCW가 결정적으로 영향을 미치는 부분의 기호를 기술하라.

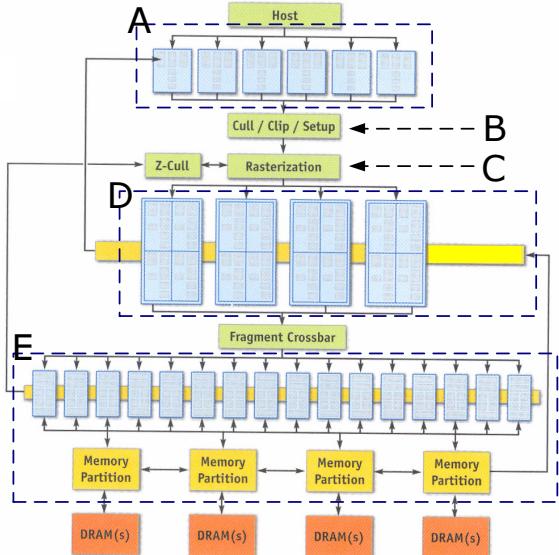


그림 3: NVIDIA GeForce 6 Series GPU 구조

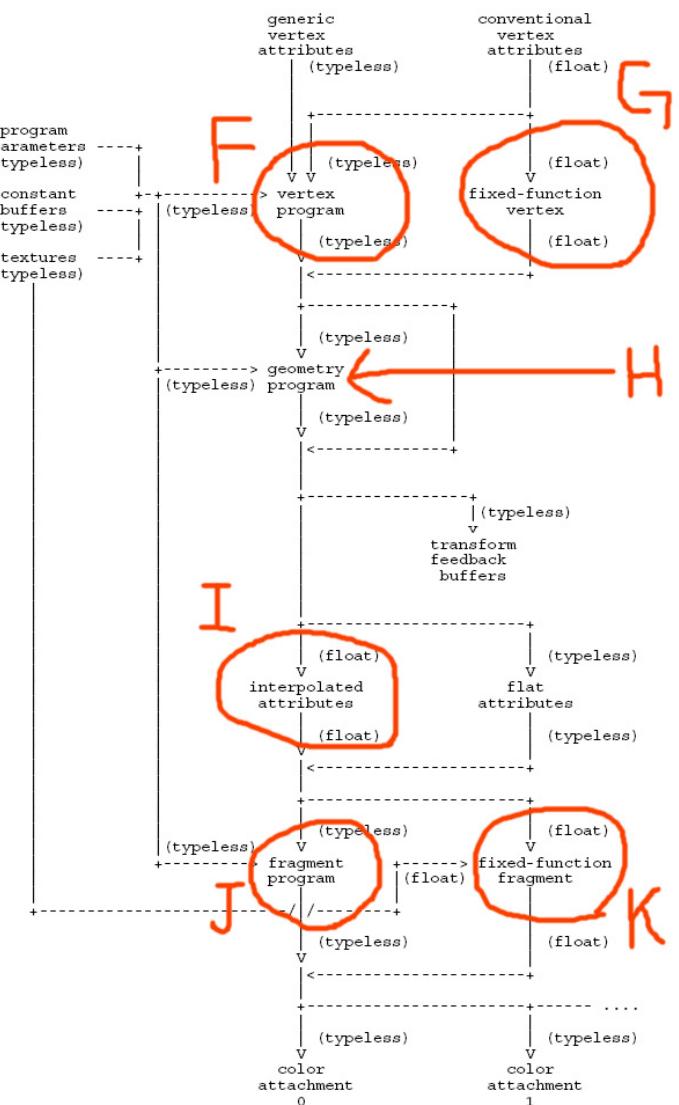


그림 4: NVIDIA G80 specification 사용 파이프라인

4. 다음은 색깔 혼합에 관한 문제이다. ( $c_S \alpha_S$ )와 ( $c_D \alpha_D$ )를 각각 두 이미지 S와 D의 대응되는 화소의 미리 곱한 색깔(pre-multiplied color)이라 할 때, 두 색깔의 합성을 통하여 생성한 결과 색깔은 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 미리 곱한 색깔 (0.6, 0.6, 0.6, 0.6)으로 어떤 화소를 칠한다고 하면, 그것은 그 픽셀을 어떻게 칠한다는 것을 의미하는지 정확히 기술하라.  
 (b) 만약  $F_S = 1 - \alpha_D$ 이고  $F_D = 1$ 일 경우 두 색깔을 어떻게 합성하겠다는 것인지 정확히 기술하라.

(c) 그림 5와 같은 상황에서  $F_S$ 와  $F_D$ 는 각각 얼마인가?

(d) 그림 6의 (a)는 흰색의 사각형을 그린 후 빨간색의 사각형을 그 위에 그린 상황을 보여주고 있다. 만약 (b)에서와 같이 빨간색의 사각형을 그릴 때 이미 흰색의 사각형이 그려진 지역에만 빨간색이 나타나도록 하기 위하여 아래와 같이 프로그램을 작성하였다. 이때 (A)와 (B)에 들어갈 OpenGL 인자를 다음에서 골라라 (여기서 전체 영역은 (0.0, 0.0, 0.0, 0.0)의 색으로 초기화가 되어 있으며, `draw_rectangle()` 함수의 첫 네 인자는 위치, 다음 네 인자는 색깔에 관한 것임).

`GL_ZERO, GL_ONE, GL_SRC_ALPHA,  
GL_DST_ALPHA, GL_ONE_MINUS_SRC_ALPHA,`

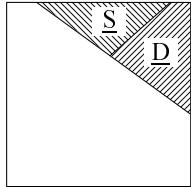
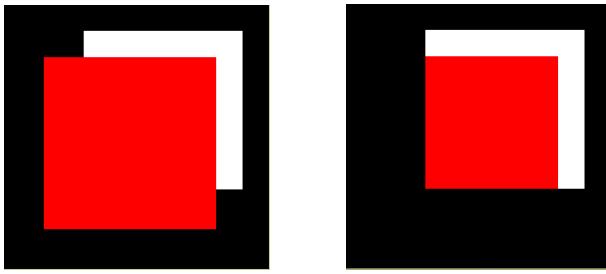


그림 5: 합성 예 1



(a)

(b)

그림 6: 합성 예

## GL\_ONE\_MINUS\_DST\_ALPHA

```
void blend(void) {
 glBlendFunc(GL_ONE, GL_ZERO);
 draw_rectangle(-2.0, 4.0, -2.0, 4.0,
 1.0, 1.0, 1.0, 1.0);
 glBlendFunc(GL_ONE_MINUS_DST_ALPHA, GL_ZERO);
 draw_rectangle(-3.5, 3.0, -3.5, 3.0,
 1.0, 0.0, 0.0, 1.0);
}
```

## 5. 다음은 텍스처 필터링에 관한 문제이다.

- (a) 레벨 3의 mip맵 텍스처의 한 텍셀이 나타내는 영역의 면적은 레벨 5의 mip맵 텍스처의 한 텍셀이 나타내는 영역의 면적의 몇 배일까?
- (b) 해상도가  $512 \times 512$ 이고 각 텍셀 당 GL\_UNSIGNED\_BYTE와 GL\_RGBA 형식을 사용하는 텍스처 이미지에 대하여 모든 가능한 레벨에 대하여 mip맵을 구성할 경우 얼마 만큼의 텍스처 메모리가 필요할까? 계산 과정을 기술할 것.
- (c) 그림 7(a)에서  $a$ 는 픽셀에 대한 원상(pre-image)의 중점을 가리키고,  $b$ 부터  $e$ 까지는  $a$ 를 포함하는 주변 텍셀의 중심점을 나타낸다.  $b, c, d$ , 그리고  $e$  지점의 텍셀 색깔이 각각  $(1, 1, 1), (1, 0, 0), (1, 1, 0)$ , 그리고  $(0, 1, 1)$ 이라고 하자. 각각 최근 필터와 선형 필터를 사용할 경우  $a$  지점에 대하여 계산되는 색깔은 무

엇일까? 이 그림에서 숫자는 거리에 대한 비율을 나타내고 있는데, 반드시 계산 과정을 밝힐 것.

(d) 그림 7(b)는 축소 필터는 GL\_LINEAR\_MIPMAP\_LINEAR를, 확대 필터는 GL\_NEAREST를 사용한 경우의 렌더링 결과이다. 과연 이 그림의 상황은 축소 상황이 발생한 것인지, 아니면 확대 상황이 발생한 것인지 구체적으로 기술하라.

(e) 그림 7(c)는 축소 상황이 발생한 경우이다. 서로 다른 레벨의 mip맵 텍스처에 대해서 서로 다른 색깔의 텍스처 이미지를 사용하였는데, 과연 이 그림은 삼선형 필터(trilinear interpolation filter)를 사용한 결과라 할 수 있는가? 예/아니오로 답하고 그렇게 답한 이유를 밝혀라.

6. 다음은 Cg API를 통한 쉐이더 프로그래밍과 품의 조명 모델에 관한 문제이다. 아래에 주어진 Cg 버텍스 쉐이더 및 픽셀 쉐이더를 보고 답하라. 참고로 이 예에서는 원근 투영과 점광원을 사용하고 있음.

```
struct _output {
 float4 X: TEXCOORD0;
 float3 Y: TEXCOORD1;
 float4 position: POSITION;
};

_output main(float4 A: POSITION,
 float4 B: NORMAL,
 uniform float4x4 ModelViewProj:
 state.matrix.mvp,
 uniform float4x4 ModelView:
 state.matrix.modelview,
 uniform float4x4 ModelViewIT:
 state.matrix.(4).invtrans) {
 output OUT;
}

OUT.X = mul(ModelView, A); // (1)
OUT.Y = mul(ModelViewIT, B).xyz; // (2)
OUT.position = mul(ModelViewProj, A); // (3)
return OUT;
}
```

=====

```
struct _output { float3 color: COLOR; };
_output main(float4 X: TEXCOORD0,
 float3 Y: TEXCOORD1,
 uniform float4 g_a:
 state.lightmodel.ambient,
 uniform float4 lp:
 state.light[0].position,
 uniform float4 la:
 state.light[0].ambient,
```

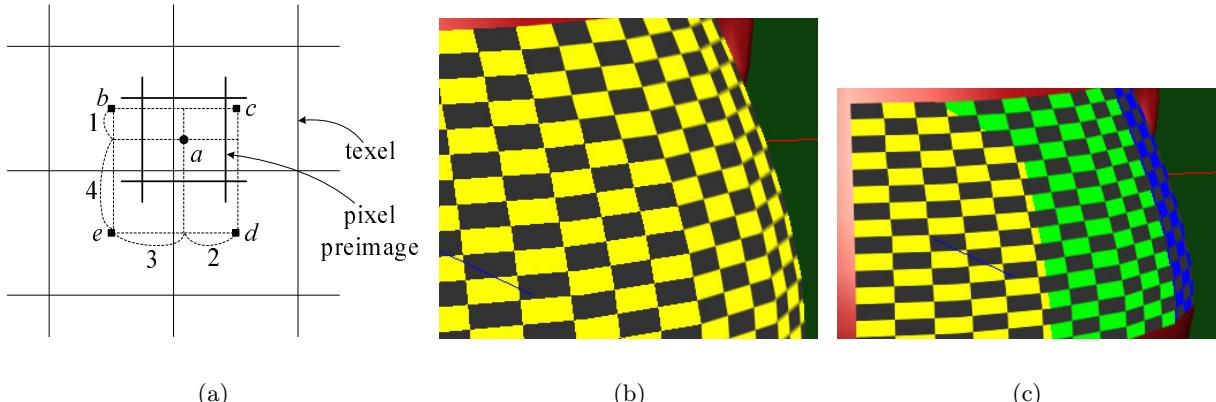


그림 7: 텍스춰 필터링

```

uniform float4 ld:
 state.light[0].diffuse,
uniform float4 ls:
 state.light[0].specular,
uniform float4 ma:
 state.material.ambient,
uniform float4 md:
 state.material.diffuse,
uniform float4 ms:
 state.material.specular,
uniform float me:
 state.material.shininess) {
_output OUT;

OUT.color = g_a * ma;
float3 N = normalize(Y);
float3 L = normalize(lp - X);
float NdotL = dot(N, L);
if(NdotL >= 0.0) { // (5)
 OUT.color += ma * la;
 float3 What1 = normalize(-X); // (6)
 float3 What2 = normalize(L + What1); // (7)
 float NdotWhat2 = dot(N, What2);
 float4 li = lit(NdotL, NdotWhat2, me);
 float3 D = md * li.y;
 float3 S = ms * li.z;
 OUT.color += (8);
}
return OUT;
}

```

- (a) 이 프로그램은 각 픽셀을 통해서 보이는 물체의 각 지점에서 풍의 조명 모델을 적용하여 쉐이딩 계산을 하는 풍 쉐이딩 계산을 위한 쉐이더 코드이다. 이 프로그램에서는 OpenGL의 어떤 좌표계에서 쉐이딩 계산을 하고 있을까? OC, MC, WdC, CC, NDC, EC, WC 등의

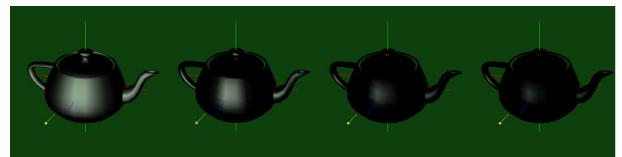


그림 8: 라이팅 계산

기호를 사용하여 답하라.

- (b) 보편적인 렌더링 관련 쉐이더 코드 작성 시 버텍스 쉐이더에서는 꼭 해주어야 하는 작업이 하나 있다. 위의 버텍스 쉐이더에서 이 작업과 가장 관련이 깊은 문장의 번호는 (1), (2), (3) 중 어떤 것이며, 이 작업은 구체적으로 무엇을 의미하는지 정확히 기술하라.
- (c) 문맥상 버텍스 쉐이더 코드의 (4) 부분에 들어갈 내용을 정확히 기술하라. 대소문자를 구별할 것.
- (d) (사실 없어도 문제는 없으나) (5) 번의 if 문장은 어떠한 경우에 지역 조명 모델식을 계산하지 않겠다는 것인지 그 상황을 설명하라.
- (e) (6) 번 문장의 What1 변수에는 정확히 어떤 지점에서 어느 방향을 가리키는 벡터가 저장이 될까?
- (f) (7) 번 문장의 What2 변수가 저장하는 벡터를 통칭 무엇이라고 하는가?
- (g) 문맥상 (8) 번에 들어갈 내용을 C 언어 문맥에 맞게 정확히 기술하라.
- (h) 그림 8은 위의 쉐이더 프로그램에서 어떤 특정 변수 값을 바꾸어가며 조절한 렌더링 효과를 보여주고 있다. 과연 어떤 변수인지 정확히 기술하라.
- (i) 다음은 수업시간에 다룬 풍의 조명 모델의 계산식의 한 예이다. 이 식에서 사용하는 조명

모델과 위의 퍽셀 쉐이더에서 사용하는 조명 모델과의 차이를 두 가지 기술하라.

$$\begin{aligned} I_\lambda &= I_{a\lambda} \cdot k_{a\lambda} \\ &+ I_{l\lambda} \cdot \{k_{d\lambda} \cdot (N \cdot L) + k_{s\lambda} \cdot (N \cdot H)^n\} \end{aligned}$$

- (j) 그림 4에서 위의 쉐이더 코드의 두 번째에 있는 `_output main()` 함수가 수행되는 부분에 해당하는 기호를 기술하라.
7. 다음은 다시 Binary Space Partitioning Tree에 관한 문제이다. 그림 9의 함수 `display_bspt_back_to_front()`는 시점의 위치 `viewer[3]`가 주어졌을 때, 뒤에서 앞으로 (back-to-front) 가면서 bspt의 다각형들을 그려주는 함수이다.

- (a) 이 예제 프로그램의 문맥상 (A)에 들어갈 가장 적절한 내용의 C 코드를 작성하라. 이 함수는 결과로 BSPT\_FRONT 또는 BSPT\_BACK을 리턴해야 한다.
- (b) (B), (C), (D), (E)에 들어갈 내용을 정확한 C 코드 형태로 기술하라.
- (c) bspt가 나타내는 물체를 OpenGL의 블렌딩 기능을 사용하여 투명하게 그려주려 한다. (F)에 들어갈 내용을 정확한 C 코드 형태로 기술하라.
- (d) 위의 문제를 모두 제대로 풀었다는 가정하에, 만약 뒤에서 앞으로 (back-to-front)가 아니라 앞에서 뒤로 가면서 (front-to-back) 다각형들을 그려 주려면 이 프로그램의 어느 부분을 어떻게 고쳐야 할지, 문장 번호를 언급하며 정확히 기술하라.

8. 다음은 카메라의 조작에 관한 문제이다.

