단국대학교 응용컴퓨터공학 컴퓨터그래픽스 기말고사 (2018년 가을학기) 2018년 12월 13일 목요일

학과 _____ 학번 _____ 이름 _____

기말고사

담당교수: 박경신

12/20 23:59까지 online.dankook.ac.kr 이러닝으로 학번_이름_final.zip 으로 묶어서 이러닝에 제출한다.

주의: 단순 번역이나 있는 그대로 복사/붙이기는 금지함. 반드시 본인이 참고한 문서의 인용을 넣어줌.

각 문제당 1장 이상씩 서술한다. 장수 제한 없음.

1. 다음 문제에 간단히 답하시오. (10점)

1) **버텍스 쉐이더(vertex shader) 코드에서 in, out, uniform 변수란 무엇인가? 프레그먼크 쉐이더 (fragment shader) 코드에서 in, out, uniform 변수란 무엇인가?**
버텍스 쉐이더 또는 프레그먼트 쉐이더에서 uniform 변수는 전역 변수이며, primitive 마다 바뀔 수 있고, OpenGL 프로그램에서 쉐이더로 값을 변경하는데 사용함. Uniform 변수는 vertex shader와 fragment shader 모두에서 사용가능. 쉐어더 코드 내부에서는 상수임.
버텍스 쉐이더에서 in 변수란 vertex 마다 들어오는 데이터, out 변수란 vertex shader에서 계산된 후 fragment shader로 넘겨지는 값임
프레그먼트 쉐이더에서 in 변수란 vertex shader에서 넘겨준 데이터, out 변수란 fragment shader에서 계산된 최종 결과 값임

2) **직교 투영 (Orthographic Projection), 축측 투영 (Axonometric Projection), 등축투영 (Isometric Projection), 경사 투영 (Oblique Projection), 원근 투영 (Perspective Projection)의 특징을 자세히 서술하라**
직교 투영은 투영선(projector)이 투영면(projection plane)에 수직이다.
축측 투영은 투영선(projector)은 투영면(projection plane)에 수직이지만, 투영면은 객체에 대해 어떠한 방향에서도 존재할 수 있다.
등축 투영은 축측 투영의 하나인데 그 중에서 투영면이 사각형 객체의 모서리에서 만나는 세 개의 주면에 대해서 대칭으로 놓여지는 것이다.
경사 투영은 투영선(projector)은 투영면(projection plane)과 임의의 각을 가질 수 있으나 투영면에 평행한 면내의 각은 보존된다.
원근(투시) 투영은 객체가 관측자로부터 멀리 떨어질수록 크기가 축소된다. 이러한 크기변화는 자연스러운 모습의 관측을 제공한다.

3) **Flat Shading, Gouraud Shading, Phong Shading을 자세히 비교 서술하라.**
Flat shading은 일명 Constant shading이라고도 하며, 폴리곤의 한 면을 동일한 색으로 채우는 방식이다.
Gouraud shading은 일명 smooth shading라고도 불리며, 폴리곤의 각 정점의 색을 linear interpolation하여 내부를 부드럽게 보간하는 방식이다. (**Per-vertex shading**)
Phong shading은 (Gouraud shading이 정점의 색을 이용하여 내부를 보간하는 반면) 폴리곤의 각 정점의 normal을 가지고 폴리곤 내부의 각 픽셀마다 normal이 보간되어 specular reflection이 더욱 정확하게 나타나게 하는 방식이다. (**Per-pixel shading**)

4) **Texture Mapping의 축소, 확대, 밉맵필터를 자세히 비교 서술하라.**
   1. 확대 필터 (MAG_FILTER): 텍스쳐 이미지를 확대하여 보다 넓은 면적의 다각형 영역에 매핑하고자 할 때 사용되는 필터
   2. 축소 필터 (MIN_FILTER): 텍스쳐 이미지를 축소하여 보다 작은 면적의 다각형 영역에 매핑하고자 할 때 사용되는 필터
   3. 밉맵(MIPMAP_FILTER): 텍스쳐 이미지에서 일련의 축소된 크기의 밉맵 택스쳐를 생성하여, 작은 객체에 텍스쳐 맵핑을 할 시 보간 문제를 줄여주는 필터

