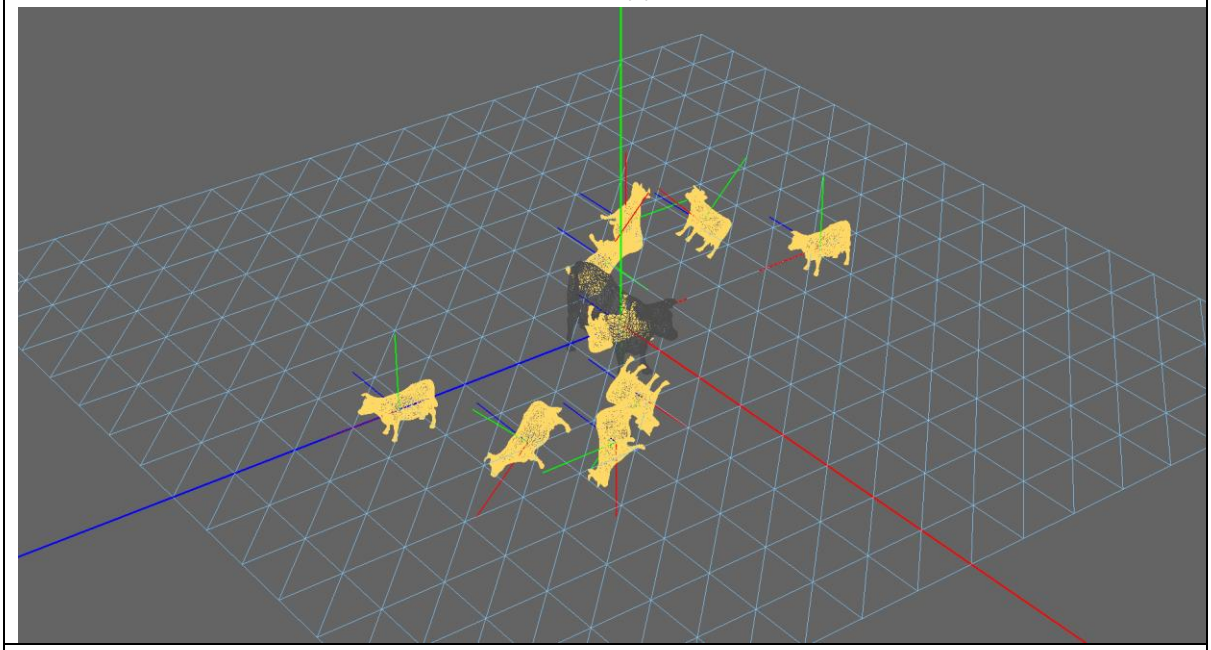
기초 컴퓨터 그래픽스 HW4

20181617 김채연

1. 다음은 간단한 모델링 변환에 관한 문제이다. 주어진 질문에 답하시오.

1) 아래의 코드는 그림(a)에 있는 sin함수를 따라 도시한 소 모델을 적절한 모델링 변환을 통하여 세상에 배치해주는 프로그램의 일부이다. 이 코드가 올바르게 작동하기 위하여 (A)~(I)까지 들어가야 할 값을 정확히 기술하여라. (원점에 배치되어 있는 회색의 소는 주변의 소보다 2배 크다. 노랑색의 소는 Z축기준 오른쪽에서 왼쪽으로 이동한다.)

<그림 (a)>



<코드>

```
ModelViewProjectionMatrix = glm::translate(ViewMatrix, glm::vec3(0.0f, 0.0f, 0.0f));
ModelViewProjectionMatrix = glm::scale(ModelViewProjectionMatrix, glm::vec3(2.0f, 2.0f, 2.0f));
ModelViewProjectionMatrix = ProjectionMatrix * ModelViewProjectionMatrix;
glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE,
&ModelViewProjectionMatrix[0][0]);
glLineWidth(2.0f);
draw_axes();
glLineWidth(1.0f);
draw_object(OBJECT_COW, 0.3f, 0.3f, 0.3f);                    => 회색소

for (int i = 0; i <= 360; i += (A)) {
        float angle = (float)i;
```

```
        ModelViewProjectionMatrix = glm::translate(ViewMatrix, glm::vec3(2* (B), (C), (D)));
        ModelViewProjectionMatrix = glm::rotate(ModelViewProjectionMatrix, (E),
glm::vec3(0.0f, 1.0f, 0.0f));
        ModelViewProjectionMatrix = glm::rotate(ModelViewProjectionMatrix, (F), glm::vec3((G),
(H),(I)));
        ModelViewProjectionMatrix = ProjectionMatrix * ModelViewProjectionMatrix;
        glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE,
&ModelViewProjectionMatrix[0][0]);
        glLineWidth(2.0f);
        draw_axes();
        glLineWidth(1.0f);
        draw_object(OBJECT_COW, 255/255.0f, 217/255.0f, 102/255.0f);
}                                                                    =>노랑색소
```