- (a) 지금 광부들이 사용하는 것과 같은 헤드 라이트를 사용하고자, 즉 광원을 항상 카메라의 기준점에서 위로 3.0만큼 떨어진 곳에 위치시키려 한다. 이를 위하여 아래의 코드에서 `glLightfv(GL_LIGHT0, GL_POSITION, li_pos);` 함수를 정확히 어느 지점에서 호출해야하는지, 그리고 그때 (A), (B), (C)에 들어갈 내용은 무엇인지 기술하라.

```
GLfloat li_pos[4] = { (A), (B),
(C), 1.0 }; // (1)

:
glMatrixMode(GL_MODELVIEW); // (2)
glLoadIdentity(); // (3)
gluLookAt(v[0], v[1], v[2],
c[0], c[1], c[2],
```

0.0, 1.0, 0.0); // (4)

- (b) 초기 상태의 카메라 프레임이 눈 좌표계의 좌표축과 일치되어 있는 상태에서, 카메라 프레임에 대하여  $M_1 \rightarrow M_2 \rightarrow M_3$  순서로 변환을 하였다면, 그 경우 뷰잉 변환 행렬  $M_V$ 는 무엇인지 기술하라.

- (c) 초기 상태  $C_0 = (e, u, v, n) = ((0\ 0\ 0), (1\ 0\ 0), (0\ 1\ 0), (0\ 0\ 1))$ 인 카메라를 사용자의 조작을 통하여  $C_1 = ((0\ 10\ 10), (0\ 0\ 1), (0\ 1\ 0), (-1\ 0\ 0))$ 와 같은 상태로 변환하는 상황을 아래와 같은 뷰잉 변환 코드로 구현한다고 가정하자.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMultMatrixf(cam.mat);
glTranslatef((A), (B), (C));
```

i. 이때 (A), (B), (C)에 들어갈 내용을 기술하라.

ii. 올바른 변환을 위하여 `cam.mat`이 저장해야 할 4행 4열 변환 행렬의 내용을 정확히 기술하라.

```

typedef struct {
 int nv; // number of vertices
 float *vertex; // pointer to vertex data
 float *normal; // pointer to normal data
 float plane[4]; // plane equation
} Polygon;

typedef struct _bspt { // data structure for bspt
 Polygon *poly;
 struct _bspt *fchild; // for the front side
 struct _bspt *bchild; // for the back side
} BSPT;

int check_side(float *pos, Polygon *poly) {
 (A)
}

BSPT *bspt; // object in BSPT
float viewer[3]; // camera position
:
void display_bspt_back_to_front(BSPT *bspt, float *viewer) {
 int viewer_side; // (1)

 if (bspt == NULL) return; // (2)
 viewer_side = check_side(viewer, bspt->poly); // (3)

 if (viewer_side == BSPT_BACK) { // (4)
 display_bspt_back_to_front((B), viewer); // (5)
 draw_bspt_poly(bspt->poly); // (6)
 display_bspt_back_to_front((C), viewer); // (7)
 }
 else /* viewer_side == BSPT_FRONT */ // (8)
 display_bspt_back_to_front((D), viewer); // (9)
 draw_bspt_poly(bspt->poly); // (10)
 display_bspt_back_to_front((E), viewer); // (11)
 }
}

void display(void) {
 glBlendFunc((F)); // (12)
 glEnable(GL_BLEND); // (13)
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // (14)
 display_bspt_back_to_front(bspt, viewer); // (15)
 glutSwapBuffers();
}

```

그림 9: BSPT 관련 함수

## 기초 컴퓨터 그래픽스 기말고사 (43-170)

담당 교수: 임 인 성

- 답은 반드시 답안지에 기술할 것. 공간이 부족 할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명 기한 후 기술할 것. 그 외의 경우의 답안지 뒤 쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.

1. 다음 단답식 문제에 답하라.

- (a) 카메라의 프레임을 WC의 좌표계와 일치시킨 상태에서, 카메라에 대하여 행렬  $M_1, M_2, M_3$ 가 나타내는 변환을 순서대로 가하여 카메라를 배치한다고 할 때의 뷰잉 변환 행렬  $M_V$ 를 위의 행렬이나 그 것들의 역행렬로 표현하라.
- (b) 하프웨이 벡터 (halfway vector)는 어떤 가정하에 렌더링 계산의 효율을 증대시킬 수 있을까?
- (c) 보편적인 렌더링 작업을 수행할 때, 버텍스 쉐이더가 반드시 해주어야하는 계산이 한 가지 있는데, 과연 이는 무엇일까?
- (d) OpenGL 파이프라인에서 뒷면 제거 (back-face culling) 연산은 어떤 좌표계에서 수행이 되는지, OC, MC, WdC, CC, NDC, EC, WC 등의 기호를 사용하여 답하라.
- (e) 꼭지점별 연산(per-vertex operation)과 화소별 연산(per-pixel operation)의 경계가 되는 렌더링 연산의 이름은 무엇인가?
- (f) 텍스춰 매핑 과정에서 어떤 화소에 대한 텍스춰 공간에서의 원상 (preimage)이란 무엇을 뜻하는 말인가?
- (g) 불투명한 물체에 대해 광선 추적법을 적용할 경우, 주로 정방사 방향에서 입사되는 빛이 물체 표면에서 반사되는 것과 또 다른 부류의 빛이 물체 표면에서 반사되는 것을 고려하는데, 과연 이 후자의 빛은 어떤 빛인가?
- (h) 다음 OpenGL 렌더링 파이프라인의 연산 중 화소별 연산에 해당하는 것의 번호를 모두 나열하라.
  - i. depth buffering

- ii. view volume clipping
  - iii. lighting computation in EC
  - iv. texture mapping
  - v. perspective transformation
  - vi. back-face culling
  - vii. blending
  - (i) 전통적인 OpenGL 파이프라인에서 다음 연산들을 먼저 수행되는 순서대로 번호를 나열하라.
    - i. lighting
    - ii. blending
    - iii. perspective division
    - iv. depth buffering
    - v. texture mapping
    - vi. viewport transformation
  2. 다음은 색깔 혼합에 관한 문제이다. 두 장의 이미지 S와 D의 합성에 대하여 생각하자.  $(c_S \alpha_S)$ 와  $(c_D \alpha_D)$ 를 두 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 칠하는 것인지 정확히 기술하라.
  - (b) 그림 1에 도시하는 합성 연산의 경우  $F_S$ 와  $F_D$ 의 값은 각각 얼마인가?
  - (c) 만약 S over D 연산을 적용한다면, 합성 후  $\alpha_O$ 는 어떤 값을 가질지 그 식을 S와 D의 관련 값을 사용하여 정확히 기술하라.
  - (d) 그림 2와 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

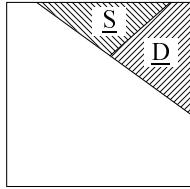


그림 1: 합성 예 1

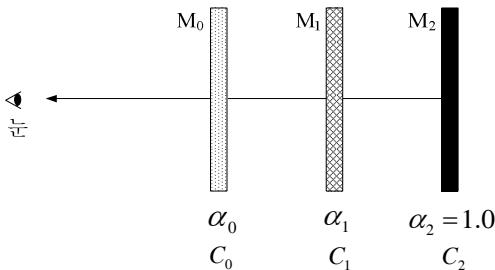


그림 2: 합성 예 2

$C_1$ 이고 빛을  $1 - \alpha_1$ 의 비율 만큼만 통과시키고, 제일 오른쪽에 있는  $M_2$ 는 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  $\alpha_{01}$  값이 무엇일지 정확히 기술하라.

- (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 3(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) {
 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((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 3(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 값들 중 적절한 인자를 선택하라.

GL\_ZERO, GL\_ONE,  
GL\_SRC\_ALPHA,  
GL\_DST\_ALPHA,  
GL\_ONE\_MINUS\_SRC\_ALPHA,  
GL\_ONE\_MINUS\_DST\_ALPHA

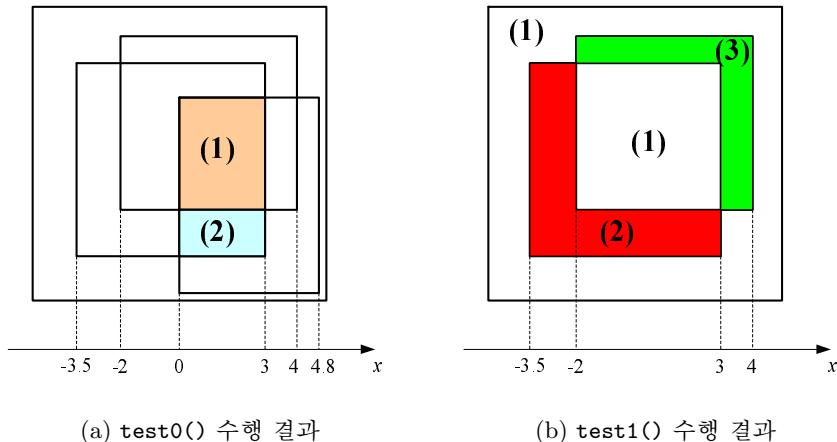


그림 3: 색깔의 혼합

3. 다음은 라이팅 계산에 관한 문제이다.

- (a) 아래의 식은 Phong의 조명 모델의 한 형태를 기술한 것이다.

$$\begin{aligned} I_\lambda &= I_{a\lambda} \cdot k_{a\lambda} \\ &+ \sum_{i=0}^{m-1} f_{att}(\delta_i) \cdot I_{i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) \\ &+ k_{s\lambda} \cdot (R \circ V)^n\} \\ (|N| = |L_i| = |R| = |V| = 1) \end{aligned}$$

수업 시간에 배운 내용을 고려할 때, 잘못된 내용이 있을 경우 모두 수정하라.

- (b) 여기서  $R$ 은 어떤 값일지 위 식에서 사용된 다른 변수를 사용하여 표현하라 (위의 식에서 잘못된 부분을 수정한 변수를 사용할 것).
- (c)  $f_{att}(\delta_i)$ 는 빛의 감쇠 효과를 표현하기 위한 것인데, 이 때  $\delta_i$  변수가 의미하는 값은 무엇인지 정확히 기술하라.
- (d) 그림 4에는 Cg를 사용하여 라이팅 계산을 코딩한 쉐이더 예를 보여주고 있다 (본 프로그램에서는 평행 광원을 사용하고 있음). 문맥 상 버텍스 쉐이더의 `SomeMatrix` 변수에는 어떤 내용의 값이 저장되어 있는지 정확히 기술하라. 참고로 `ModelView` 변수는 모델링 변환 행렬과 뷰잉 변환 행렬의 곱을 저장하고 있다.
- (e) 마찬가지로 버텍스 쉐이더의 `AnotherMatrix` 변수에는 어떤 내용의 값이 저장되어 있는지 정확히 기술하라.

- (f) 문맥 상 (A)에 들어갈 내용을 Cg의 문법에 맞게 기술하라.
- (g) 픽셀 쉐이더의 변수 `computer`는 현재 처리하고자 하는 픽셀을 통하여 보이는 물체 지점에서 어떤 방향을 저장하고 있다. 과연 어떤 방향인지 (그 방향의 이름이 아니라 내용을) 정확히 기술하라.
- (h) 픽셀 쉐이더에서 변수 `computer`를 `normalize(*)` 함수를 통하여 그 길이를 1로 만들어 주고 있다. 이러한 작업을 해주어야 하는 이유는?
- (i) 만약에 위의 변수에 대해 `normalize(*)` 함수를 호출하지 않을 경우 렌더링 결과에 심각한 문제가 발생할까? 예/아니오로 답하고 그 이유를 분명히 밝혀라.
- (j) 반면에 픽셀 쉐이더의 변수 `sogang`에 대해서는 정규화 계산을 수행하지 않는데, 그 이유는 무엇일까?
- (k) `float4 A = lit(a, b, c);` 문장과 같이 `lit()` 함수를 호출하면  $a$ ,  $b$ ,  $c$ 에 대해 적절한 값을 계산하여 벡터  $A$ 에 집어 넣어주게 되는데, 과연  $A.x$ ,  $A.y$ ,  $A.z$ 에는 어떤 값이 저장될지 각각에 대하여 정확히 기술하라. (힌트: 프로그램의 문맥상 어렵지 않게 추측할 수 있음)
- (l) 이 문제 앞에 주어진 조명 모델 식에서 이 픽셀 쉐이더의 밑줄 친 변수 `diffuse`가 저장하고 있는 값과 의미상 가장 부합하는 부분을 정확히 기술하라.
- (m) 위의 픽셀 쉐이더 예에서는 어떤 좌표계에서 라이팅 계산을 하고 있는지, OC,

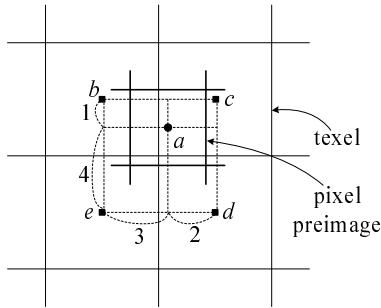


그림 5: 선형 필터

MC, WdC, CC, NDC, EC, WC 등의 기호를 사용하여 답하라.

- (n) 과연 이 쉐이더는 이 문제 처음에 기술한 형태의 Phong의 조명 모델을 사용하고 있는가 (모든 오류를 수정하였다는 가정하에)? 예/아니오로 답하고 그 이유를 설명하라.
4. 다음은 OpenGL 환경에서의 텍스춰 필터링에 관한 문제이다.
- (a) 레벨 3의 mip맵 텍스춰의 한 텍셀이 나타내는 영역의 면적은 레벨 1의 mip맵 텍스춰의 한 텍셀이 나타내는 영역의 면적의 몇 배일까? 참고로 레벨 0이 최고 해상도임.
  - (b) 그림 5에서  $a$ 는 픽셀에 대한 원상(pre-image)의 중점을 가리키고,  $b$ 부터  $e$ 까지는  $a$ 를 포함하는 주변 텍셀의 중심점을 나타낸다.  $b$ ,  $c$ ,  $d$ , 그리고  $e$  지점의 텍셀 색깔이 각각  $(1, 1, 1)$ ,  $(1, 0, 0)$ ,  $(1, 1, 0)$ , 그리고  $(0, 1, 1)$ 이라고 하자. 이때 선형 필터를 사용할 경우  $a$  지점에 대하여 계산되는 색깔은 무엇일까? 반드시 계산 과정을 밝혀라 (이 그림에서 숫자는 거리에 대한 비율을 나타냄).
  - (c) 그림 6(a)는 확대 필터로 `GL_NEAREST`와 `GL_LINEAR` 중의 하나를, 축소 필터로는 다음 네 가지 필터 중 하나를 사용하여 렌더링한 결과이다 (이 과정에서 서로 다른 레벨의 텍스춰로 서로 다른 색깔의 이미지를 사용하였음).

```
GL_NEAREST_MIPMAP_NEAREST,
GL_LINEAR_MIPMAP_NEAREST,
GL_NEAREST_MIPMAP_LINEAR,
GL_LINEAR_MIPMAP_LINEAR
```

이 결과를 볼 때, 과연 확대 필터와 축소 필터로 각각 어떤 필터를 사용하였을까?

(d) 이 그림을 볼 때, 과연 확대 상황이 발생하고 있을까? 아니라면 그 이유를, 그렇다면 구체적으로 어느 부분에서 확대 상황이 발생하고 있는지 간단한 그림을 사용하여 정확히 기술하라.

(e) 같은 그림에서 육면체의 왼쪽 측면을 보면 서로 다른 색깔의 텍스춰가 적용되고 있는데, 파란색과 녹색의 텍스춰 중 어떤 색깔의 텍스춰의 레벨이 더 높은지, 그리고 그 이유를 설명하라.

(f) 그림 6(b)는 여섯 개의 사각형으로 이루어진 육면체의 각 면에 대하여 동일한 텍스춰를 적용하여 렌더링한 결과이다. 이 예에서는  $[0, 1] \times [0, 1]$  영역의 텍스춰 공간 전체에 대한 텍스춰 이미지를 각 사각형에 적용되도록, 각 사각형의 꼭지점에 텍스춰 좌표를 부여하였다. 반면에 그림 6(c)는 다른 렌더링 인자는 모두 동일한 상황에서, 텍스춰 행렬 스택만 조작하여 생성한 결과이다. 즉 다음과 같은 코드를 적절히 삽입하였는데, (A) 안에 들어갈 OpenGL 문장(들)을 기술하라. 이 그림에서 텍스춰가 적용된 이미지 영역의 크기가 각 방향 반으로 줄어들었고, 그 중심은 육면체 각 면의 중심에 해당한다. 여기서 텍스춰 랩 (wrap) 모드는 `GL_CLAMP`임.