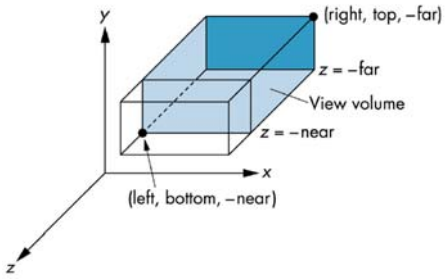
학과 _____ 학번 _____ 이름 _____

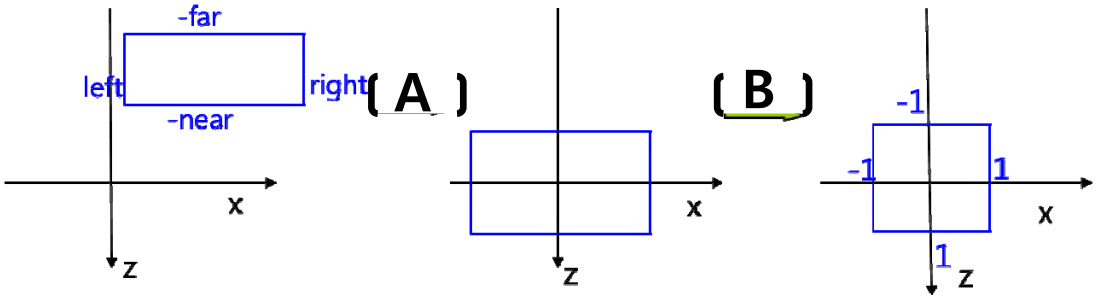
2. 다음 Projection 행렬 질문에 답하라. (25점)

1) 원근 정규화에 대해 설명하라. (5점)

원근 정규화는 원근 투영을 직교투영으로 바꾸는 작업이다.
즉, x=±z, y=±z, z=near/far =>x=±1, y=±1, z=±1로 정규화 (normalization).
이 과정에서 원근감이 생성된다.

2) glm::ortho(0, 10, 0, 10, 0, 10) 함수가 직육면체 관측공간에서 정규화된 관측 공간으로 변환시키는 직교 투영 행렬 (orthographic projection matrix)을 생성해 내는 과정을 보여주고 있다. 각 단계별로 필요한 (A)와 (B) 그리고 최종 행렬(4x4)을 계산하라. (10점)





(A) 행렬: 지정된 관측공간의 중심을 정규관측공간의 중심으로 이동시키는 행렬 T을 적용

(B) 행렬: *x = ±1, y=±1, z=±1* 로 바꿔주기 위하여 관측공간의 변의 길이가 2가 되도록 크기 변환 행렬 S 를 적용

$$A=\begin{pmatrix}1 & 0 & 0 & -5\\ 0 & 1 & 0 & -5\\ 0 & 0 & 1 & 5\\ 0 & 0 & 0 & 1\end{pmatrix}$$

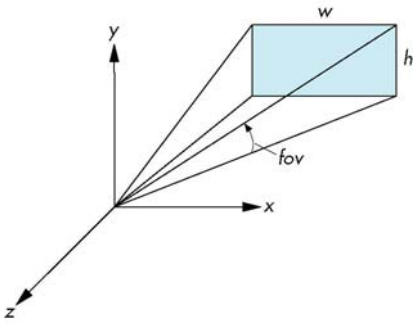
$$B=\begin{pmatrix}\frac{1}{5} & 0 & 0 & 0\\ 0 & \frac{1}{5} & 0 & 0\\ 0 & 0 & -\frac{1}{5} & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

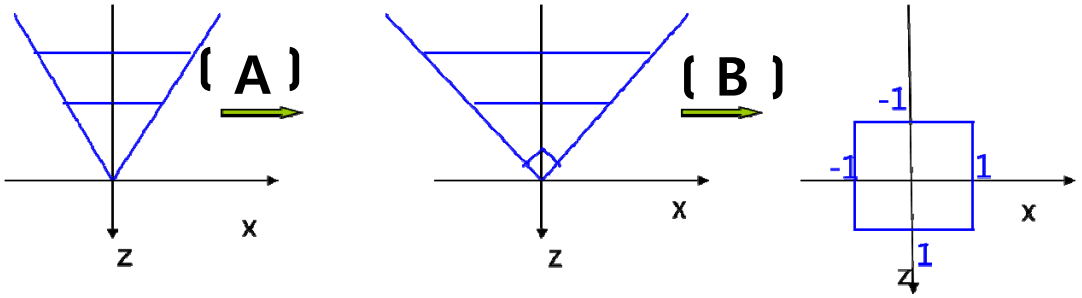
$$ortho=B*A=\begin{pmatrix}\frac{1}{5} & 0 & 0 & -1\\ 0 & \frac{1}{5} & 0 & 1\\ 0 & 0 & -\frac{1}{5} & -1\\ 0 & 0 & 0 & 1\end{pmatrix}$$