답:

(A) : 45

(B) : sinf((angle-180)*TO_RADIAN)

(C) : 0.0f

(D) : (angle-180)*TO_RADIAN

(E) : -90 * TO_RADIAN

(F) : angle*TO_RADIAN

(G) : 0.0f

(H) : 0.0f

(I) : 1.0f

해설:

(A) : 노랑색 소가 총 9개 있으므로 360/9 = 45, 답은 45이다.

(B) : X-Z평면상에 Z축을 기준으로 sin함수를 그리고 있다. 그리고 angle의 범위는 0~360인데 그려진 sin함수는 -180~180의 범위를 가지고 있고 angle은 RADIAN으로 바꿔주어야 한다. 그러므로 X좌표는 sinf((angle-180)*TO_RADIAN) 이다.

(C) : Y축의 변화가 없으므로 Y좌표는 0.0f 이다.

(D) : X-Z 평면상에 Z축을 기준으로 sin함수를 그리고 있다. sin함수의 범위가 -180~180이므로 0~360의 범위인 angle에 180을 빼준 뒤 RADIAN으로 바꿔준다. 그러므로 Z좌표는 (angle-180)*TO_RADIAN 이다.

(E) : 회색소가 X축을 바라보고 있는데 노랑색 소는 Z축을 바라보고 있기 때문에 Y축을 회전축으로 -90도를 회전시켜주어야 한다. 그러므로 답은 -90 * TO_RADIAN 이다.

(F) : 노랑색 소는 앞에서 -90도 회전된 뒤 Z축을 기준으로 9개의 소가 점점 360도 회전하므로 angle만큼 회전한다고 볼 수 있다. 그러므로 답은 angle*TO_RADIAN 이다.

(G) : X축을 기준으로 회전하지 않으므로 0.0f 이다.

(H) : Y축을 기준으로 회전하지 않으므로 0.0f 이다.

(I) : Z축을 기준으로 회전하므로 1.0f 이다.

```
for (int i = 0; i <= 360; i += 45) {
		float angle = (float)i;
		ModelViewProjectionMatrix = glm::translate(ViewMatrix, glm::vec3(2* sinf((angle - 180) * TO_RADIAN), 0.0f, (angle - 180) * TO_RADIAN));
		ModelViewProjectionMatrix = glm::rotate(ModelViewProjectionMatrix, (-90) * TO_RADIAN, glm::vec3(0.0f, 1.0f, 0.0f));
		ModelViewProjectionMatrix = glm::rotate(ModelViewProjectionMatrix, angle*TO_RADIAN, glm::vec3(0.0f, 0.0f, 1.0f));
		ModelViewProjectionMatrix = ProjectionMatrix * ModelViewProjectionMatrix;
		glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE, &ModelViewProjectionMatrix[0][0]);
		glLineWidth(2.0f);
		draw_axes();
		glLineWidth(1.0f);
		draw_object(OBJECT_COW, 255/255.0f, 217/255.0f, 102/255.0f);
}
```

2) 문제 1)의프로그램이 제대로 완성되었을 때 X-Z평면에서 Z축을 기준으로 sin함수에 위치해 있는 물체를 XY평면에서 X축 기준으로 바꾸려면 위의 코드에서 무엇을 바꾸어야 하는가?

답 : ModelViewProjectionMatrix = glm::translate(ViewMatrix, glm::vec3(2* sinf((angle - 180) * TO_RADIAN), 0.0f, (angle - 180) * TO_RADIAN));
이 코드를

ModelViewProjectionMatrix = glm::translate(ViewMatrix, glm::vec3((angle - 180) * TO_RADIAN, 2* sinf((angle - 180) * TO_RADIAN),0.0f);

로 바꾼다.