```
glMatrixMode(GL_TEXTURE);
glPushMatrix();
(A)
glMatrixMode(GL_MODELVIEW);

:
glMatrixMode(GL_TEXTURE);
glPopMatrix();
```

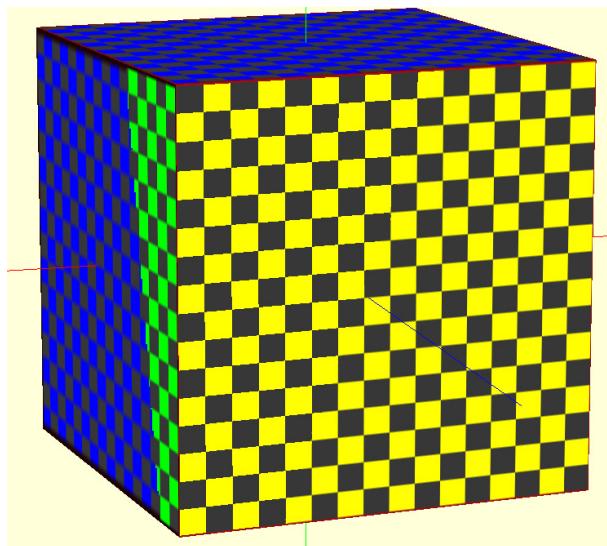
```
struct _output {
 float4 position: POSITION; float4 ppp: TEXCOORD0; float3 nnn: TEXCOORD1;
 float3 L: TEXCOORD2; float3 H: TEXCOORD3;
};

_output main(float4 position: POSITION, float4 normal: NORMAL,
 uniform float4 light_position, uniform float4x4 SomeMatrix,
 uniform float4x4 ModelView, uniform float4x4 AnotherMatrix) {
 _output OUT; OUT.position= mul(SomeMatrix, position);
 OUT.ppp = mul(ModelView, position);
 OUT.nnn = normalize(mul(AnotherMatrix, normal).xyz);
 OUT.L = normalize(light_position - OUT.ppp).xyz;
 float3 V = normalize((A)); OUT.H = normalize(OUT.L + V);
 return OUT;
}

struct _output { float3 color: COLOR; };
_output main(float4 position: TEXCOORD0, float3 graphics: TEXCOORD1,
 float3 sogang: TEXCOORD2, float3 computer: TEXCOORD3,
 uniform float4 global_ambient, uniform float4 light_position,
 uniform float4 light_ambient, uniform float4 light_diff_spec,
 uniform float4 mat_ambient, uniform float4 mat_diffuse,
 uniform float4 mat_specular, uniform float mat_shininess) {
 _output OUT;

 OUT.color = global_ambient * mat_ambient;
 float3 G = normalize(graphics); float GdotS = dot(G, sogang);
 if(GdotS >= 0.0) {
 OUT.color += mat_ambient * light_ambient;
 computer = normalize(computer); float GdotC = dot(G, computer);
 float4 lighting = lit(GdotS, GdotC, mat_shininess);
 float3 diffuse = mat_diffuse * lighting.y;
 float3 specular = mat_specular * lighting.z;
 OUT.color += light_diff_spec * (diffuse + specular);
 }
 return OUT;
}
```

그림 4: Cg 쉐이더를 사용한 라이팅 계산



(a)



(b)



(c)

그림 6: 텍스춰 필터링

## 기초 컴퓨터 그래픽스 기말고사 (43-170)

담당 교수: 임 인 성

- 답은 반드시 답안지에 기술할 것. 공간이 부족 할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명 기한 후 기술할 것. 그 외의 경우의 답안지 뒤 쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.
- 답안지에 학과, 학번, 이름 외에 본인의 암호를 기입하면, 성적 공고 시 학번 대신 암호를 사용 할 것임.

1. 공학인증 관련 설문에 참여하였는가?
2. 다음의 카메라 설정에 관한 문제에 답하라.

- (a) 아래의 코드는 카메라의 위치, 즉 투 영 참조점,에 고정된 헤드 라이트를 설정해주는 예를 보여주고 있다. 이때 `glLightfv(GL_LIGHT0, GL_POSITION, li_pos);` 문장을 정확히 어느 지점에 놓아야 하는지, 그리고 그때 (A), (B), (C)에 들어갈 값은 무엇인지 정확히 기술하라.

```
GLfloat li_pos[4]
= {(A), (B), (C), 1.0 };
:
```

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(v[0], v[1], v[2],
 c[0], c[1], c[2],
 0.0, 1.0, 0.0);
```

- (b) 초기 상태의 카메라 프레임이 눈 좌표계의 좌표축과 일치되어 있는 상태에서, 카메라 프레임에 대하여  $M_1 \rightarrow M_2 \rightarrow M_3$  순서로 변환을 하였다면, 그 경우 뷰잉 변환을 해주는 OpenGL 코드를 `glMultMatrixf()` 함수를 사용하여 작성하라(위의 세 행렬은 각각 `GLfloat` 타입의 포인터 변수 `m1, m2, m3`가 가리키는 곳에 저장되어 있고, 그 행렬들의 역행렬은 각각 `mi1, mi2, mi3`가 가리키는 곳에 저장되어 있음).

- (c) 세상 좌표계의 (10.0, 0.0, 0.0) 점에 위치하는 카메라를 고려하자. 이 지점에서 세상

좌표계를 기준으로 음의  $x$ 축 방향으로 세상을 바라보고 있고, 상이 맷히는 화면을 기준으로 오른쪽은 음의  $z$ 축 방향, 위쪽은 양의  $y$ 축 방향으로 설정이 되어 있다. 이 카메라에 대한 뷰잉 변환을 다음과 같이 하려 할 때, (D)와 (E)에 들어갈 내용을 C/C++ 문법에 맞게 정확히 기술하라.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef((D));
glTranslatef((E));
```

3. 다음은 임의의 광선  $\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$ 와 세 개의 꼭지점으로 구성된 삼각형  $T = (\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$ 와의 교점을 구해 (물론 교차할 경우),  $T$ 를 기준으로 하는 barycentric coordinates로 표현해주는 것에 관한 문제이다. 여기서 각 벡터의 좌표값은 다음과 같고, 편의상  $\mathbf{d}$ 의 길이는 1임.

$$\mathbf{o} = \begin{pmatrix} o_x \\ o_y \\ o_z \end{pmatrix}, \mathbf{d} = \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix}, \mathbf{p}_i = \begin{pmatrix} p_{xi} \\ p_{yi} \\ p_{zi} \end{pmatrix}$$

- (a) 이 광선과 삼각형의 교점을  $\mathbf{p}(b_1, b_2) = (1 - b_1 - b_2)\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{p}_2$ 와 같이 표현하려한다. 지금 광선의 출발점에 해당하는  $\mathbf{o}$ 로부터의 거리에 해당하는  $t$ 와 교점의 barycentric coordinates에 해당하는  $b_1$ 과  $b_2$ 를 계산해야하는데, 이는 아래와 같이 3원 1차 연립 방정식을 풀어 해결할 수 있다. 이때 3행 3열 행렬의 9개의 원소  $m_1, m_2, \dots, m_9$ 의 각 값을 정확히 기술하라 (반드시 유도 과정을 기술할 것).

$$\begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{bmatrix} \begin{pmatrix} t \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} o_x - p_{x0} \\ o_y - p_{y0} \\ o_z - p_{z0} \end{pmatrix}$$

- (b)  $\mathbf{s} = \mathbf{o} - \mathbf{p}_0, \mathbf{e}_1 = \mathbf{p}_1 - \mathbf{p}_0, \mathbf{e}_2 = \mathbf{p}_2 - \mathbf{p}_0$ 라 할 때, 이 연립 방정식의 해는 다음과 같이 표현할 수 있다. 여기서  $\cdot$ 과  $\times$ 는 각각 벡터의 내적 (inner product)과 외적 (cross product)를 나타냄.

$$\begin{bmatrix} t \\ b_1 \\ b_2 \end{bmatrix} = \frac{1}{(\mathbf{d} \times \mathbf{e}_2) \cdot \mathbf{e}_1} \begin{bmatrix} (\mathbf{s} \times \mathbf{e}_1) \cdot \mathbf{e}_2 \\ (\mathbf{d} \times \mathbf{e}_2) \cdot \mathbf{s} \\ (\mathbf{s} \times \mathbf{e}_1) \cdot \mathbf{d} \end{bmatrix}$$

- 이 식을 부주의하게 프로그래밍할 경우, 오버 플로우 에러가 발생할 수 있다. 과연 이런 경우는 기하학적으로 어떠한 상황에 해당하는가?
- (c) 바로 위 문제에서  $t$ ,  $b_1$ ,  $b_2$  값을 오버 플로우 없이 계산을 하였다고 가정하자. 이 경우 과연 각 변수 값이 어떤 범위의 값을 가질 때, 광선과 삼각형이 교차한다고 말할 수 있을까?
- (d) 3차원 공간의 두 벡터  $\mathbf{a}$ 와  $\mathbf{b}$ 의 외적  $\mathbf{a} \times \mathbf{b}$ 를 계산하는데 정확히 몇 번의 (덧셈/뺄셈), 곱셈, 그리고 나눗셈 연산이 필요한지를 밝혀라. 외적의 계산식을 통하여 설명하고, 덧셈과 뺄셈 연산을 같은 연산으로 취급할 것.
- (e) 위의 식을 사용하여 한 광선과 삼각형의 교점을 계산하기 위하여  $t$ ,  $b_1$ ,  $b_2$ 를 계산하는데 소요되는 (덧셈/뺄셈), 곱셈, 그리고 나눗셈 연산의 회수를 구하여라. 각 부분에 대한 회수를 상세히 밝힌 후 각 연산에 대한 총 회수를 계산할 것 (주의: 일반적으로 나눗셈은 연산 비용이 높으므로, 한 벡터의 각 원소를 동일한 값으로 나누어주는 연산은 일단 그 값으로 나눗셈을 한 후 벡터의 각 원소에 곱해주는 방식을 취한다고 가정).
4. 그림 1에는 광선과 구와의 교점을 계산해주는 코드가 주어져 있다.
- (a) 여기서 전역 변수 ray에는 광선의 출발점에 해당하는 ( $ray.eye[0]$ ,  $ray.eye[1]$ ,  $ray.eye[2]$ )와 광선의 방향 ( $ray.dir[0]$ ,  $ray.dir[1]$ ,  $ray.dir[2]$ )이 저장되어 있다. 프로그램의 문맥상  $ray\_hit\_SPHERE(*)$  함수 시작 전에  $obj->expr[0]$ ,  $obj->expr[1]$ ,  $obj->expr[2]$ ,  $obj->expr[3]$  변수들은 각각 어떤 값으로 초기화 되어 있을지 정확히 기술하라.
- (b)  $ray\_hit\_SPHERE(*)$  함수는 obj 변수가 가리키는 구와 ray 변수가 표현해주는 광선과의 교점을 계산하여 적절한 교점이 있으면 그 교점에 대한 매개 변수 t 값을, 아니면 -1 값을 리턴해주는 역할을 한다. 이 함수에서는 변수 A에 1 값을 지정해주고 있는데, 이는 어떠한 조건을 가정하고 있는 것일까?
- (c) 프로그램의 문맥상 (X)에 들어갈 코드 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라.
- (d)  $ray\_hit\_SPHERE(*)$  함수는 line (1)에서 line (4)까지 네 개의 지점에서 함수 값을 리턴해준다. 이 때 광선의 출발점이 구의 내부에 있는 경우 (예를 들어, 유리구 안에서 세상을 바라볼 경우), 이 함수는 어느 지점에서 리턴이 될지 해당 라인 번호를 기술하고, 반드시 그 이유를 정확히 밝혀라.
- (e) 광선을 연결한 직선이 구와는 교차하나, 광선의 출발점의 뒤쪽에서 교차하는 경우, 이 함수는 어느 지점에서 리턴이 될지 해당 라인 번호를 기술하라.
- (f) 만약 광선이 구와 성공적으로 교차를 한다면  $compute\_point\_and\_normal()$  함수에서는 구상의 교차점  $pt[k]$  좌표와 그 점에서의 법선 벡터  $norm[k]$ 를 계산해주게 된다. 이 때 문맥상 (Y)에 들어갈 코드 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라.
5. 그림 2는 Phong 쉐이딩 계산을 위한 Cg의 꼭지점 쉐이더 (vertex shader)의 한 구현 예를 보여주고 있다. 여기서 변수 ModelViewProj, ModelView, ModelViewIT에는 각 변수의 이름에 해당하는 변환 행렬이 연결되어 있다고 가정하고 아래 문제에 답하라.
- (a) 문맥 상 (X)에 들어갈 내용을 Cg의 문법에 맞게 기술하라.
- (b) 문맥 상 (Y)에 들어갈 내용을 Cg의 문법에 맞게 기술하라.
- (c) 문맥 상 이 쉐이더의 수행이 종료하였을 때, 꼭지점의 0번 텍스춰 좌표 속성에는 어떤 기하 성질이 저장될까? 어떤 좌표계에서의 어떤 기하 성질인지를 정확히 밝혀라.
- (d) 문맥 상 이 쉐이더의 수행이 종료하였을 때, 꼭지점의 3번 텍스춰 좌표 속성에는 어떤 기하 성질이 저장될까? 어떤 좌표계에서의 어떤 기하 성질인지를 정확히 밝혀라 (수식이 아니라 말로 설명할 것).
6. 그림 3에는 m개의 광원에 대한 두 가지 형태의 풍의 조명 모델이 주어져 있다.
- (a) 문맥 상 잘못된 곳을 찾아 고쳐라.
- (b) 카메라에서 바라보는 방향에 직접적으로 영향을 받는 변수를 모두 나열하라.

```

typedef enum { SPHERE, BOX, POLYHEDRON } OBJECT_TYPE;

typedef struct _object {
 OBJECT_TYPE obj_type; double *expr; double ka[3], kd[3], ks[3], n;
 struct _object *next;
} OBJECT;
OBJECT *obj_list;

typedef struct _ray { double eye[3], dir[3]; } RAY; RAY ray;

#define SET_RAY_EYE ray.eye[0] = cam.e[0], ray.eye[1] = cam.e[1], ray.eye[2] = cam.e[2];
#define DOT_PROD3(a, b, res) res = a[0]*b[0] + a[1]*b[1] + a[2]*b[2]

double ray_hit_SPHERE(OBJECT *obj) {
 int k; double A, B, C, D, Dsqrt, t; double eminusg[3];

 A = 1;
 for (k = 0; k < 3; k++) eminusg[k] = ray.eye[k] - obj->expr[k];
 DOT_PROD3(ray.dir, eminusg, B); B *= 2.0;
 DOT_PROD3(eminusg, eminusg, C); C -= (X);

 D = B*B - 4.0*A*C;
 if (D < 0.0) return -1.0; // line (1)
 Dsqrt = sqrt(D);
 if ((t = 0.5*(-B - Dsqrt)/A) >= 0.0) return t; // line (2)
 if ((t = 0.5*(-B + Dsqrt)/A) >= 0.0) return t; // line (3)
 return -1; // Line (4)
}

void compute_point_and_normal(OBJECT *obj, double t, double *pt, double *norm) {
 int k;

 if (obj->obj_type == SPHERE) {
 for (k = 0; k < 3; k++) {
 pt[k] = ray.eye[k] + t*ray.dir[k]; norm[k] = (Y);
 }
 }
 else ...
}

void ray_hit(OBJECT **this_obj, double *t) {
 OBJECT *cur_obj; double cur_t;

 *this_obj = NULL; *t = -1.0;
 for (cur_obj = obj_list; cur_obj; cur_obj = cur_obj->next) {
 if (cur_obj->obj_type == SPHERE)
 cur_t = ray_hit_SPHERE(cur_obj);
 else ...
 if (cur_t >= 0.0) {
 if (*t < 0.0) { *t = cur_t; *this_obj = cur_obj; }
 else if (cur_t < *t) { *t = cur_t; *this_obj = cur_obj; }
 }
 }
}

```

그림 1: 광선과 구의 교점 계산

```

struct _output {
 float4 position: POSITION; float4 a: TEXCOORD0; float3 b: TEXCOORD1;
 float3 A: TEXCOORD2; float3 B: TEXCOORD3;
};

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

 OUT.position= mul((X), position);
 OUT.a = mul(ModelView, position);
 OUT.b = normalize(mul(ModelViewIT, normal).xyz);

