단국대학교 응용컴퓨터공학 컴퓨터그래픽스 기말고사 (2019년 가을학기) 2019년 12월 9일 월요일

학과 _____ 학번 _____ 이름 _____

기말고사

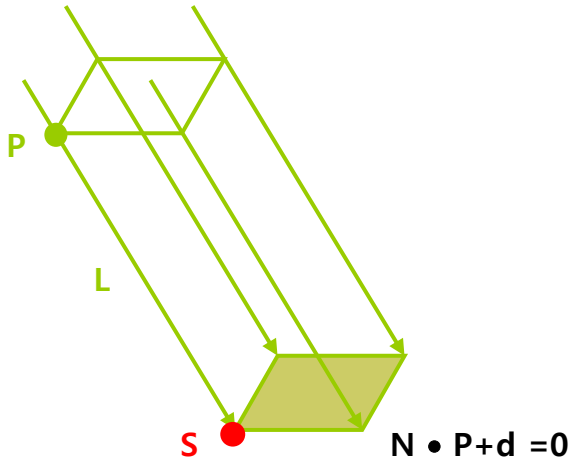
담당교수: 박경신

2019/12/20(금) 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 질문에 답하라. (30점)

1) 다음 그림에서와 같이 3차원 공간의 점 P (x, y, z)가 벡터 L 방향으로 가는 광선 (P(t) = P + t L)은 평면에 한 점 S (Sx, Sy, Sz)를 만난다. 평면의 공식 (N • P + d = 0)을 이용하여 t 를 계산하라. 그리고 S 를 계산하는 공식을 유도하라. (10점)



S = P + tL & N • S + d = 0

$$N \cdot (p + tL) + d = 0$$
$$tN \cdot L = -d - p \cdot N$$
$$t = \frac{-(d + p \cdot N)}{N \cdot L}$$
$$\therefore s = p + \left[\frac{-(d + p \cdot N)}{N \cdot L}\right] L$$

2) glm::ortho(5, 10, 5, 10, 5, 10)와 glm::frustum(5, 10, 5, 10, 5, 10)이 생성하는 Projection 행렬(4x4)을 유도하라. 전체 풀이 과정을 보일 것. (10점)

$$\text{ortho} = ST = \begin{pmatrix} \frac{2}{5} & 0 & 0 & 0 \\ 0 & \frac{2}{5} & 0 & 0 \\ 0 & 0 & -\frac{2}{5} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -7.5 \\ 0 & 1 & 0 & -7.5 \\ 0 & 0 & 1 & 7.5 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{5} & 0 & 0 & -3 \\ 0 & \frac{2}{5} & 0 & -3 \\ 0 & 0 & -\frac{2}{5} & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

학과 _____ 학번 _____ 이름 _____

$$\text{frustum}=NSH=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&-3&-20\\0&0&-1&0\end{pmatrix}\begin{pmatrix}2&0&0&0\\0&2&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}1&0&\frac{3}{2}&0\\0&1&\frac{3}{2}&0\\0&0&1&0\\0&0&0&1\end{pmatrix}=\begin{pmatrix}2&0&3&0\\0&2&3&0\\0&0&-3&-20\\0&0&-1&0\end{pmatrix}$$