학과 _____ 학번 _____ 이름 _____

3) glm::perspective(45, 1, 1, 100) 함수가 절두체에서 정규화된 관측 공간으로 변환시키는 원근 투영 행렬 (perspective projection matrix)을 생성하는 과정을 보여주고 있다. 각 단계별로 필요한 (A)와 (B)의 4x4 행렬을 유도하라. (10점)

glm::perspective(45, 1, 1, 100) = glm::frustum(-1/tan(22.5), 1/tan(22.5), -1/tan(22.5), 1/tan(22.5), 1, 100)





(A) 행렬: 지정된 관측공간을 *x=±z , y=±z, z = near/far* 로 크기변환 행렬 S을 적용

(B) 행렬: *x=±z , y=±z, z = near/far* 관측공간을 x=±1, y=±1, z=±1로 바꿔주는 원근정규화(perspective normalization) 행렬 N을 적용

$$A=\begin{pmatrix} -\frac{2}{2\tan(\frac{45}{2})} & 0 & 0 & 0 \\ 0 & -\frac{2}{2\tan(\frac{45}{2})} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -2.414214 & 0 & 0 & 0 \\ 0 & -2.414214 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$B=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{101}{99} & -\frac{200}{99} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$perspective=B*A=\begin{pmatrix} 2.414 & 0 & 0 & 0 \\ 0 & 2.414 & 0 & 0 \\ 0 & 0 & -\frac{101}{99} & -\frac{200}{99} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

학과 _____ 학번 _____ 이름 _____

3. 다음 View 행렬 질문에 답하라. (20점)

1) 다음 코드에서 밑줄 친 변환함수인 glm::translate, glm::rotate 함수를 제거하고, 화면에 동일하게 View가 나타나도록 glm::lookAt 사용하여 View 행렬 코드를 작성하라. (5점)

```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    spMain.useProgram();
    View = glm::mat4(1.0f); // glm::lookAt(                                    );
    spMain.setUniform("gView", View);
    glm::mat4 T = glm::translate(glm::mat4(1.0f), glm::vec3(0, 0, -10));
    glm::mat4 R = glm::rotate(glm::mat4(1.0f), -M_PI/2.0, glm::vec3(0, 1, 0));
    World = T * R;
    spMain.setUniform("gModel", World);
    drawTeapot();
    glutSwapBuffers();
}
```

물체에 glm::translate(0, 0, -10)과 glm::rotate(-90, glm::vec3(0,1,0))를 적용시킨 것은 카메라가 x+ 에서 바라 본 화면과 동일함. 즉, glm::lookAt함수는 eye(10, 0, 0), at(0, 0, 0), up(0, 1, 0)를 사용하면 된다.

glm::lookAt(glm:::vec3(10, 0, 0), glm::eye(0, 0, 0), glm::vec3(0, 1, 0)); // lookAt

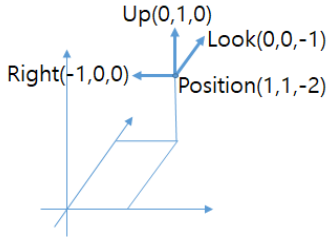
2) 위와는 반대로, 다음 코드에서 밑줄 친 glm::lookAt 함수를 제거하고, 화면에 동일하게 View가 나타나도록 변환함수인 glm::translate, glm::rotate 사용하여 World 행렬 코드를 작성하라. (5점)

```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    spMain.useProgram();
    View = glm::lookAt(glm:::vec3(0, 0, 15), glm::eye(0, 0, 0), glm::vec3(0, 1, 0));
    spMain.setUniform("gView", View);
    glm::mat4 T = glm::mat4(1.0f);
    glm::mat4 R = glm::mat4(1.0f);
    World = T * R;
    spMain.setUniform("gModel", World);
    drawTeapot();
    glutSwapBuffers();
}
```

glm::lookAt함수는 eye(0, 0, 15), at(0, 0, 0), up(0, 1, 0)는 카메라가 z+ 15에서 물체를 바라보는 것으로, glm::translate(0, 0, -15)))를 사용하면 동일한 시점이 된다.