 OUT.A = normalize(light_position - OUT.a).xyz;
 float3 V = normalize((Y));
 OUT.B = normalize(OUT.A + V);
 return OUT;
}

```

그림 2: Phong 쉐이딩을 위한 꼭지점 쉐이더 예

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (R \circ V)^n\} \quad (1)$$

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) + k_{s\lambda} \cdot (N \circ H_i)^n\} \quad (2)$$

그림 3: Phong의 조명 모델

- (c) 평행 투영과 평행 광원을 사용할 경우 물체의 위치와 방향이 바뀌었을 때 영향을 받는 변수를 모두 나열하라 ( $d_i$ 는 제외).
- (d) 점 광원을 사용할 경우 광원의 위치의 변화에 영향을 받는 변수를 모두 나열하라.
- (e)  $f_{att}(d_i)$ 는 빛의 감쇠 효과를 표현하기 위한 것인데, 이 때  $d_i$  변수가 의미하는 값은 무엇인지 정확히 기술하라.
- (f) 그림 4에는 위의 Cg 꼭지점 쉐이더 문제에 대응하는 퍽셀 쉐이더 (pixel shader)가 주어져 있다. 이 쉐이더는 광원을 한 개만 사용하고 있으며, 빛의 감쇠 효과를 무시하고 있다. 그 외에 이 쉐이더의 내용과 그림 3의 식 (2)의 조명 모델과 가장 큰 차이는 무엇일까?
- (g)  $\text{float4 A} = \text{lit(a, b, c)}$ ; 문장과 같이  $\text{lit()}$  함수를 호출하면  $a, b, c$ 에 대해 적절한 값을 계산하여 벡터  $A$ 에 집어 넣어주게 되는데, 과연  $A.x, A.y, A.z$ 에는 어떤 값이 저장될지 각각에 대하여 정확히 기술하라. (힌트: 프로그램의 문맥상 어렵지 않게 추측할 수 있음)
- (h) 그림 3의 식 (2)의 조명 모델 식에서 이 퍽셀 쉐이더의 밑줄 친 변수  $\text{diffuse}$ 가 저장하고 있는 값과 의미상 가장 부합하는 부분을 정확히 기술하라.

7. 다음은 색깔의 혼합에 관한 문제이다.

- (a) 아래는 OpenGL의 색깔 혼합 기능을 사용하여 뒤에서 앞으로 가면서 (back-to-front) 미리 곱한 색깔 (pre-multiplied color)을 순서대로 섞는데 사용하는 식이다. 이때 S와 D는 각각 원시 색깔 (source color)과 목적 색깔 (destination color)을 나타내는데, 올바른 계산이 수행이 되려면  $\rho_S$ 와  $\rho_D$ 에는 어떤 값이 지정되어야 할까? 아래 식의 변수들을 적절히 사용하여 답하라.

$$\begin{pmatrix} c_O \\ \alpha_O \end{pmatrix} = \rho_S \begin{pmatrix} c_S \\ \alpha_S \end{pmatrix} + \rho_D \begin{pmatrix} c_D \\ 1 \end{pmatrix}$$

- (b) 이번에는 앞에서 뒤로 가면서 (front-to-back) 미리 곱한 색깔을 순서대로 섞는데 사용하는 식이다. 이때 올바른 계산이 수행이 되려면  $\rho_S$ 와  $\rho_D$ 에는 어떤 값이 지정되어야 할까? 아래 식의 변수들을 적절히 사용하여 답하라.

$$\begin{pmatrix} c_O \\ \alpha_O \end{pmatrix} = \rho_S \begin{pmatrix} \alpha_D C_D \\ \alpha_D \end{pmatrix} + \rho_D \begin{pmatrix} \alpha_S C_S \\ \alpha_S \end{pmatrix}$$

- (c) 아래의 프로그램에서 함수 `test0()`을 수행시킬 경우 그림 5(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) {
 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((A), (B));
 rect(-3.5, 3.0, -3.5, 3.0, 1.0, 0.0,
 0.0, 1.0);
 glDisable(GL_BLEND);
}
```

- (d) 다음 함수 `test1()`을 수행시켰을 때 그림 5(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,`

```

struct _output { float3 color: COLOR; };

_output main(float4 position: TEXCOORD0, float3 normal: TEXCOORD1, float3 L: TEXCOORD2,
 float3 H: TEXCOORD3, uniform float4 global_ambient, uniform float4 light_position,
 uniform float4 light_ambient, uniform float4 light_diff_spec, uniform float4 mat_ambient,
 uniform float4 mat_diffuse, uniform float4 mat_specular, uniform float mat_shininess) {
 _output OUT;

 OUT.color = global_ambient*mat_ambient;

 float3 N = normalize(normal);
 float NdotL = dot(N, L);
 if (NdotL >= 0.0) {
 OUT.color += mat_ambient*light_ambient;
 H = normalize(H);
 float NdotH = dot(N, H);
 float4 lighting = lit(NdotL, NdotH, mat_shininess);
 float3 diffuse = mat_diffuse*lighting.y;
 float3 specular = mat_specular*lighting.z;

 OUT.color += light_diff_spec*(diffuse + specular);
 }
 return OUT;
}

```

그림 4: Phong 쉐이딩을 위한 픽셀 쉐이더 예

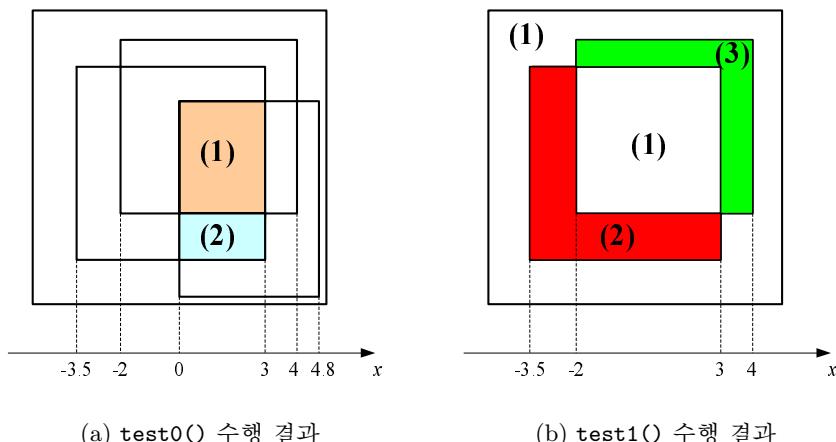


그림 5: 색깔의 혼합

GL\_ONE\_MINUS\_SRC\_ALPHA,  
GL\_ONE\_MINUS\_DST\_ALPHA

8. 다음은 OpenGL 환경에서의 텍스춰 필터링에 관한 문제이다.

- (a) 레벨 0 (최고 해상도)의 mip맵 텍스춰의 한 텍셀이 나타내는 영역의 면적은 레벨 3의 mip맵 텍스춰의 한 텍셀이 나타내는 영역의 면적의 몇 배일까?
- (b) 그림 6(a)에서  $a$ 는 픽셀에 대한 원상(pre-image)의 중점을 가리키고,  $b$ 부터  $e$ 까지는  $a$ 를 포함하는 주변 텍셀의 중심점을 나타낸다.  $b, c, d$ , 그리고  $e$  지점의 텍셀 색깔이 각각  $(1, 1, 1), (1, 0, 0), (1, 1, 0)$ , 그리고  $(0, 1, 1)$ 이라고 하자. 이때 선형 필터를 사용할 경우  $a$  지점에 대하여 계산되는 색깔은 무엇이 될까? 반드시 계산 과정을 밝혀라 (이 그림에서 숫자는 거리에 대한 비율을 나타냄).
- (c) 그림 6(b)는 축소 필터로 GL\_LINEAR를, 확대 필터로는 GL\_NEAREST를 사용한 경우의 렌더링 결과이다. 지금 축소 및 확대 두 필터가 동시에 적용이 되고 있는데, 과연 이러한 상황에는 왼쪽과 오른쪽 부분에 각각 어떤 필터가 적용되고 있을까? 그리고 그 이유는 무엇일까?
- (d) 그림 6(c)는 축소 상황이 발생한 경우이다. 서로 다른 레벨의 mip맵 텍스춰에 대해서 서로 다른 색깔의 텍스춰 이미지를 로드한 후, 축소 필터로 GL\_LINEAR\_MIPMAP\_NEAREST를 사용한 결과인데, 만약 이 축소 필터 대신에 GL\_LINEAR\_MIPMAP\_LINEAR를 사용할 경우 렌더링 결과가 어떻게 달라질까?

9. 그림 7에는 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) OpenGL의 glBlendFunc() 함수가 직접적으로 영향을 미치는 부분은?
- (b) 일반적으로 모델링 변환이 수행되는 부분은?
- (c) 원근 나눗셈 연산이 수행되는 부분은?
- (d) 프래그먼트가 최초로 생성되는 부분은?
- (e) OpenGL의 gluPerspective() 함수로 설정한 뷰 볼륨의 밖에 존재하는 기하 프리미티브들이 제거가 되는 부분은?

(f) 쉐이더를 통하여 Phong 쉐이딩을 구현할 때, Phong의 조명 모델이 적용되는 부분은?

(g) 텍스춰 매핑 계산이 수행되는 부분은?

(h) 정상적인 원근 투영 변환을 할 때, 꼭지점의 동차 좌표의 W 좌표값이 1이 아닌 값이 나타날 수 있는 부분은?

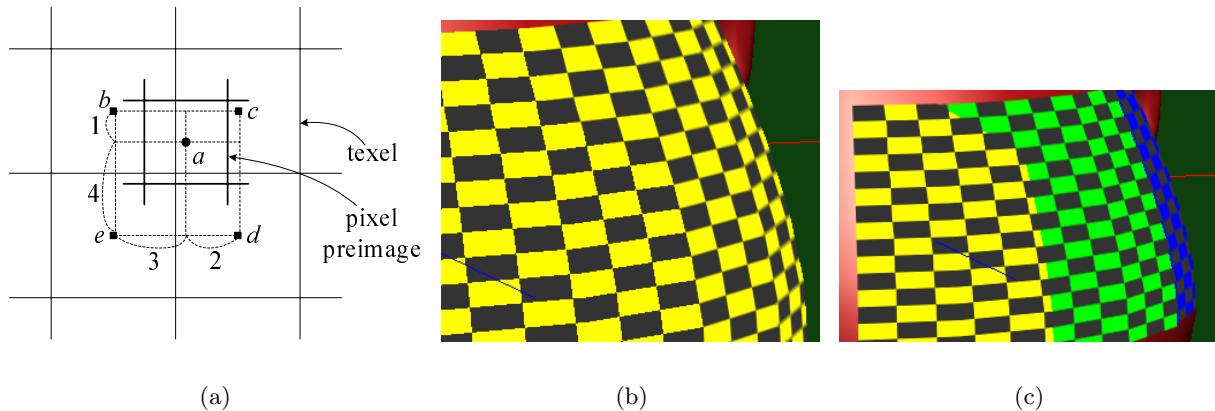


그림 6: 텍스춰 필터링

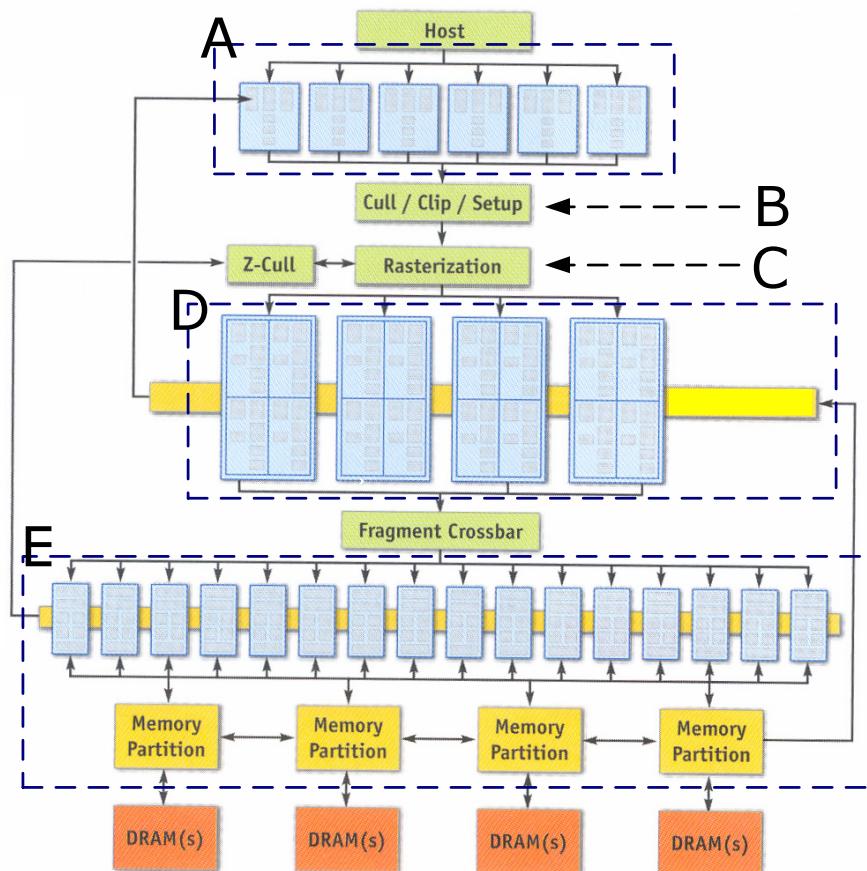


그림 7: NVIDIA GeForce 6 Series GPU 구조

## 기초 컴퓨터 그래픽스 기말고사 (43-170)

담당 교수: 임 인 성

- 답은 반드시 답안지에 기술할 것. 공간이 부족 할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명 기한 후 기술할 것. 그 외의 경우의 답안지 뒤 쪽이나 연습지에 기술한 내용은 답안으로 인정 안함.
- 답안지에 학과, 학번, 이름 외에 본인의 암호를 기입하면, 성적 공고 시 학번 대신 암호를 사용 할 것임.

1. 그림 1은 계층성을 이용하여 자동차를 모델링 해주는 예를 보여주고 있다. 또한 그림 3의 프로그램은 이러한 계층적 구조를 OpenGL을 사용하여 구현한 예이다.

- 이 프로그램은 그림 1에 주어진 자동차에 대한 트리 구조를 어떤 방식으로 탐색을 하고 있는가? 자료 구조 시간에 배운 용어를 사용할 것.
- 그림 1의 트리의 한 변에 붙어있는 행렬  $M_w^1$ 의 내용을  $T(x, y, z)$ ,  $S(x, y, z)$ ,  $R(a, x, y, z)$  등의 이동 변환, 크기 변환, 회전 변환 행렬을 사용하여 정확히 기술하라.
- 프로그램의 문맥상 그림 3의 프로그램의 Line (a)와 Line (b)는 무엇을 위한 변환인지 각각에 대하여 정확히 답하라 (그림에서 오른쪽 방향과 위쪽 방향은 각각  $x$ 와  $y$ 축 방향에 해당하며, 오른손 좌표계를 사용하고 있음).
- 이 프로그램에서 `draw_car()` 함수의 호출 직전의 모델뷰 행렬 스택의 내용이 그림 2와 같다면, `draw_wheel_and_nut()` 함수 안에서 첫 번째로 `draw_nut()` 함수가 호출 (wheel 0의 nut 0에 대한)되기 직전의 이 스택의 내용을 정확히 그려라 (아래의 문제에서 잘못된 곳을 수정하기 전의 내용임). 값을 아는 변수나 식은 모두 상수로 바꾸어라. 위 문제에서와 같이  $T(x, y, z)$ ,  $S(x, y, z)$ ,  $R(a, x, y, z)$  등의 이동 변환, 크기 변환, 회전 변환 행렬을 사용하여 기술할 것.

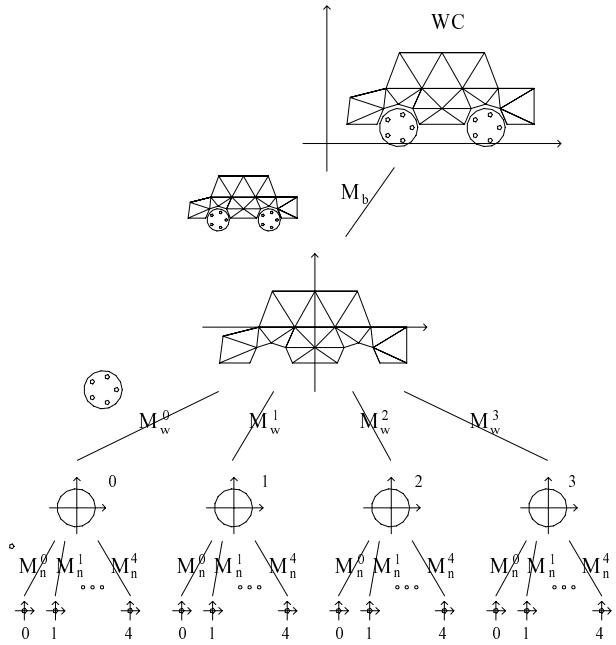


그림 1: 자동차의 계층적 표현

- 프로그램의 문맥상 그림 3의 프로그램에서 잘못된 곳을 하나 찾아 수정하라.