3) 다음에서 R1 행렬과 R2 행렬 결과의 차이를 자세히 설명하라. 코드와 실행결과를 첨부하라. (10점)

```
Camera camera1(FLY);
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    spMain.useProgram();
    View = camera1.lookAt(glm:::vec3(0, 0, 10), glm::eye(0, 0, 0), glm::vec3(0, 1, 0));
    spMain.setUniform("gView", View);
    World = glm::mat4(1.0f);
    drawTeapot();
    glm::mat4 Ry = glm::rotate(glm::mat4(1.0f), M_PI/2.0, glm::vec3(0, 1, 0));
    glm::mat4 Rx = glm::rotate(glm::mat4(1.0f), M_PI/2.0, glm::vec3(1, 0, 0));
    glm::mat4 Rz = glm::rotate(glm::mat4(1.0f), M_PI/2.0, glm::vec3(0, 0, 1));
    glm::mat4 R1 = Rz * Rx * Ry;
    World = glm::translate(glm::mat4(1.0f), glm::vec3(-2.5f, 0, 0)) * R;
    spMain.setUniform("gModel", World);
    drawTeapot();
    glm::mat4 R2 = glm::yawPitchRoll(M_PI/2, M_PI/2, M_PI/2);
    World = glm::translate(glm::mat4(1.0f), glm::vec3(2.5f, 0, 0)) * R;
    spMain.setUniform("gModel", World);
    drawTeapot();
    glutSwapBuffers();
}
```

```
// (1) 왼쪽 (초록색)은 World Coordinate System을 중심으로 회전하는 방식
glm::mat4 Ry = glm::rotate(glm::mat4(1), M_PI/2, glm::vec3(0, 1, 0));
glm::mat4 Rx = glm::rotate(glm::mat4(1), M_PI/2, glm::vec3(1, 0, 0));
glm::mat4 Rz = glm::rotate(glm::mat4(1), M_PI/2, glm::vec3(0, 0, 1));
R = Rz * Rx * Ry;
// WCS의 90도 yaw 회전 후 주전자의 입구는 WCS의 z- (주전자 머리는 WCS의 y+)
// WCS의 90도 추가 pitch 회전 후 주전자의 입구는 y+ (주전자 머리는 WCS의 z+)
// WCS의 90도 추가 roll 회전 후 주전자의 입구는 WCS의 x- (주전자 머리는 WCS의 z+)
// (2) 오른쪽 (파란색)은 Local Coordinate System을 중심으로 회전하는 방식
YPR = glm::yawPitchRoll(M_PI/2, M_PI/2, M_PI/2);
// LCS의 90도 yaw (주전자 머리) 회전 후 주전자의 입구는 WCS의 z- (LCS의 x+축은 WCS의 z-축)
// LCS의 90도 추가 pitch (주전자 입구) 회전 후 주전자의 입구는 z- (주전자 머리는 WCS의 x+)
// LCS의 90도 추가 roll 회전(주전자 옆등) 회전 후 주전자의 입구는 x+ (주전자 머리는 WCS의 z+)
```

학과 _____ 학번 _____ 이름 _____

2. 다음은 조명에 관한 아래의 질문에 답하시오. (30점)

$$I = K_a I_a + \sum_{i=o}^{m-1} f_{att}(d_i)\{K_d I_d (N \cdot L_i) + K_s I_s (R_i \cdot V)^n\}$$

$$I = K_a I_a + \sum_{i=o}^{m-1} f_{att}(d_i)\{K_d I_d (N \cdot L_i) + K_s I_s (N \cdot H_i)^{n'}\}$$

1) 두 가지 형태의 직접 조명 모델 공식에서 램버트 법칙 (Lambert's Law)을 표현해주는 부분을 정확히 서술하라. (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.
조명공식에서는 cosTheta는 법선벡터 (N)과 광원벡터 (L)간의 내적은 두 벡터간의 입사각의 코사인과 비례하며, diffuse reflection과의 곱에서 사용된다. (N • L)

2) 위의 직접 조명 모델 공식에서 카메라에서 바라보는 방향에 직접적으로 영향을 받는 변수를 모두 설명하라. (5점)

퐁 직접 조명 모델에서 정반사 경면광 (specular reflection)은 카메라의 시점에 따라 바뀌게 됨. 위의 공식에서 $(R \cdot V)^n$ 와 $(H \cdot N)^{n'}$
블린 직접 조명 모델은 Reflection 벡터 대신 Halfway 벡터를 사용하여 정반사 경면광을 계산한다. Halfway 벡터 H=(L+V)/|L+V|

3) 위의 직접 조명 모델 공식에서 간접적으로 들어오는 빛과 관련 있는 변수를 모두 설명하라. (5점)