해설 : XZ평면의 Z기준을 XY평면에서 X축 기준으로 바꾸려면 기존 코드의 X좌표를 Y좌표로, Y좌표를 Z좌표로, Z좌표를 X좌표루 바꾸면 된다.

3) 아래의 코드는 소 모델을 적절한 모델링 변환을 통하여 세상에 배치해주는 프로그램의 코드이다. 프로그램이 작동할 때 나타나는 실행 결과에 대한 설명으로 보기에서 옳은 것을 고르시오.

<코드>

```
ModelViewProjectionMatrix = glm::translate(ViewMatrix, glm::vec3(0.0f, 0.0f, 0.0f));
ModelViewProjectionMatrix = glm::scale(ModelViewProjectionMatrix, glm::vec3(2.0f, 2.0f, 2.0f));
ModelViewProjectionMatrix = ProjectionMatrix * ModelViewProjectionMatrix;
glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE,
&ModelViewProjectionMatrix[0][0]);
glLineWidth(2.0f);
draw_axes();
glLineWidth(1.0f);
draw_object(OBJECT_COW, 0.3f, 0.3f, 0.3f);                                    => 회색소

for (int i = -5; i <= 5; i += 1) {
        float z = (float)i;
        if(i>=0)
                ModelViewProjectionMatrix = glm::translate(ViewMatrix, glm::vec3(0.0f, -(z-
2.5)*(z-2.5)*3/2.5/2.5+3, z));
        else if (i < 0)
```

```
                ModelViewProjectionMatrix = glm::translate(ViewMatrix, glm::vec3(0.0f, -(z +
2.5) * (z + 2.5)*3/2.5/2.5 + 3, z));
        ModelViewProjectionMatrix = glm::rotate(ModelViewProjectionMatrix, (z+5)*36 *
TO_RADIAN, glm::vec3(-1.0f, 1.0f, 0.0f));
        ModelViewProjectionMatrix = glm::scale(ModelViewProjectionMatrix, glm::vec3(1-(z +
5) / 20, 1-(z + 5) / 20, 1-(z + 5) / 20));
        ModelViewProjectionMatrix = ProjectionMatrix * ModelViewProjectionMatrix;
        glUniformMatrix4fv(loc_ModelViewProjectionMatrix, 1, GL_FALSE,
&ModelViewProjectionMatrix[0][0]);
        glLineWidth(2.0f);
        draw_axes();
        glLineWidth(1.0f);
        draw_object(OBJECT_COW, 255/255.0f, 102/255.0f, 153/255.0f);
}                                                                    =>핑크색 소
```

<보기>

ㄱ. 물체(핑크색 소) 가 Z축의 +로 갈수록 크다.
ㄴ. 물체(핑크색 소) 가 Z축의 -로 갈수록 크다.
ㄷ. 물체(핑크색 소) 의 크기는 변화가 없다.
ㄹ. 물체(핑크색 소) 는 원점에서 Y축의 +를 바라보고 있다.
ㅁ. 물체(핑크색 소) 는 원점에서 Y축의 -를 바라보고 있다.
ㅂ. 물체(핑크색 소) 는 원점에서 X축의 +를 바라보고 있다.
ㅅ. 물체(핑크색 소) 는 X-Y평면상에 있다.
ㅇ. 물체(핑크색 소) 는 Y-Z평면상에 있다.
ㅈ. 물체(핑크색 소) 는 X-Z평면상에 있다.
ㅊ. 물체(핑크색 소) 는 대칭형태로 위치한다.

답 : ㄴ, ㅁ, ㅇ, ㅊ,

해설 :

주어진 코드의 translate를 보면 x 좌표는 0, z 좌표가 i이고 y의 좌표는 z좌표의 부호에 따라 y=(3/2.5/2.5)*(z+-2.5)^2+3이다. 이를 통해 물체는 Y-Z평면상에 있다는 것과 Y축을 기준으로 대칭형태인 것을 확인할 수 있다. 그리고 scale 을 보면 x, y, z 모두 1-(z + 5) / 20 이므로 z가 증가할수록 점점 작아진다. 그리고 물체가 원점일 때는 i가 0일 때이므로 rotate를 보면 (z+5)*36*TO_RADIAN인 180도 만큼 (-1.0f, 1.0f, 0.0f) 회전하므로 처음방향인 x축 +방향에서 회전하여 y축의 -방향을 바라보고 있는 것을 알 수 있다.