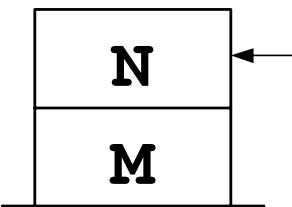


그림 2: 모델뷰 행렬 스택의 내용

```
void draw_wheel_and_nut(float angle) {
 int i;

 draw_wheel();
 for (i = 0; i < 5; i++) {
 glPushMatrix();
 glRotatef(72.0*i, 0.0, 1.0, 0.0);
 glTranslatef(rad-0.5, 0, ww); // rad = 1.7, ww = 1.0
 draw_nut();
 glPopMatrix();
 }
}

void draw_car(void) {
 float angle_y = 0.0, angle_z = 0.0;

 angle_y = wheel_rot_angle_in_y();
 angle_z = wheel_rot_angle_in_z();

 draw_body();

 glPushMatrix();
 glTranslatef(-3.9, -3.5, 4.5);
 glRotatef(angle_y, 0.0, 1.0, 0.0); // Line (a)
 glRotatef(angle_z, 0.0, 0.0, 1.0); // Line (b)
 draw_wheel_and_nut(angle_y); // wheel 0
 glPopMatrix();

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

 // draw wheel 2 here

 // draw wheel 3 here
}
```

그림 3: 계층성을 이용한 자동차 그리기

2. 다음은 라이팅과 쉐이딩에 관한 문제이다. 아래의  $m$ 개의 광원에 대한 풍의 조명 모델과 그림 4의 쉐이더 프로그램을 보고 답하라.

$$\begin{aligned} I_\lambda &= I_{a\lambda} \cdot k_{a\lambda} \\ &+ \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \circ L_i) \\ &\quad + k_{s\lambda} \cdot (N \circ H_i)^n\} \end{aligned}$$

- (a) 변수  $I_{a\lambda}, k_{a\lambda}, d_i, I_{l_i\lambda}, k_{d\lambda}, N, L_i, k_{s\lambda}, H_i$ ,  $n$ 중 카메라의 위치에 영향을 받는 변수를 모두 나열하라.
  - (b) 물체의 위치와 방향이 바뀌었을 때 영향을 받는 변수를 모두 나열하라.
  - (c) 광원의 위치 변화에 영향을 받는 변수를 모두 나열하라.
  - (d)  $H_i$ 는 어떤 상황에서 특히 효율적으로 사용이 되는가? 두 가지 조건을 명시하라.
  - (e) 플랫 쉐이딩, 풍 쉐이딩, 그리고 고우러드 쉐이딩 등 세 가지 방법을 계산량이 적은 것부터 순서대로 나열하라.
  - (f) 위의 세 가지 쉐이딩 방법 중 그림 4의 쉐이더는 어떤 방법을 구현한 것일까?
  - (g) 그림 4(b)의 픽셀 쉐이더의 경우 라이팅 계산에 사용되는 기하 벡터들은 MC, WC, EC, CC, NDC, WdC와 같은 OpenGL 좌표계중 어느 좌표계에서의 의미를 가질까?
  - (h) 그림 4(b)의 픽셀 쉐이더를 보면 위의 풍의 조명 모델식에서의  $N \circ L_i$ 에 해당하는 부분을 계산한 후, 그 값에 따라 처리를 다르게 하고 있다. 과연 무엇을 어떻게 하고 있는 것일까? 그것의 직관적인 의미는?
  - (i) 이 픽셀 쉐이더에서 위의 풍의 조명 모델식의 변수  $n$ 에 해당하는 상수/변수의 이름은?
  - (j) 프로그램의 문맥상 이 픽셀 쉐이더의 Line (a)의 괄호 안에 들어갈 내용을 정확히 기술하라.
  - (k) 위의 풍의 조명 모델식에서 이 픽셀 쉐이더의 Line (b)의 변수 `specular`가 저장하고 있는 값과 의미상 가장 부합하는 부분을 정확히 기술하라.
3. 다음은 색깔 혼합(blending)에 관한 문제이다. 두 장의 이미지 A와 B의 합성에 대하여 생각

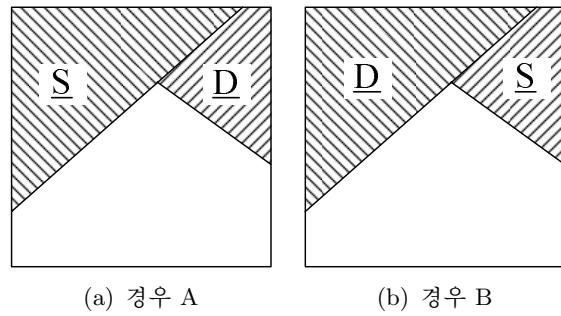


그림 5: 두 가지 혼합의 경우

하자.  $(c_A \alpha_A)$ 와  $(c_B \alpha_B)$ 를 두 이미지의 대응되는 화소의 미리 곱한 색깔(pre-multiplied color)이라 하자.  $F_A$ 와  $F_B$ 를 각각 A와 B의 화소에서  $\alpha_A$ 와  $\alpha_B$ 의 비율로 존재하는 미립자들 중 결과 이미지 O에 살아 남는 미립자의 비율이라 하면, 합성의 결과 생성되는 미리 곱한 색깔은 다음과 같다.

$$\begin{pmatrix} c_O \\ \alpha_O \end{pmatrix} = F_A \begin{pmatrix} c_A \\ \alpha_A \end{pmatrix} + F_B \begin{pmatrix} c_B \\ \alpha_B \end{pmatrix}$$

- (a) 투명한 물체를 렌더링하거나, 안개 등의 기상 효과, 텍스처의 혼합, 앤티앨리어싱 기법들을 구현하는데 유용하게 사용이 되는 A over B 연산의 경우  $F_A$ 와  $F_B$ 는 각각 얼마인가?
- (b) 다음과 같이 여섯 개의 이미지 A, B, C, D, E, F에 대하여 A over B over C over D over E over F 같은 합성 연산을 생각하자. 이 다섯 개의 over 연산자를 임의의 순서대로 선택하여 계산해도 항상 같은 결과가 나오려면, over 연산자에 대하여 결합 법칙이 성립해야 한다. 과연 이 법칙이 성립하는지 또는 아닌지를 수학적으로 증명하라 (Hint:  $(a + b) + c = a + (b + c)$ ).
- (c) 약간 투명한 물체들을 카메라를 기준으로 하여 앞에서 뒤로 가면서 (front-to-back) 차례대로 그려주려 한다. 이때 RGBA 탑입의 색깔 버퍼에 이미 저장되어 있는 미리 곱한 색깔  $(c_D \alpha_D) = (\alpha_D C_D \alpha_D)$ 과 현재 렌더링 파이프라인을 타고 들어오는 프래그먼트 색깔  $(c_S \alpha_S) = (\alpha_S C_S \alpha_S)$ 와 합성을 해주려 한다. 이때 그림 5의 두 가지 경우 중 이 문제에 더 적합한 경우는 A와 B 중 어떤 것일까?
- (d) 합성의 결과 새로이 생성되는 색깔을  $(c_O \alpha_O) = (\alpha_O C_O \alpha_O)$ 라 할 경우  $\alpha_O$ 의 식을 기술하라.

```

struct _output {
 float4 position: POSITION;
 float4 pEC: TEXCOORD0;
 float3 nEC: 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);
 OUT.pEC = mul(ModelView, position);
 OUT.nEC = mul(ModelViewIT, normal).xyz;

 return OUT;
}

```

(a) 버텍스 쉐이더

```

struct _output { float3 color: COLOR; };

_output main(float4 position: TEXCOORD0, float3 normal: TEXCOORD1,
 uniform float nv_flag, uniform float4 global_ambient,
 uniform float4 light_position, uniform float4 light_ambient,
 uniform float4 light_diff_spec, uniform float4 mat_ambient,
 uniform float4 mat_diffuse, uniform float4 mat_specular,
 uniform float mat_shininess) {
 _output OUT;
 OUT.color = global_ambient * mat_ambient;

 float3 N = normalize(normal);
 float3 L = normalize(light_position - position);
 float NdotL = dot(N, L);
 if (NdotL >= 0.0) {
 OUT.color += mat_ambient * light_ambient;
 float3 V = normalize(-position);
 float3 H = normalize(____); // Line (a)
 float NdotH = dot(N, H);

 float4 lighting = lit(NdotL, NdotH, mat_shininess);

 float3 diffuse = mat_diffuse * lighting.y;
 float3 specular = mat_specular * lighting.z;

 OUT.color += light_diff_spec * (diffuse + specular); // Line (b)
 }
 return OUT;
}

```

(b) 퍼셀 쉐이더

그림 4: 버텍스 쉐이더와 퍼셀 쉐이더 예

- (e) 이때 합성된 화소에 칠해줄 실제 색깔(미리 곱한 색깔이 아닌)  $C_O$ 에 대한 식을  $\alpha_D$ ,  $\alpha_S$ ,  $C_D$ ,  $C_S$  등으로 표현하라.
- (f) 위 세 문제에서의 혼합 연산을 OpenGL의 색깔 혼합 기능을 사용하여 구현하려 할 경우, 원시 인자(source factor)와 목적 인자(destination factor)로 어떤 값을 사용해야 할까? 다음 값들 중 적절한 인자를 선택하라.

```
GL_ZERO, GL_ONE
GL_SRC_ALPHA,
GL_DST_ALPHA,
GL_ONE_MINUS_SRC_ALPHA,
GL_ONE_MINUS_DST_ALPHA
```

- (g) 그림 6에는 세 개의 이미지가 주어져 있다. BRICK 이미지와 NAGEL 이미지 모두 RGBA 타입의 이미지로서, BRICK 이미지의 경우 모든 A 채널 값이 1.0으로 저장되어 있다. 반면 NAGEL 이미지의 경우 흰색 부분의 화소에 대해서는 A 채널 값이 0.0으로 저장되어 있고, 흰색을 제외한 나머지 영역의 화소에 대해서는 1.0으로 지정이 되어 있다. 이때 아래와 같은 코드를 사용하여 NAGEL 이미지의 색깔이 있는 영역(즉 A 채널 값이 1.0으로 지정되어 있는 부분)의 색깔을 BRICK 이미지의 색깔로 대치하여, 결과 이미지와 같은 그림을 생성하려 한다. 이때 `glBlendFunc(*, *)`; 함수 호출에 사용되는 두 개의 인자 를 순서대로 정확히 기술하라.

```
draw_NAGEL_image();
 glEnable(GL_BLEND);
 glBlendFunc(_____, _____);
 draw_BRICK_image();
 glDisable(GL_BLEND);
```

4. 다음은 NVIDIA사의 R. Fernando 등이 작성한 *Programming Graphics Hardware*이라는 문서의 일부분이다. 괄호 안에 들어갈 영어 단어를 정확하게 기술하라.

Texture filtering is used to compute the color of a screen pixel based on its footprint in the texture map. A pixel of the texture map is usually referred as a (**A**). When a screen pixel covers one (**A**) or less - texture (**B**) -, its color is taken as the closest (**A**) from the pixel's footprint center, or is computed by (**C**) filtering, that



(a) BRICK 이미지 (b) NAGEL 이미지



(c) 결과 이미지

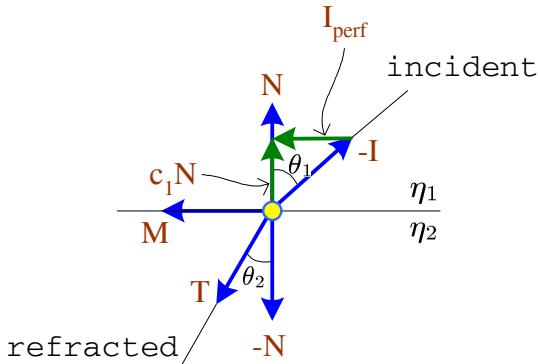
그림 6: 색깔 혼합 예

is (**C**) interpolation of the four closest (**A**s). When it covers several (**A**s) - texture (**D**) -, (**E**) is the preferred solution: Precomputed lower resolution versions of the original texture map - called (**F**) levels - are stored along with the full resolution version and the right (**F**) level is selected to come down back to the (**B**) case.

(**G**) filtering is when (**C**) filtering is performed twice based on two consecutive (**F**) levels and the results are averaged together. In addition to the filtering method, (**H**) filtering can also be optionally selected when performing a texture lookup. (**H**) filtering increases quality for the cases where the pixel's footprint is elongated in one direction: It consists in performing the filtering computations above at several points in the pixel's footprint along this direction.

5. 다음은 광선 추적법에 관한 문제이다.

- (a) 그림 7은  $\eta_1$ 의 굴절 지수를 가지는 1번 매질을 통하여  $I$  벡터 방향으로 들어오는 광선이  $\eta_2$ 의 굴절 지수를 가지는 2번 매질로 들어갈 때, Snell의 법칙을 사용하여 굴절 방향  $T$ 를 구하는 과정을 보여주고 있다 (여기서  $I$ ,  $N$ ,  $M$ ,  $T$ 는 모두 길이가 1인 벡터임).  $\theta_1$  각도는 쉽게 알 수가 있는

그림 7: 굴절 방향  $T$ 의 계산

- 데, 이 때 미지의 각  $\theta_2$ 에 대해  $\sin \theta_2$  값을 알고 있는 값들을 사용하여 표현하라 ( $I$ ,  $N$ ,  $\eta_1$ ,  $\eta_2$ ,  $\theta_1$ 들이 알고 있는 값들임).
- $T$  벡터는  $T = \sin \theta_2 M - \cos \theta_2 N$ 과 같이 표현할 수 있는데, 이때  $\cos \theta_2$  값을  $\eta_1$ ,  $\eta_2$ ,  $\cos \theta_1$ 의 값을 사용하여 유도하라 ( $M$ 은  $N$ 에 수직인 벡터임).
  - 그림 8에는 광선 추적법 코드 중 광선과 구와의 교점을 구하는 부분이 주어져 있다. 여기서 전역 변수 ray에는 광선의 출발점 (`ray.eye[0]`, `ray.eye[1]`, `ray.eye[2]`)과 광선의 방향 (`ray.dir[0]`, `ray.dir[1]`, `ray.dir[2]`)이 저장되어 있다. 프로그램의 문맥상 `ray_hit_SPHERE(*)` 함수 내에서 `obj->expr[0]`, `obj->expr[1]`, `obj->expr[2]`, `obj->expr[3]` 변수들은 각각 어떤 값을 저장하고 있는지 정확히 기술하라.
  - 프로그램의 문맥상 Line (a)의 밑줄 친 부분에 들어갈 코드 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라.
  - `ray_hit_SPHERE(*)` 함수는 `obj` 변수가 가리키는 구와 `ray` 변수가 나타내는 광선과의 교점을 계산하여 적절한 교점이 있으면 그 교점에 대한 매개 변수 `t` 값을, 아니면 `-1` 값을 리턴해주는 역할을 한다. 이 함수에서 어떠한 가정이 성립하지 않을 경우 Line (b)에서 변수 `A`에 1 값을 지정해주는 것에 문제가 발생할지, 그 전제 조건을 정확히 기술하라.
  - 프로그램의 문맥 상 Line (c)의 밑줄 친 부분에 들어갈 코드 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라.
  - 프로그램의 문맥 상 Line (d)의 밑줄 친

부분에 들어갈 코드 내용을 C/C++ 언어 문법에 맞게 정확히 기술하라.

- 그림 9에는 최신 그래픽스 카드인 NVIDIA GeForce 6 Series의 구조가 도시되어 있다. 각 문제에 대하여 가장 연관성이 높은 파이프라인의 영역 이름 (A, B, C, D, E 중의 하나)을 기술하라.
  - 실제로 텍스춰 맵핑 계산이 수행되는 부분은?
  - 연속 공간의 기하 프리미티브들이 이산 공간의 정보로 변하는 부분은?
  - 위의 문제에서 다른 색깔 혼합이 실제로 이루어지는 부분은?
  - 뷰잉 변환이 수행되는 부분은?
  - 뷰폿 변환이 수행되는 부분은?
  - Hierarchical modeling과 가장 관련이 많은 부분은?
  - Depth buffering 기법을 사용하여 어느 물체의 어느 부분이 카메라를 통하여 보이게 되는지를 최종적으로 결정하는 부분은?
  - 전형적인 fixed-function pipeline 중의 하나인 90년대의 OpenGL 파이프라인에서 풍의 조명 모델이 적용되는 부분은?
  - 위 문제의 OpenGL 파이프라인에서 고우라드 쉐이딩 기법을 적용할 때 꼭지점을 따라온 색깔들이 각각의 해당 픽셀들의 값으로 변하게 되는 부분은?
  - 프로그래머가 설정한 3차원 세상의 기하 프리미티브들 중 뷔 볼륨 밖의 것들이 제거되는 부분은?
  - OpenGL의 `GL_LINEAR_MIPMAP_LINEAR` 인자와 가장 관련이 많은 부분은?

```

typedef enum { SPHERE, BOX, POLYHEDRON } OBJECT_TYPE;

typedef struct _object {
 OBJECT_TYPE obj_type; double *expr; double ka[3], kd[3], ks[3], n;
 struct _object *next;
} OBJECT;
OBJECT *obj_list;

typedef struct _camera { double e[3], u[3], v[3], n[3]; double fovy, aspect; } CAMERA;
CAMERA cam;

typedef struct _window {
 int w, h; double d; double wc[3], wh, ww; double a[3], b[3], c[3]; unsigned char *pixmap;
} WINDOW;
WINDOW win;

typedef struct _ray { double eye[3], dir[3]; } RAY;
RAY ray;

#define SET_RAY_EYE ray.eye[0] = cam.e[0], ray.eye[1] = cam.e[1], ray.eye[2] = cam.e[2];
#define DOT_PROD3(a, b, res) res = _____ // Line (a)

double ray_hit_SPHERE(OBJECT *obj) {
 int k; double A, B, C, D, Dsqrt, t; double eminusg[3];

 A = 1; // Line (b)
 for (k = 0; k < 3; k++) eminusg[k] = ray.eye[k] - obj->expr[k];
 DOT_PROD3(ray.dir, eminusg, B); B *= _____; // Line (c)
 DOT_PROD3(eminusg, eminusg, C); C -= obj->expr[3]*obj->expr[3];

 D = B*B - 4.0*A*C;
 if (D < 0.0) return -1.0;
 Dsqrt = sqrt(D);
 if ((t = 0.5*(-B - Dsqrt)/A) >= 0.0) return t;
 if ((t = 0.5*(-B + Dsqrt)/A) >= 0.0) return t;
 _____; // Line (d)
}

void ray_hit(OBJECT **this_obj, double *t) {
 OBJECT *cur_obj; double cur_t;

 *this_obj = NULL; *t = -1.0;
 for (cur_obj = obj_list; cur_obj; cur_obj = cur_obj->next) {
 if (cur_obj->obj_type == SPHERE)
 cur_t = ray_hit_SPHERE(cur_obj);
 else
 ;
 if (cur_t >= 0.0) {
 if (*t < 0.0) { *t = cur_t; *this_obj = cur_obj; }
 else if (cur_t < *t) { *t = cur_t; *this_obj = cur_obj; }
 }
 }
}