Ka Ia 환경반사 (ambient reflection)는 광원에 직접 노출되지 않는 면에 밝기를 부여함

4) 정반사 (reflection) 벡터, 즉 정반사 물질이 입사 광선을 반사시키는 방향을 수식으로 나타내라. (5점)

R = 2(L • N)N - L

5) Sphere의 slice와 stack의 개수를 16개를 썼을 때와 256개를 썼을 때 Gouraud Shading과 Phong Shading를 사용했을 때 차이점을 설명하라. 4가지 각 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 Sphere (slice=stack=16)에 사용했을 시 제대로 된 명암표현이 되지 않는다. 반면 Sphere (slice=tack=256)를 사용했을 시 Sphere (slice=stack=16)과 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 Sphere (slice=stack=16)에 사용했을 시 정점의 법선벡터를 보간함으로서 Gouraud shading과 비교하여 제대로 된 명암표현을 (즉 더 자연스러운 경면광 표현) 할 수 있다. Sphere (slice=tack=256)를 사용했을 시 Sphere (slice=stack=16)와 비교하여 좀더 나은 경면광 표현을 볼 수 있다.

학과 _____ 학번 _____ 이름 _____

3. 다음 OpenGL 텍스쳐 매핑(Texture Mapping)과 블렌딩(Blending) 질문에 답하라. (40점)

1) Texture Mapping의 Two-part mapping을 자세히 서술하라. (5점)

Texture mapping의 기법은 Two-part mapping으로 한다.

1. 텍스쳐 이미지를 먼저 구(sphere), 원기둥(cylinder), 입방체(cube)와 같은 간단한 3차원 매개변수형 표면에 텍스쳐를 맵핑한다.
2. 두번째 매핑(second mapping)은 중간 객체 (intermediate object) 상의 텍스쳐 값을 원하는 표면 (actual object)에 맵핑하는 것으로, 중간 매개표면으로부터의 법선 사용, 객체 표면으로부터의 법선 사용, 객체 중심으로부터의 벡터 사용 방식이 있다.

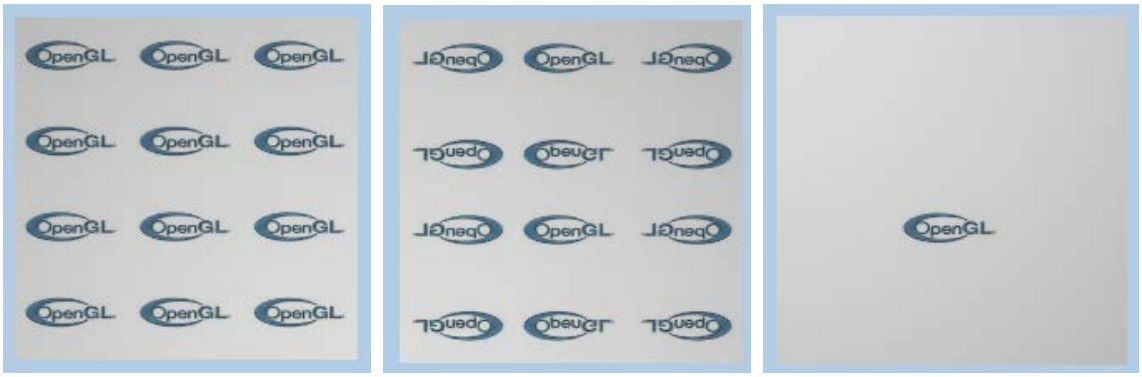
2) 아래 그림과 같이 나타나도록 Quad에 texture coordinate (텍스쳐 좌표)와 texture wrapping 방식을 지정하라. 코드와 실행 결과를 첨부하라. (10점)

Quad (-1,-1,0), (1, -1, 0), (1, 1, 0), (-1, 1, 0)에 텍스쳐 좌표는 (-1, -1), (2, -1), (2, 3), (-1, 3)를 사용.