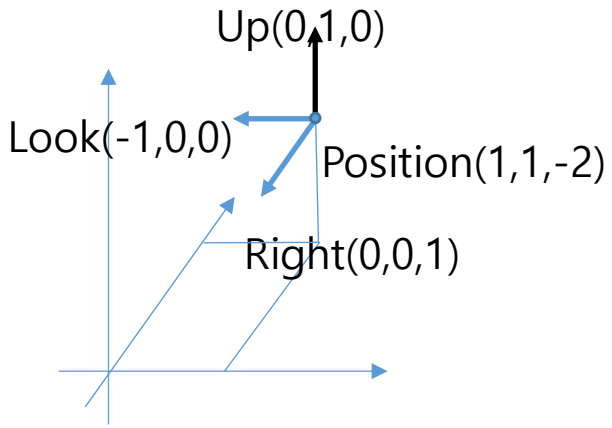
glm::translate(glm::mat4(1.0f), glm:::vec3(0, 0, -15));

3) 아래의 카메라 위치와 방향에서 90도 YAW 회전한 후에 다시 90도 Pitch 회전한 View 행렬을 계산하라. 최종 회전 후의 Right, Up, Look 벡터의 방향을 그림으로 도식화하고 View 행렬을 유도하라. (10점)

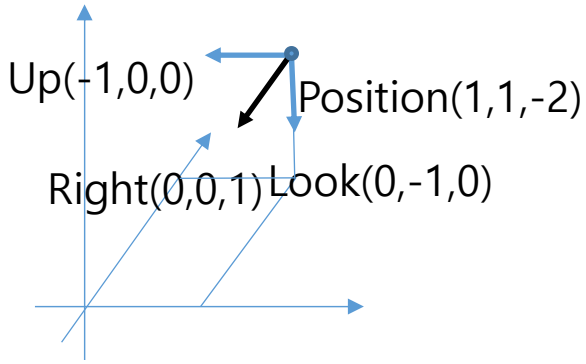


학과 _____ 학번 _____ 이름 _____

Yaw90 회전 후



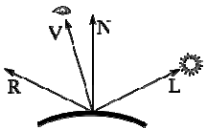
Yaw90 후 Pitch90 회전 후



$$\text{View} = \begin{pmatrix} rx & ry & rz & -p.r \\ ux & uy & uz & -p.u \\ lx & ly & lz & -p.l \\ 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 1 & 2 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4. 다음은 조명 (lighting) 공식에 관한 문제이다. 아래의 질문에 답하시오. (20점)

$$I = K_a I_a + \sum_{i=0}^{m-1} f_{att}(d)\left\{K_d I_d (N \bullet L) + K_s I_s (R \bullet V)^n\right\} + E$$



1) 조명공식에서 램버트 법칙 (Lambertian Law)을 자세히 설명하라. 이 부분에 대한 쉐이더 코드를 적어라. (5점)

램버트 코사인 법칙은 난반사 (diffuse reflection)에서 면의 밝기가 입사각 (즉, 광원 벡터와 법선 벡터의 사이각)의 코사인에 정비례 함을 말하는 것이다. 즉, 면이 서 있는 방향에 따라 차등적 밝기를 제공하여 입체감을 부여할 수 있다.
Fragment shader 코드에서 cosTheta는 법선벡터 (N)과 광원벡터 (L)간의 내적은 두 벡터간의 입사각의 코사인과 비례하며, diffuse reflection과의 곱에서 사용된다.
float cosTheta = clamp(dot(N, L), 0, 1);

2) 블린 (Blinn) 직접 조명 모델을 설명하라. 블린 모델의 경우 위의 공식에서 어떤 부분을 바꿔서 사용하는지, 공식을 적고 자세히 설명하라. (5점)

블린 직접 조명 모델은 Reflection 벡터 대신 Halfway 벡터를 사용하여 정반사 경면광을 계산한다. $(R \bullet V)^n$ 대신 $(H \bullet N)^{n'}$를 사용함.

3) (정점이 4개인) Quad와 (정점이 10000개인) Grid를 각각 Gouraud Shading과 Phong Shading를 사용했을 때 실행화면의 차이점을 설명하라. 각각 실행결과 화면(wireframe mode와 solid drawing mode 모두)과 코드의 차이점을 보여라. (10점)

Gouraud shading은 일명 smooth shading라고도 불리며, 폴리곤의 각 정점의 색을 linear interpolation하여 내부를 부드럽게 보간하는 방식이다. (**Per-vertex shading**) 정점에서 lighting연산이 된 Gouraud shading을 Quad에 사용했을 시 정점이 4개이므로 제대로 된 명암표현이 되지 않는다. 반면 Grid를 사용했을 시 정점이 10000개이므로 Gouraud shading의

Quad와 비교하여 좀 더 부드럽고 자연스러운 명암이 표현이 된다.