```

그림 8: 광선과 구의 교점 계산

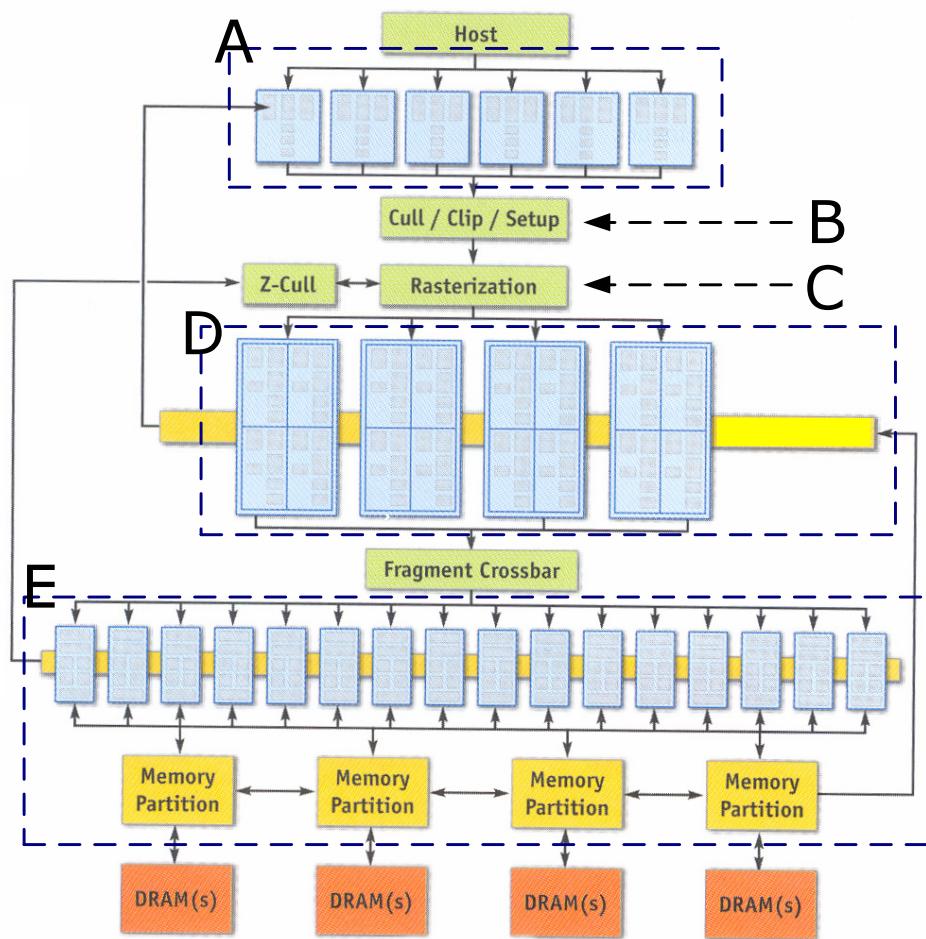


그림 9: NVIDIA GeForce 6 Series Architecture

기초 컴퓨터 그래픽스(43-170) 기말고사

담당교수: 서강대학교 컴퓨터학과 임인성

1. 다음 문제에 간략히 답하라.
  - (a) OpenGL 파이프라인에서 원근 나눗셈은 어느 좌표계와 어느 좌표계 사이에서 일어나는 계산인가?
  - (b) 전통적인 OpenGL 파이프라인에서 고우러드 쉐이딩을 할 경우 풍의 조명 모델이 적용되는 좌표계는?
  - (c) 다음 그래픽스 계산들을 전통적인 OpenGL 파이프라인에서 수행되는 순서대로 나열하라: depth test, backface culling, viewing transform, perspective division, texture mapping
  - (d) 만약 NVIDIA GeForceFX와 같은 programmable GPU상에서 풍쉐이딩을 구현한다면, 물체를 구성하는 각 꼭지점에 어떤 벡터 세 개를 붙어서 래스터화 과정으로 보내줘야 할까?
  - (e) 수업 시간에 배운 그래픽스 계산 중 programmable GPU가 제공하는 픽셀 쉐이더와 가장 밀접한 연관이 있는 계산은 어떤 것일까?
  - (f) 마지막 6번 문제에 있는 BSPT와 Quake 3에서 사용하는 BSPT의 구조를 비교해 볼 때, 가장 큰 차이점 한 가지를 들어라.
  - (g) BSPT로 표현된 물체의 다각형을 시점이 결정되었을 때 앞에서 뒤로 가면서 (front-to-back) 정렬 (sorting)하는 문제를 생각하자. BSPT가  $n$ 개의 다각형으로 구성되어 있을 때, 한 번 정렬하는데 필요한 시간의 복잡도는?
  - (h) 일반적 라이팅 계산을 할 때 물체의 기반인 되는 색깔은 어떤 반사를 위한 물체 색깔에 지정을 할까?
  - (i) 고우러드 쉐이딩과 풍 쉐이딩의 가장 큰 차이는 다면체 모델 표면의 어떤 지점에서 라이팅 계산을 하는가이다. 고우러드 쉐이딩이 물체의 각 꼭지점 지점마다 라이팅 계산을 하는 것이라면, 풍 쉐이딩은 물체 표면의 어떤 지점마다 라이팅 계산을 하는 것일까?
  - (j) 텍스춰 매핑 과정에서 어떤 화소에 대한 원상 (preimage)이란 무엇을 뜻하는 말인가?
2. 그림 1은 E. Haines 등이 저술한 Real-Time Rendering이라는 책에 있는 그림이다. 3차원 게임이나 가상현실 소프트웨어 등 실시간 그래픽스 소프트웨어는 개념적으로 이 그림에서와 같이 세 단계로 그래픽스 데이터가 처리된다고 할 수 있다. 아래의 각 과정/개념은 각각 위의 어느 단계와 가장 밀접한 관계가 있는지 밝혀라 (Application은 A로, Geometry는 G로, Rasterizer는 R로 기술하라).
  - (a) Pixel shader
  - (b) BSPT
  - (c) Blending
  - (d) Perspective division
  - (e) Viewing transformation
  - (f) Collision detection
  - (g) Vertex shader
  - (h) Texture mapping
  - (i) Depth buffering
  - (j) Backface culling
3. 다음은 카메라의 조작에 관한 문제이다.
  - (a) 다음의 코드를 보고 답하라.
 

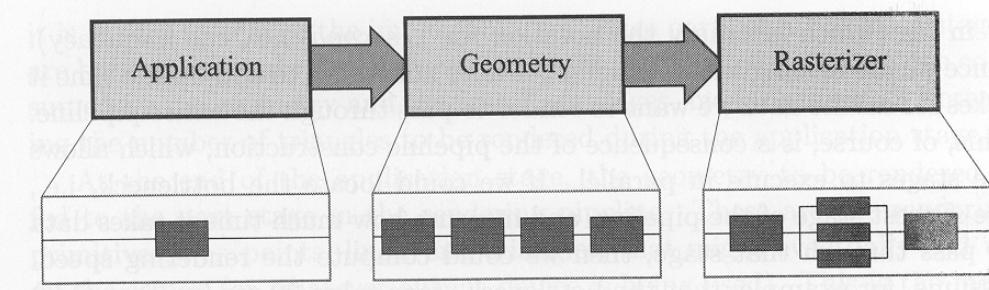
```
GLfloat li_pos[4] = {(A), (B),
 (C), 1.0};

 :
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
 gluLookAt(v[0], v[1], v[2],
 c[0], c[1], c[2],
 0.0, 1.0, 0.0);
```

지금 헤드 라이트를 사용하고자, 즉 광원을 항상 카메라의 위치에 놓고자 한다.

```
glLightfv(GL_LIGHT0, GL_POSITION,
 li_pos);
```

함수를 정확히 어느 지점에서



**Figure 2.2.** The basic construction of the rendering pipeline, consisting of three stages: application, geometry, and the rasterizer. Each of these stages may be a pipeline in itself, as illustrated below the geometry stage, or a stage may be (partly) parallelized as shown below the rasterizer stage. In this illustration, the application stage is a single process, but this stage could also be pipelined or parallelized.

그림 1: 렌더링 파이프라인

호출해야하는지, 그리고 그때 (A), (B), (C)에 들어갈 내용은 무엇인지 기술하라.

- (b) 초기 상태의 카메라 프레임이 눈 좌표계의 좌표축과 일치되어 있는 상태에서, 카메라는 물체에 대하여  $M_1 \rightarrow M_2 \rightarrow M_3$  순서로 변환을 하였다면, 그 경우 뷔잉 변환을 해주는 OpenGL 코드를 `glMultMatrixf()` 함수를 사용하여 작성하라(위의 세 행렬은 각각 GLfloat 타입의 포인터 변수 `m1`, `m2`, `m3`가 가리키는 곳에 저장되어 있고, 그 행렬들의 역행렬은 각각 `mi1`, `mi2`, `mi3`가 가리키는 곳에 저장되어 있음.).

- (c) 어떤 카메라가 세상 좌표계의 (10.0, 0.0, 0.0)에 해당하는 위치에 놓여 있다. 이 지점에서 세상 좌표계를 기준으로 음의  $y$ 축 방향으로 세상을 바라보고 있고, 상이 맷히는 화면을 기준으로 오른쪽은 양의  $x$ 축 방향, 위쪽은 음의  $z$ 축 방향으로 설정이 되어 있다. 이 카메라에 대한 뷔잉 변환을 다음과 같이 하려 할 때, (A)와 (B)에 들어갈 내용을 기술하라.

```

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef((A));
glTranslatef((B));

```

4. 다음은 색깔 혼합에 관한 문제이다. 두 장의 이미지 S와 D의 합성에 대하여 생각하자. ( $c_S \alpha_S$ )와 ( $c_D \alpha_D$ )를 두 이미지의 대응되는 화소의 미리 곱한 색깔(pre-multiplied color)이라

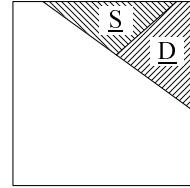
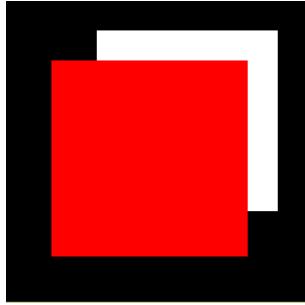


그림 2: 합성 예 1

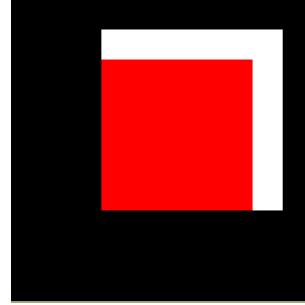
하자. 두 색깔의 합성을 통하여 생성한 결과 색깔을 다음과 같이 계산한다고 하자.

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

- (a) 만약  $(0.8, 0.0, 0.0, 0.8)$ 인 색깔로 어떤 화소를 칠한다고 하면, 그것은 그 픽셀을 어떻게 칠하는 것일까?  
 (b) 만약  $F_S = 1 - \alpha_D$ 이고  $F_D = 1$ 일 경우 두 색깔을 어떻게 합성하는 것인지 정확히 기술하라.  
 (c) 그림 2과 같은 상황에서  $F_S$ 와  $F_D$ 는 각각 얼마인가?  
 (d) 그림 3의 (a)는 흰색의 사각형을 그린 후 빨간색의 사각형을 그 위에 그린 상황을 보여주고 있다. 만약 (b)에서와 같이 빨간색의 사각형을 그릴 때 이미 흰색의 사각형이 그려진 지역에만 빨간색이 나타나도록 하기 위하여 그림 4와 같이 프로그램을 작성하였다. 이때 (A)와 (B)에 들어갈 OpenGL 인자를 다음에서 골라라(여기서 전체 영역은  $(0.0, 0.0, 0.0, 0.0)$ 의 색으로 초기화가 되어 있다고 가정함).



(a)



(b)

그림 3: 합성 예 2

```
void blend(void) {
 glBlendFunc(GL_ZERO, GL_ZERO);
 draw_rectangle(-2.0, 4.0, -2.0, 4.0, 1.0, 1.0, 1.0, 1.0);
 glBlendFunc((A) , (B));
 draw_rectangle(-3.5, 3.0, -3.5, 3.0, 1.0, 0.0, 0.0, 1.0);
}
```

그림 4: 합성 관련 함수

GL\_ZERO, GL\_ONE,  
GL\_SRC\_ALPHA, GL\_DST\_ALPHA,  
GL\_ONE\_MINUS\_SRC\_ALPHA,  
GL\_ONE\_MINUS\_DST\_ALPHA

## 5. 다음은 라이팅 계산에 관한 문제이다.

- (a) 점 광원, 평행 광원, 그리고 분산 광원을 비교할 때 계산 비용이 낮은 것으로부터 높은 순서대로 나열하라.
- (b) 풍 쉐이딩, 플랫 쉐이딩, 그리고 고우러드 쉐이딩을 비교할 때 계산 비용이 낮은 것으로부터 높은 순서대로 나열하라.
- (c) 카메라의 위치와 점광원의 위치가 각각  $(2.0, 3.0, 1.0)$ 과  $(-2.0, 4.0, -1.0)$ 이고, 좌표가  $(0.0, 0.0, 0.0)$ 인 곡면 위의 점  $P$ 에서의 법선 벡터  $N$ 이  $(0.0, 1.0, 0.0)$ 일 때, 정반사 계산에 쓰이는 반사 벡터  $R$ 의 값을 구하라 (반드시 계산 과정을 기술하라).
- (d) 아래의 식에서 이상한 부분을 모두 지적하고 옳게 고쳐라.

$$I_\lambda = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N_i \cdot L_i) + k_{s\lambda} \cdot (N_i \cdot H_i)^n\}$$

## 6. 다음은 Binary Space Partitioning Tree에 관한 문제이다. 그림 5의 함수

`display_bspt_back_to_front()`는 시점의 위치 `viewer[3]`가 주어졌을 때, 뒤에서 앞으로 (back-to-front) 가면서 `bspt`의 다각형들을 그려주는 함수이다.

- (a) 이 예제 프로그램의 문맥상 (A)에 들어갈 가장 적절한 내용의 C 코드를 작성하라. 이 함수는 결과로 BSPT\_FRONT 또는 BSPT\_BACK을 리턴해야 한다.
- (b) (B), (C), (D), (E)에 들어갈 내용을 정확한 C 코드 형태로 기술하라.
- (c) `bspt`가 나타내는 물체를 OpenGL의 블렌딩 기능을 사용하여 투명하게 그려주려한다. (F)에 들어갈 내용을 정확한 C 코드 형태로 기술하라.

```
typedef struct {
 int nv; // number of vertices
 float *vertex; // pointer to vertex data
 float *normal; // pointer to normal data
 float plane[4]; // plane equation
} Polygon;

typedef struct _bspt { // data structure for bspt
 Polygon *poly;
 struct _bspt *fchild; // for the front side
 struct _bspt *bchild; // for the back side
} BSPT;

int check_side(float *pos, Polygon *poly) {
 (A)
}

BSPT *bspt; // object in BSPT
float viewer[3]; // camera position
:
void display_bspt_back_to_front(BSPT *bspt, float *viewer) {
 int viewer_side;

 if (bspt == NULL) return;
 viewer_side = check_side(viewer, bspt->poly);

 if (viewer_side == BSPT_BACK) {
 display_bspt_back_to_front((B) , viewer);
 draw_bspt_poly(bspt->poly);
 display_bspt_back_to_front((C) , viewer);
 }
 else { /* viewer_side == BSPT_FRONT */
 display_bspt_back_to_front((D) , viewer);
 draw_bspt_poly(bspt->poly);
 display_bspt_back_to_front((E) , viewer);
 }
}

void display(void) {
 glBlendFunc((F));
 glEnable(GL_BLEND);
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 display_bspt_back_to_front(bspt, viewer);
 glutSwapBuffers();
}
```

그림 5: BSPT 관련 함수

## 기초 컴퓨터 그래픽스(43-170) 기말고사

담당교수: 서강대학교 컴퓨터학과 임 인 성

1. 다음 문제에 답하라.