좌측 GL_REPEAT는 텍스쳐 좌표가 [0, 1]를 넘쳤을 때 텍스쳐를 반복 사용.

중간 GL_MIRRORED_REPEAT는 텍스쳐 좌표가 [0, 1]를 넘쳤을 때 텍스쳐를 반사하면서 반복 사용.

우측 GL_CLAMP_TO_EDGE는 텍스쳐 좌표가 [0, 1]를 넘쳤을 때 텍스쳐를 더 이상 그리지 않음.



3) 다음 텍스쳐 매핑 코드에 주석을 달아라. 아래의 코드에서 확대, 축소, 밉맵 필터와 관련된 부분을 모두 찾아서 적어라. 확대, 축소, 밉맵 필터에 최근점 필터링(NEAREST)나 선형 필터링(LINEAR)를 사용시 차이점을 자세히 서술하라. (5점)

```
bool Texture2D::load(const char * filename, bool mipmaps) {
        GLsizei xdim2,ydim2;
        GLenum type;
        unsigned char *imgPtr;
        // load the original image
        imageData = simage_read_image(filename, &width, &height, &numComponents);
        if (!imageData) {
                printf("ERROR: %s is NOT loaded\n", filename);
                return false;
        }
        //
        xdim2 = 1;
        while (xdim2 <= width) xdim2 *= 2;
        xdim2 /= 2;
```

```
        ydim2 = 1;
        while (ydim2 <= height) ydim2 *= 2;
        ydim2 /= 2;
        // rescale the image
        if ((width != xdim2) || (height != ydim2)) {
                rescaledImageData = simage_resize(imageData, width, height,
                                        numComponents, xdim2, ydim2);
                imgPtr = rescaledImageData;
        }
        else    imgPtr = imageData;
        // set image type by numComponents
        if (numComponents == 4) type = GL_RGBA;
        else if (numComponents == 3) type = GL_RGB;
        else if (numComponents == 2) type = GL_LUMINANCE_ALPHA;
        else if (numComponents == 1) type = GL_LUMINANCE;
        // create and initialize a texture object
        if (uiTexture == 0) glGenTextures(1, &uiTexture);
        glBindTexture(GL_TEXTURE_2D, uiTexture);
        // define texture
        glTexImage2D(GL_TEXTURE_2D, 0, (GLint) type, xdim2, ydim2, 0, type,
                    GL_UNSIGNED_BYTE, imgPtr);
        if (mipmaps) glGenerateMipmap(GL_TEXTURE_2D);
        // set magnification & minification & mipmap filter
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER, magFilter);
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER, minFilter);
        if (minFilter == GL_LINEAR_MIPMAP_NEAREST || minFilter ==
GL_LINEAR_MIPMAP_LINEAR) {
                glGenerateMipmap(GL_TEXTURE_2D); // generate mipmip
                glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_BASE_LEVEL, 0);
                glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAX_LEVEL, 4);
        }
        // set wrapping filter
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, wrapS);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, wrapT);
        if (wrapS == GL_CLAMP_TO_BORDER || wrapT == GL_CLAMP_TO_BORDER) {
                const GLfloat borderColor[] = { 1.0f, 1.0f, 1.0f, 1.0f };
                glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR,
borderColor);
```

```
        }
        glBindTexture(GL_TEXTURE_2D, 0);
        // create and initialize a sampler object
        if (uiSampler == 0)
                glGenSamplers(1, &uiSampler);
        // set magnification & minification & mipmap filter
        glSamplerParameteri(uiSampler, GL_TEXTURE_MAG_FILTER , magFilter);
        glSamplerParameteri(uiSampler, GL_TEXTURE_MIN_FILTER , minFilter);
        glSamplerParameterf(uiSampler, GL_TEXTURE_MAX_ANISOTROPY_EXT, 16.0f);
        // set wrapping filter
        glSamplerParameteri(uiSampler, GL_TEXTURE_WRAP_S, wrapS);
        glSamplerParameteri(uiSampler, GL_TEXTURE_WRAP_T, wrapT);
        if (wrapS == GL_CLAMP_TO_BORDER || wrapT == GL_CLAMP_TO_BORDER) {
                const GLfloat borderColor[] = { 1.0f, 1.0f, 1.0f, 1.0f };
                glSamplerParameterfv(uiSampler, GL_TEXTURE_BORDER_COLOR,
borderColor);
        }
        return true;
}
```