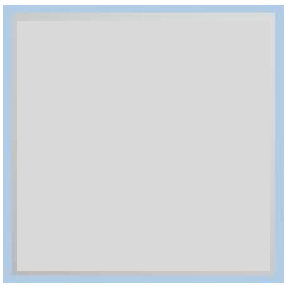
Phong shading은 (Gouraud shading이 정점의 색을 이용하여 내부를 보간하는 반면) 폴리곤의 각 정점의 normal을 가지고 폴리곤 내부의 각 픽셀마다 normal이 보간되어 specular reflection이 더욱 정확하게 나타나게 하는 방식이다. (**Per-pixel shading**)
pixel에서 lighting연산이 된 Phong shading을 Quad에 사용했을 시 정점의 법선벡터를 보간함으로서 Gouraud shading과 비교하여 제대로 된 명암표현을 (즉 더 자연스러운 경면광 표현) 할 수 있다.
Grid를 사용했을 시 Quad와 비교하여 좀더 나은 경면광 표현을 볼 수 있다. 또한 Gouraud shading된 Grid가 색의 보간으로 인한 뭉게짐 현상이 나타나는 반면 Phong shading의 Grid는 경면광 표현이 제대로 표현된다.

5. 다음 OpenGL 텍스쳐 매핑(Texture Mapping)과 블렌딩(Blending) 질문에 답하라. (15점)

1) Quad의 각 꼭지점 p1(-2,-2,0), p2(2,-2,0), p3(2,2,0), p4(-2,2,0)에 텍스쳐 좌표(texture coordinates)를 t1(-1,-1), t2(0,-1), t3(0,0), t4(-1,0)로 지정했을 때, texture wrapping 방식 (1)GL_REPEAT, (2) GL_MIRRORED_REPEAT, (3)GL_CLAMP_TO_EDGE 따른 각각 출력 결과 실행화면을 보여라. 그리고 t1(-2,-2), t2(0,-2), t3(0,0), t4(-2,0)로 지정했을 때, texture wrapping 방식 (4)GL_REPEAT, (5) GL_MIRRORED_REPEAT, (6)GL_CLAMP_TO_EDGE 따른 각각 출력 결과 실행화면을 보여라. (opengl.jpg 사용할 것) (10점)

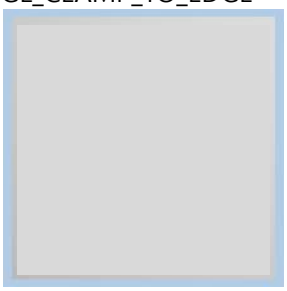
(1) GL_REPEAT

(2) GL_MIRRORED_REPEAT


(3) GL_CLAMP_TO_EDGE

(4) GL_REPEAT

(5) GL_MIRRORED_REPEAT


(6) GL_CLAMP_TO_EDGE

2) Blending을 사용한 Multi-texturing의 원리를 설명하라. 여기에서 사용한 glBlendFunc(GL_ONE, GL_ONE); 또는 glBlendFunc(GL_ZERO, GL_SRC_COLOR); 블렌딩 함수는 각각 무엇인가? (5점)

블렌딩을 사용한 다중패스 멀티 텍스쳐링 (multi-pass multi-texturing) 방식은 동일한 다각형 자체를 여러 번 렌더링하는 것으로, 배경이미지를 먼저 그리고 난 후, 그 위에 라이트매핑 이미지를 합성(blending)하여 여러 번 다각형을 그리는 것이다.
glBlendFunc(GL_ONE, GL_ONE) 블렌딩함수는 뎃셈 블렌딩 (add blending)으로 배경 이미지와 라이트 맵 이미지가 합산된 결과이다.

학과 _____ 학번 _____ 이름 _____

6. 본인의 이름(모빌 예제처럼 계층적 구조를 가지고, 움직이는)을 각각 geometryPositionColor, geometryPositionNormal, geometryPositionNormalTexture을 사용하여 프로그램을 만들어서 실행결과 화면을 보여라. source code와 실행파일을 포함한 프로젝트 파일 전체를 같이 제출하라. (20점)

- 끝 -