- (a) OpenGL 파이프라인에서 풍의 조명 모델 계산이 수행되는 좌표계는?
- (b) 다음 렌더링 연산을 OpenGL 파이프라인에서 수행되는 순서대로 나열하라.
  - (1) 래스터화, (2) 뷰잉 변환, (3) 라이팅 계산, (4) 텍스춰 매핑
- (c) 다음 렌더링 연산을 OpenGL 파이프라인에서 수행되는 순서대로 나열하라.
  - (1) 텍스춰 매핑, (2) 뷰잉 볼륨에 대한 절단, (3) 색깔 혼합(blending), (4) 깊이 버퍼링
- (d) OpenGL 렌더링 파이프라인에서 텍스춰 매핑 모듈로 홀러들어가는 프래그먼트를 구성하는 4가지 정보는 무엇인가?
- (e) 다음 그래픽스 계산 중 OpenGL 파이프라인에서 구현이 되어 있지 않은 계산을 모두 나열하라.
  - (1) 풍의 조명 모델의 적용, (2) BSPT 계산, (3) 충돌 검사, (4) 은면 제거, (5) 텍스춰 좌표의 변환, (6) 투영 변환
- (f) 꼭지점별 연산(per-vertex operation)과 픽셀별 연산(per-pixel operation)의 경계가 되는 렌더링 연산은 무엇인가?
- (g) 텍스춰 축소 상황에서 최근 필터보다 이선형 필터가 좋은 결과를 낳고는 하는데, 축소의 정도가 심해져도 과연 그러할까? 답을 말하고 그 이유를 설명하라.

2. 다음은 카메라의 조작에 관한 문제이다.

- (a) 초기 상태, 즉 카메라 프레임이 눈 좌표계 좌표축과 일치되어 있는 상태에서, 카메라는 물체에 대하여  $M_1 \rightarrow M_2 \rightarrow M_3$  순서로 변환을 하였다면, 그 경우 뷰잉 변환을 해주는 OpenGL 코드를 작성하라(위의 세 행렬은 각각 GLfloat 타입의 포인터 변수  $m1$ ,  $m2$ ,  $m3$ 가 가리키는 곳에 저장되어 있고, 그 행렬들의 역행렬은 각각  $mi1$ ,  $mi2$ ,  $mi3$ 가 가리키는 곳에 저장되어 있음.).



그림 2: 라이팅 계산

- (b) 초기 상태로 주어진 카메라  $C_0 = ((0\ 0\ 0), (1\ 0\ 0), (0\ 1\ 0), (0\ 0\ 1))$ 인 카메라가 사용자 조작을 통하여  $C_1 = ((-10\ 0\ 0), (0\ 0\ -1), (0\ 1\ 0), (1\ 0\ 0))$ 와 같은 상태로 변환이 되었을 때의 아래와 같은 뷰잉 변환 코드를 생각하자.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glMultMatrixf(cam.mat);
glTranslatef((A), (B),
(C));
```

- i. (A), (B), 그리고 (C)에 들어갈 내용은?
- ii. 올바른 변환을 위하여 cam.mat이 저장해야 할 4행 4열 변환 행렬은?

3. 다음은 라이팅 계산에 관한 문제이다.

- (a) 점 광원, 평행 광원, 그리고 분산 광원을 비교할 때 계산 비용이 높은 것으로부터 낮은 순서로 나열하라.
- (b) 풍 쇼이딩, 플랫 쇼이딩, 그리고 고우러드 쇼이딩을 비교할 때 계산 비용이 높은 것으로부터 낮은 순서로 나열하라.
- (c) 위의 세 가지 쇼이딩 방법 중 LERP과 가장 관계가 적은 것은?
- (d) 그림 1의 식에서 이상한 부분을 지적하고 옳게 고쳐라.
- (e) 그림 2는 그림 1의 식의 어떤 변수와 가장 관련이 깊을까?
- (f) 그림 1의 식에서 카메라에서 바라보는 방향에 영향을 받는 변수를 모두 나열하라.

$$I_\lambda = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \cdot L_i) + k_{s\lambda} \cdot (N \cdot H)^n\}$$

그림 1: 라이팅 계산 수식

- (g) 그림 1의 식에서 물체의 위치와 방향이 바뀌었을 때 영향을 받는 변수를 모두 나열 하라.
4. 다음은 색깔 혼합에 관한 문제이다. 두 장의 이미지 A와 B의 합성에 대하여 생각하자.  $(c_A \alpha_A)$ 와  $(c_B \alpha_B)$ 를 두 이미지의 대응되는 화소의 미리 곱한 색깔(pre-multiplied color)이라 하자.  $F_A$ 와  $F_B$ 를 각각 A와 B의 화소에서의  $\alpha_A$ 와  $\alpha_B$ 의 비율로 존재하는 미립자들 중 결과 이미지 O에 살아 남는 미립자의 비율이라 하면, 합성의 결과 생성되는 미리 곱한 색깔은 다음과 같다.

$$\begin{pmatrix} c_O \\ \alpha_O \end{pmatrix} = F_A \begin{pmatrix} c_A \\ \alpha_A \end{pmatrix} + F_B \begin{pmatrix} c_B \\ \alpha_B \end{pmatrix}$$

- (a) 만약  $F_A = \alpha_B$ 이고  $F_B = 1 - \alpha_A$ 일 경우 어떤 방식으로 이미지가 합성이 되는지 설명하라. 이유를 기술할 것.
- (b) 투명한 물체를 렌더링하거나, 안개 등의 기상 효과, 텍스춰의 혼합, 앤티앨리어싱 기법들을 구현하는데 유용하게 사용이 되는 A over B 연산의 경우  $F_A$ 와  $F_B$ 는 각각 얼마인가?
- (c) 어떤 픽셀의 미리 곱한 색깔이  $(0.5, 0.5, 0.5, 0.5)$ 이라면, 이 픽셀은 어떤 RGB 색깔로 어떻게 칠해져 있는 상태인가?
- (d) 약간 투명한 물체들을 카메라를 기준으로 하여 뒤에 있는 것부터 시작하여 앞으로 오면서 차례대로 그려주려 한다. OpenGL의 색깔 혼합 기능을 사용하려 할 때, 원시 인자(source factor)와 목적 인자(destination factor)로 어떤 값을 사용해야 할까? 다음 값들 중 적절한 인자를 선택하라.

```
GL_ZERO, GL_ONE
GL_SRC_ALPHA
GL_DST_ALPHA
GL_ONE_MINUS_SRC_ALPHA
GL_ONE_MINUS_DST_ALPHA
```

5. 다음은 텍스춰 필터링에 관한 문제이다.

- (a) 레벨 2의 mip맵 텍스춰의 한 텍셀이 나타내는 영역의 면적은 레벨 5의 mip맵 텍스춰의 한 텍셀이 나타내는 영역의 면적의 몇 배일까?
- (b) 해상도가  $512 \times 512$ 이고 각 텍셀 당 GL\_UNSIGNED\_BYTE와 GL\_RGBA 형식을 사용하는 텍스춰 이미지에 대하여 모든 가능한 레벨에 대하여 mip맵을 구성할 경우 얼마 만큼의 텍스춰 메모리가 필요할까(계산 과정을 기술할 것)?
- (c) 그림 3(a)에서 a는 픽셀에 대한 원상(pre-image)의 중점을 가리키고, b부터 e까지는 a를 포함하는 주변 텍셀의 중심점을 나타낸다. b, c, d, 그리고 e 지점의 텍셀 색깔이 각각  $(1, 1, 1)$ ,  $(1, 0, 0)$ ,  $(1, 1, 0)$ , 그리고  $(0, 1, 1)$ 이라고 하자. 각각 최근 필터와 선형 필터를 사용할 경우 a 지점에 대하여 계산되는 색깔은 무엇이 될까?, 필요하다면 반드시 계산 과정을 밝혀라. (이 그림에서 숫자는 거리에 대한 비율을 나타냄.)
- (d) 그림 3(b)는 축소 필터는 GL\_LINEAR\_MIPMAP\_LINEAR를, 확대 필터는 GL\_NEAREST를 사용한 경우의 렌더링 결과이다. 과연 이 그림의 상황은 축소 상황이 발생한 것인지, 아니면 확대 상황이 발생한 것인지 구체적으로 기술하라.
- (e) 그림 3(c)는 축소 상황이 발생한 경우이다. 서로 다른 레벨의 mip맵 텍스춰에 대해서 서로 다른 색깔의 텍스춰 이미지를 사용하였는데, 과연 이 그림은 삼선형 필터(trilinear interpolation filter)를 사용한 결과라 할 수 있는가? 예/아니오로 답하고 그렇게 답한 이유를 밝혀라.

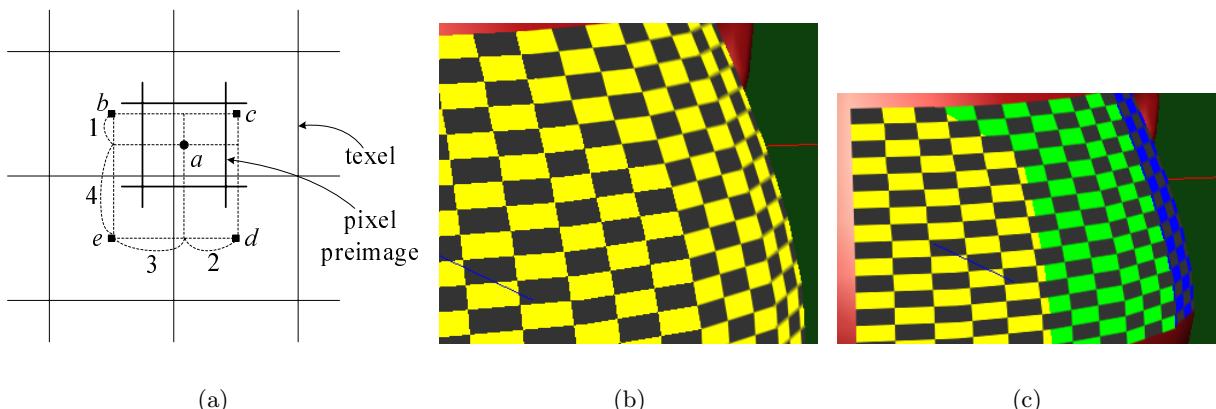


그림 3: 텍스춰 필터링

6. 다음은 Binary Space Partitioning Tree (BSPT)에 관한 문제이다. 그림 4의 함수 `display_bspt_back_to_front()`는 시점의 위치 `viewer[3]`가 주어졌을 때, 뒤에서 앞으로(back-to-front) 가면서 bspt의 다각형들을 그려주는 함수이다.

- (a) (A)에 들어갈 C 코드는 무엇일까?
- (b) (B), (C), (D), (E)에 들어갈 C 코드는 무엇일까?
- (c) 어떤 BSPT가  $n$ 개의 다각형으로 구성이 되어 있을 경우, 이 BSPT의 높이(height)는 최소 얼마에서 최대 얼마까지 가능한가? Big-O 기호를 사용하여 기술하라.
- (d) `los()` 함수는 `start` 지점에 있는 사람이 `end` 지점의 사람을 볼 수 있는지를 결정해주는 함수로서, 볼 수 있으면 1 값을, 그리고 볼 수 없으면 0 값을 리턴 해준다. 여기서 문맥상 함수 `AAA()`는 어떤 경우에 어떤 값을 리턴해주는 함수일까?
- (e) 이 함수에서 (F)와 (G)에 들어갈 C 코드는 무엇일까?
- (f) 이 함수에서 잘못된 부분을 지적하고 고쳐라.

```

typedef struct _bspt { // data structure for bspt
 Polygon *poly;
 struct _bspt *fchild; // for the front side
 struct _bspt *bchild; // for the back side
} BSPT;

void display_bspt_back_to_front(BSPT *bspt, float *viewer) {
 int viewer_side;

 if ((A)) return;
 viewer_side = check_side(viewer, bspt->poly);

 if (viewer_side == BSPT_FRONT) {
 display_bspt_back_to_front((B) , viewer);
 draw_bspt_poly(bspt->poly);
 display_bspt_back_to_front((C) , viewer);
 }
 else {
 display_bspt_back_to_front((D) , viewer);
 draw_bspt_poly(bspt->poly);
 display_bspt_back_to_front((E) , viewer);
 }
}

int los(BSPT *bspt, float *start, float *end) {
 int cs, ce, side0, side1;

 if (!bspt) return 1;
 cs = check_side(start, bspt->poly);
 ce = check_side(end, bspt->poly);

 if ((cs == BSPT_FRONT) && (ce == BSPT_FRONT)) {
 return los(bspt->fchild, start, end);
 }
 else if ((cs == BSPT_BACK) && (ce == BSPT_BACK)) {
 return los(bspt->bchild, start, end);
 }
 else {
 if (AAA(bspt->poly, start, end) return 0;
 side0 = los((F) , start, end);
 side1 = los((G) , start, end);
 return side0 || side1;
 }
}

```

그림 4: BSPT 관련 함수

기초 컴퓨터 그래픽스 기말고사 (43-170)

반: \_\_\_\_\_ 학과: \_\_\_\_\_ 학번: \_\_\_\_\_ 이름: \_\_\_\_\_

1. 다음 문제에 답하라.

- (a) OpenGL 파이프라인에서 `glEnable(GL_CULL_FACE)`; 문장이 설정하는 계산이 실제로 수행이 되는 좌표계는?

(답) \_\_\_\_\_

- (b) OpenGL 파이프라인에서 풍의 조명 모델 계산이 수행되는 좌표계는?

(답) \_\_\_\_\_

- (c) “계층적 모델링 기법을 이용하여 물체를 그릴 때 넓이 우선 탐색을 하는 것이 더 효율적이다”라는 주장이 맞는지 틀린지 말하고, 그 이유를 설명하라.

(답) \_\_\_\_\_

- (d) OpenGL 파이프라인에서의 퍽셀 데이터, 즉 프래그먼트가 포함하는 네 가지 정보는 무엇인가?

(답)

- a. \_\_\_\_\_
- b. \_\_\_\_\_
- c. \_\_\_\_\_
- d. \_\_\_\_\_

- (e) 텍스춰 축소(minification) 상황에서 최근 필터보다 이선형 필터가 좋은 결과를 놓고는 하는데, 축소의 정도가 심해져도 과연 그러할까? 답을 말하고 그 이유를 설명하라.

(답) \_\_\_\_\_

2. 다음 코드에 대하여 프로그래머 입장에서 최적화를 하려고 한다. 할 수 있는 방법 두 가지

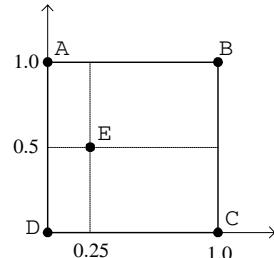


그림 1: 이선형 보간

를 기술하라.