확대필터와 축소필터에 포인트 샘플링 방식인 최근점 필터링 (GL_NEAREST) 사용하면 에일리어싱(aliasing) 문제가 발생함
확대필터와 축소필터에 최근점 필터링 대신 텍셀의 이웃을 포함한 텍셀 그룹의 가중 평균을 사용한 선형 필터링 (GL_LINEAR)를 사용하면 에일리어싱 문제를 줄여줌
밉맵필터는 일련의 축소된 크기의 밉맵 택스쳐를 생성하여, 작은 객체에 텍스쳐 맵핑을 할 시 보간 문제를 줄여주는 필터임.

4) 검정색 배경화면에서 두 개의 텍스쳐 이미지(벽돌 이미지와 라이트맵 이미지)를 겹쳐서 multi-texturing 할 때, glBlendFunc(GL_ZERO, GL_ONE), glBlendFunc(GL_ONE, GL_ONE), glBlendFunc(GL_ZERO, GL_SRC_COLOR), glBlendFunc(GL_SRC_COLOR, GL_DST_COLOR), glBlendFunc(GL_ZERO, GL_ONE_MINUS_SRC_COLOR) 블렌딩 함수는 각각 무엇인지 공식을 적고 자세히 설명하라. 코드와 실행 결과를 첨부하라. (10점)

glBlendFunc(GL_ZERO, GL_ONE) 블렌딩함수는 0*Cs + 1*Cd = Cd 즉 배경 이미지가 그려짐.

glBlendFunc(GL_ONE, GL_ONE) 블렌딩함수는 1*Cs + 1*Cd = Cs + Cd 덧셈 블렌딩 (add blending)으로 배경 이미지와 라이트 맵 이미지가 합산된 결과

glBlendFunc(GL_ZERO, GL_SRC_COLOR) 블렌딩함수는 0*Cs + Cs*Cd = Cs*Cd 곱셈 블렌딩 (modulate blending) 으로 배경 이미지와 라이트 맵 이미지가 합산된 결과

glBlendFunc(GL_SRC_COLOR, GL_DST_COLOR) 블렌딩함수는 Cs*Cs + Cd*Cd 방식으로 배경 이미지와 라이트 맵 이미지가 합산된 결과.

glBlendFunc(GL_ZERO, GL_ONE_MINUS_SRC_COLOR) 블렌딩함수는 0*Cs + (1-Cs)*Cd 방식으로 즉 라이트맵 이미지를 반전시켜서 합성

학과 _____ 학번 _____ 이름 _____

5) **Cylinder(slice=16)를 사용하여 geometryPositionColor, geometryPositionNormal, geometryPositionNormalTexture 방식의 차이점을 설명하라. Cylinder를 그림을 그리고 그 위에 정확한 position (정점)과 normal (법선벡터)과 texture coordinate (텍스쳐 좌표) 값을 표시하라. (10점)**

geometryPositionColor 방식은 각 정점마다 정점(vertex position)과 색(color)을 사용하여 그리며, 조명없이 면을 동일한 색으로 그려줌.
geometryPositionNormal 방식은 각 정점마다 정점(vertex position)과 법선벡터(normal)를 사용하여 그리며, Lighting 과 Material을 사용하여, 면에 색의 음영이 나타나게 그려줌.
geometryPositionNormalTexture는 각 정점마다 정점(vertex position)과 법선벡터(normal)과 텍스쳐좌표(texture coordinate)를 사용하여 그리며, Lighting과 Material과 Texture를 사용하여 면에 색 음영과 텍스쳐가 나타나게 그려줌.

4. 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 -