```
void too_bad(double f, double g,
double h) {
 int i;

 for (i = 0; i < 3; i++) {
 A[i] = B[i] + f/g;
 C[i] = D[i] + f*h/g;
 }
}
```

(답)

a. \_\_\_\_\_

b. \_\_\_\_\_

3. 그림 1에서 A, B, C, D 각 점의 색깔이 각각  $(1, 0, 0)$ ,  $(0, 1, 0)$ ,  $(0, 0, 1)$ ,  $(1, 1, 1)$ 이라 하자. 이 데이터를 이용하여 E점에서의 색깔을 이선형 보간을 통하여 구하라. (계산 과정 명기할 것.)

(답)

4. 만약 여러분이 OpenGL을 사용하여 3차원 게임 엔진을 제작한다고 할 때, 이 소프트웨어는 수업 시간에 설명한 바와 같이 Application 레벨, Geometry 레벨, Pixel 레벨 등 세 레벨로 구성할 수 있다. 다음과 같은 연산을 생각하자.

(1) 깊이 버퍼링, (2) 충돌 검사, (3) 텍스춰 매핑, (4) 투영 변환, (5) BSPT를 이용한 뷰 볼륨에 대한 물체 제거, (6) 색깔 블렌딩, (7) 풍의 조명 모델의 적용, (8) 원근 나눗셈

- (a) Application 레벨 계산과 가장 연관이 있는 연산의 번호를 모두 나열하라.

(답) \_\_\_\_\_

- (b) Geometry 레벨 계산과 가장 연관이 있는 연산의 번호를 모두 나열하라.

(답) \_\_\_\_\_

- (c) Pixel 레벨 계산과 가장 연관이 있는 연산의 번호를 모두 나열하라.

(답) \_\_\_\_\_

5. 두 장의 이미지 A와 B의 합성에 대하여 생각하자.  $(c_A \alpha_A)$ 와  $(c_B \alpha_B)$ 를 두 이미지의 대응되는 화소의 미리 곱한 색깔(pre-multiplied color)이라 하자.  $F_A$ 와  $F_B$ 를 각각 A와 B의 화소에서의  $\alpha_A$ 와  $\alpha_B$ 의 비율로 존재하는 미립자들 중 결과 이미지 O에 살아 남는 미립자의 비율이라 하면, 합성의 결과 생성되는 미리 곱한 색깔은 다음과 같다.

$$\begin{pmatrix} c_O \\ \alpha_O \end{pmatrix} = F_A \begin{pmatrix} c_A \\ \alpha_A \end{pmatrix} + F_B \begin{pmatrix} c_B \\ \alpha_B \end{pmatrix}$$

- (a) 만약  $F_A = 1 - \alpha_B$ 이고  $F_B = \alpha_A$ 일 경우 어떠한 방식으로 이미지가 합성이 되는지 설명하라. (이유도 설명할 것.)

(답) \_\_\_\_\_

- (b) A over B 형태의 합성의 경우  $F_A$ 와  $F_B$ 는 각각 얼마인가?

(답) \_\_\_\_\_

- (c) 세 장의 이미지 A, B, C의 세 개의 사각형의 실제 색깔(미리 곱한 색깔이 아님)과 알파값이 각각  $(1.0, 0.5, 0.0, 0.25)$ ,  $(0.25, 0.5, 1.0, 0.5)$ ,  $(0.5, 0.5, 0.5, 1.0)$ 이라 하자. 이때 뒤에서 앞으로(back-to-front) 순서로 over 연산을 사용하여 색깔을 합성할 경우(A가 가장 앞쪽이고, C가 가장 뒤쪽임) 보이는 색깔은 무엇이 될까? 반드시 계산 과정을 밝힐 것.

### - 뒤면 사용 -

(답) \_\_\_\_\_

- (d) 그림 3은 위의 문제에 대하여 앞에서 뒤로 (front-to-back) 순서로 색깔을 합성해 주는 OpenGL 함수이다. 잘못된 부분을 모두 고쳐라.

(답) \_\_\_\_\_

- (e) 그림 3의 코드를 뒤에서 앞으로 (back-to-front) 순서로 합성하는 코드로 바꾸어 위의 세 사각형을 합성하려 할 때, 색깔 버퍼(color buffer)에 RGB 채널 외에 알파 버퍼가 반드시 있어야 할까? 답을 말하고 반드시 그 이유를 설명하라.

(답) \_\_\_\_\_

6. 다음은 아래의 풍의 조명 모델에 관한 문제이다.

$$I_\lambda = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \cdot L_i) + k_{s\lambda} \cdot (N \cdot H_i)^n\}$$

여기서 사용되는 변수  $I_{a\lambda}$ ,  $k_{a\lambda}$ ,  $d_i$ ,  $I_{l_i\lambda}$ ,  $k_{d\lambda}$ ,  $N$ ,  $L_i$ ,  $k_{s\lambda}$ ,  $H_i$ ,  $n$ 에 대하여 답하라.

- (a) 카메라에서 바라보는 방향에 영향을 받는 변수를 모두 나열하라.

(답) \_\_\_\_\_

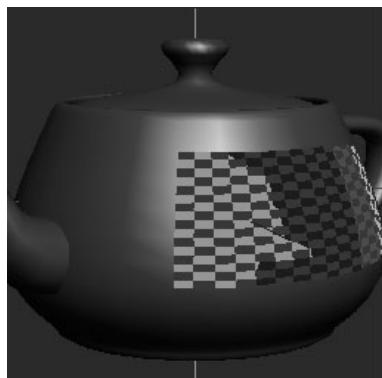


그림 2: 밀립 예

- (b) 광원의 위치의 변화에 영향을 받는 변수를 모두 나열하라.

(답) \_\_\_\_\_

- (c) 물체의 위치와 방향이 바뀌었을 때 영향을 받는 변수를 모두 나열하라.

(답) \_\_\_\_\_

- (d)  $H_i$ 는 어떤 상황에서 특히 효율적으로 사용이 되는가? 두 가지 조건을 명시하라.

(답) \_\_\_\_\_

a. \_\_\_\_\_

b. \_\_\_\_\_

7. 다음은 밀매핑에 관한 문제이다.

- (a) 그림 2는 축소 상황에서 `GL_NEAREST_MIPMAP_NEAREST` 필터를 사용하여 텍스춰 매핑을 한 이미지이다. 이 그림에서는 네 개의 서로 다른 레벨의 텍스춰를 사용하고 있는데, 어느 부분이 가장 높은 레벨의 텍스춰를 사용하는지를 말하고, 그 이유를 기술하라.

(답) \_\_\_\_\_

- (b) 만약 `GL_NEAREST_MIPMAP_NEAREST` 대신에 `GL_LINEAR_MIPMAP_LINEAR` 필터를 사용할 경우 이 이미지에 어떤 변화가 올까?

(답) \_\_\_\_\_

8. 다음은 Binary Space Partitioning Tree (BSPT)에 관한 문제이다. 그림 4의 `display_bspt_front_to_back()`은 시점의 위치 `viewer[3]`가 주어졌을 때, 앞에서 뒤 (front-to-back) 순서로 가면서 BSPT `bspt`의 다각형을 렌더링해주는 함수이다.

- (a) 문맥상 `int check_side (float *pt, Polygon *poly)` 함수는 어떤 일을 해주는 함수인가?

(답) \_\_\_\_\_

- (b) (A)에 들어갈 코드는 무엇일까?

(답) \_\_\_\_\_

- (c) (B), (C), (D), (E)에 들어갈 코드는 무엇일까?

(답) \_\_\_\_\_

(B) \_\_\_\_\_

(C) \_\_\_\_\_

(D) \_\_\_\_\_

(E) \_\_\_\_\_

- (d) 어떤 BSPT가  $n$ 개의 다각형으로 구성이 되어 있을 경우, 이 BSPT의 높이(height)는 최소 얼마에서 최대 얼마까지 가능한가? Big-O 기호를 사용하여 기술하라.

```

void front_to_back(void) {
 // OVER: Front-to-Back
 glBlendFunc(GL_ONE_MINUS_SRC_ALPHA, GL_ONE);
 glColor4f(1.0, 0.5, 0.0, 0.25); draw_rectangle0(); // 삼각형 A
 glColor4f(0.25, 0.5, 1.0, 0.5); draw_rectangle1(); // 삼각형 B
 glColor4f(0.5, 0.5, 0.5, 1.0); draw_rectangle2(); // 삼각형 C
}

```

그림 3: 색깔 합성 관련 함수

```

typedef struct { // data structure for polygon
 int nv; // number of vertices
 float *vertex; // pointer to vertex data
 float *normal; // pointer to normal data
 float plane[4]; // plane equation
} Polygon;

typedef struct _bspt { // data structure for bspt
 Polygon *poly;
 struct _bspt *fchild; // for the front side
 struct _bspt *bchild; // for the back side
} BSPT;

int check_side(float *pt, Polygon *poly) {
 :
}

void display_bspt_front_to_back(BSPT *bspt, float *viewer) {
 int viewer_side;

 if ((A)) return;
 viewer_side = check_side(viewer, bspt->poly);

 if (viewer_side == BSPT_FRONT) {
 display_bspt_front_to_back((B) , viewer);
 draw_bspt_poly(bspt->poly);
 display_bspt_front_to_back((C) , viewer);
 }
 else {
 display_bspt_front_to_back((D) , viewer);
 draw_bspt_poly(bspt->poly);
 display_bspt_front_to_back((E) , viewer);
 }
}

```

그림 4: BSPT 관련 함수

## 기초 컴퓨터 그래픽스 Quiz 2 (43-170)

반: \_\_\_\_\_ 학과: \_\_\_\_\_ 학번: \_\_\_\_\_ 이름: \_\_\_\_\_ 암호: \_\_\_\_\_

1. 그림 1의 프로그램을 보고 답하라.

- (a) 광선 추적법을 사용하면 한 물체의 표면에 주변의 물체가 자연스럽게 반사되게 할 수가 있다. 이 프로그램에서 물체에 반사되는 색깔을 계산하는데 있어 가장 밀접하게 연관이 있는 두 개의 문장은 무엇일까?

(답) ( ), ( )

- (b) 광선 추적 중 지역 조명 모델인 풍의 조명 모델을 계산하는 것과 가장 밀접하게 연관이 있는 문장은?

(답) ( )

- (c) 문장 (12)는 어떠한 계산을 하기 위한 것인가?

(답)

- (d) 어떠한 경우에 문장 (14)가 수행이 될까? 즉 어떤 경우에 ray\_hit() 함수가 0값을 return할까?

(답)

- (e) 광선 추적 계산을 하는데 있어 광선과 물체와의 교차점에서의 법선 벡터의 방향을 직접적으로 필요로 하는 문장을 모두 나열하라.

(답) ( )

2. 아래의 식을 보고 답하라.

$$I_{\lambda} = I_{a\lambda} \cdot k_{a\lambda} + \sum_{i=0}^{m-1} f_{att}(d_i) \cdot I_{l_i\lambda} \cdot \{k_{d\lambda} \cdot (N \cdot L_i) + k_{s\lambda} \cdot (N \cdot H_i)^n\}$$

- (a) 이 식에서 일반적으로 물체의 기반 색깔이 지정이 되는 변수는?

(답)

- (b) 생성되는 하이라이트의 크기를 조절하는데 사용되는 변수는?

(답)

- (c) 헤프웨이 벡터  $H_i$ 는 어떤 방향을 나타내는 벡터인가?.

(답)

- (d) 이 식에서  $f_{att}(d_i)$ 는 어떤 조명 효과를 내기 위한 것일까?

(답)

3. 그림 2는 수업 시간에 설명한 바와 같이 MC의 자동차를 WC로 배치하는 모습을 보여주고 있다. 여기서는 오른손 좌표계를 사용하고 있으므로  $y$ 축은 문제지의 위쪽을 향하고 있다. 변환 행렬  $M$ 을  $M = T(5, 0, 5) \cdot R$ 과 같은 형태로 표현하고자 할 경우, 이때의 4행 4열 행렬  $R$ 을 구하라. 여기서  $\alpha^2 + \beta^2 = 1$ 이고,  $R$ 을 계산할 때, 사인, 코사인등과 같은 함수를 사용하지 말 것.

(답)

4. 다음 물음에 답하라.

(a) 지역 조명 모델과 전역 조명 모델의 근본적인 차이는 무엇인가?

(답)

(b) OpenGL 시스템에서 각 꼭지점에 대한 라이팅 계산은 어떤 좌표계에서 수행이 되는가?

(답)

(c) OpenGL 시스템에서 레스터화(rasterization) 과정은 어떤 좌표계에서 수행이 되는가?

(답)

(d) OpenGL의 glTexEnv\*(\*) 함수는 텍스춰 매핑의 계산에 있어 구체적으로 어떤 계산을 조절하기 위한 것일까?

(답)

(e) 해상도가  $256 \times 256$ 이고 각 텍셀 당 3바이트를 사용하는 텍스춰 이미지에 대하여 모든 가능한 레벨에 대하여 밀맵을 구성할 경우 얼마 만큼의 텍스춰 메모리가 필요할까? (이유를 설명할 것.)

(답)

(f) 레스터화 과정의 결과 생성되는 프래그먼트(fragment)는 해당 픽셀의 위치 대han  $(x, y)$  좌표,  $(r, g, b, a)$  색깔, 텍스춰 좌표  $(s, t, r, q)$ , 그리고 어떤 정보가 포함이 되는가?

(답)

(g) 계층적인 정보를 표현하는데 있어 자연스럽게 사용이 될 수 있는 자료 구조 한 개를 기술하라.

(답)

(h) 해프웨이 벡터는 어떤 상황에서 효율적으로 사용이 될 수 있는가? 구체적으로 두 가지 조건을 나열하라.

(답)

(i) WC와 카메라의 프레임을 일치시킨 후, 카메라에 대하여 변환  $M_1, M_2, M_3$ 를 순서대로 가하여 카메라를 배치하는 상황을 OpenGL 함수 glMultMatrixf(M);(여기서  $M$ 은 해당 행렬)를 사용하여 구현하려 한다. 어떠한 방식으로 구현을 할 수 있는가?

(답)

5. 다음 물음에 답하라.

- (a) 두 개의 픽셀 A와 B의 합성에 관한 문제이다. 지금 A를 B 위에 놓아 **over** 연산을 취하려 한다. 두 픽셀의 색깔을 각각  $(C_A, \alpha_A)$ 와  $(C_B, \alpha_B)$ 라 할 경우 A **over** B의 결과 생성되는 색깔  $(C_O, \alpha_O)$ 를 기술하라. (주의: 여기서의 RGB 색깔 C는 미리 곱한 색깔이 아니고 원래의 색깔임.)

(답)

- (b) 그림 3을 보고 답하라. 지금 불투명도가 각각 0.3, 0.5, 그리고 1.0인 유리를 앞에서 바라보고 있다. 이때 결과적으로 보이는 색깔 C는  $C = w_1 \cdot C_0 + w_2 \cdot C_1 + w_3 \cdot C_2$ 와 같이 표현할 수 있는데  $w_1, w_2$ , 그리고  $w_3$  값이 무엇인지 구하라. (반드시 유도 과정이 있어야함.)

(답)

6. 아래의 코드를 보고 답하라. 지금 glLightfv(GL\_LIGHT0, GL\_POSITION, position)과 같은 함수의 호출을 통하여 점광원의 위치를 설정하려 한다.

(1); glMatrixMode(GL\_MODELVIEW); glLoadIdentity(); (2); gluLookAt(a, b, c, d, e, f, g, h, i);  
 (3); glTranslatef(j, k, l); glRotatef(m, n, o, p); (4); draw\_object\_in\_MC(); (5);

- (a) 만약 세상 좌표계(WC)에 고정이 되어 있는 광원이라면 어느 지점에서 위 함수를 호출하는 것이 가장 자연스러운가?

(답) ( )

- (b) 만약 기하 물체에 고정이 된 광원(예를 들어 자동차의 헤드라이트)이라면 어느 지점에서 위 함수를 호출하는 것이 가장 자연스러운가?

(답) ( )

- (c) 광부들이 쓰는 것과 같은 헤드라이트가 부착된 헬멧을 쓰고 컴컴한 동굴을 돌아 다니는 것과 같은 상황에 적합한 광원은 어느 지점에서 설정하는 것이 가장 자연스러운가?

(답) ( )

```

void TraceRay(point start, point dir, int depth, colors *color) { —(1)
 point hit_point, refl_dir, trans_dir; —(2)
 colors local_color, refl_color, trans_color; —(3)
 object hit_object; —(4)

 if (depth > MAXDEPTH) *color = BLACK; —(5)
 else if (ray_hit(start, dir, &hit_object, &hit_point)) { —(6)
 shade(hit_object, hit_point, &local_color); —(7)
 calculate_reflection(hit_object, hit_point, &refl_dir); —(8)
 calculate_transmission(hit_object, hit_point, &trans_dir); —(9)
 TraceRay(hit_point, refl_dir, depth+1, &refl_color); —(10)
 TraceRay(hit_point, trans_dir, depth+1, &trans_color); —(11)
 Combine(hit_object, local_color, refl_color, trans_color, color); —(12)
 } —(13)
 else *color = BACKGROUND_COLOR; —(14)
}
} —(15)
} —(16)

```

그림 1: 1번 문제

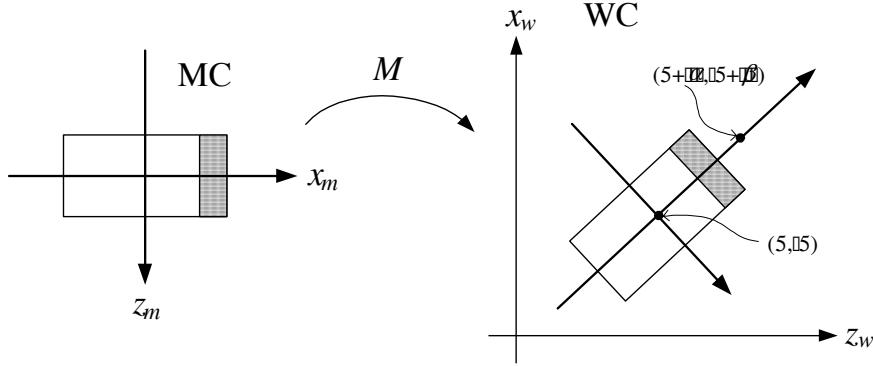
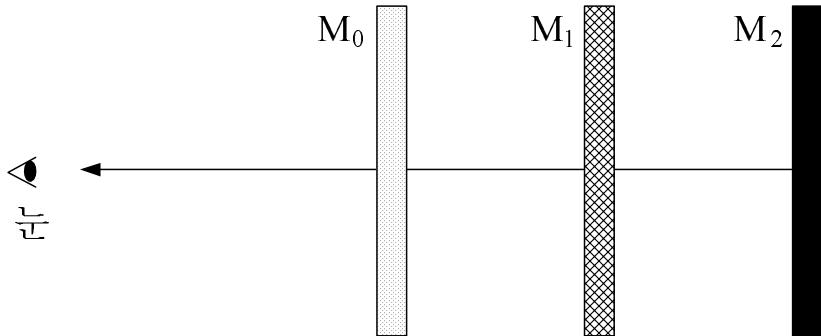


그림 2: 3번 문제



$$\alpha_0 = 0.3 \quad \alpha_1 = 0.5 \quad \alpha_2 = 1.0$$

$$C_0 \quad C_1 \quad C_2$$

그림 3: 5번 문제