단국대학교 멀티미디어공학 그래픽스 프로그래밍 기말고사 (2013년 봄학기) 2013년 6월 12일

학과 _____ 학번 _____ 이름 _____

기말고사

담당교수: 단국대학교 멀티미디어공학전공 박경신

- **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 안 함. 답에는 반드시 네모를 쳐서 확실히 표시할 것.**
- **답안지에 학과, 학번, 이름 외에 본인의 암호를 기입하면 성적공고시 학번 대신 암호를 사용할 것임.**

1. 다음은 OpenGL 프로그램의 일부이다. 아래의 질문에 답하시오. (70점)

```
GLuint initTexture(const char * flename, GLint wrap, GLint mag, GLint min)
{
   unsigned char *imageBuffer;
   int imgWidth = 0, imgHeight = 0, num = 0;
   imageBuffer = simage_read_image(filename, &imgWidth, &imgHeight, &num);
   GLuint textureID;
   if (imageBuffer != NULL) {
      glGenTextures(1, &textureID);
      glBindTexture(GL_TEXTURE_2D, textureID);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, wrap);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, wrap);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, mag);
      glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER, min);
      glTexImage2D(GL_TEXTURE_2D, 0, num == 3 ? GL_RGB : GL_RGBA, imgWidth, imgHeight,
                        0, num == 3 ? GL_RGB : GL_RGBA, GL_UNSIGNED_BYTE, imageBuffer);
      if (minfilter == GL_NEAREST || minfilter == GL_LINEAR) { }
      else {
            glGenerateMipmap(GL_TEXTURE_2D);
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_BASE_LEVEL, 0);
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAX_LEVEL, 4);
      }
      glBindTexture(GL_TEXTURE_2D, 0);
   }
}

GLuint setSquareData()
{
   squareVertices.push_back(glm::vec3(-1.0f, -1.0f, 0.0f));
   squareVertices.push_back(glm::vec3( 1.0f, -1.0f, 0.0f));
   squareVertices.push_back(glm::vec3( 1.0f, 1.0f, 0.0f));
   squareVertices.push_back(glm::vec3(-1.0f, 1.0f, 0.0f));
   squareNormals.push_back(glm::vec3(0.0f, 0.0f, 1.0f));
   squareNormals.push_back(glm::vec3(0.0f, 0.0f, 1.0f));
   squareNormals.push_back(glm::vec3(0.0f, 0.0f, 1.0f));
   squareNormals.push_back(glm::vec3(0.0f, 0.0f, 1.0f));
   squareTextureCoords.push_back(glm::vec2(-1.0f, -1.0f));
   squareTextureCoords.push_back(glm::vec2( 2.0f, -1.0f));
   squareTextureCoords.push_back(glm::vec2( 2.0f, 3.0f));
   squareTextureCoords.push_back(glm::vec2(-1.0f, 3.0f));
   squareIndices.push_back(0); squareIndices.push_back(1); squareIndices.push_back(2);
   squareIndices.push_back(0); squareIndices.push_back(2); squareIndices.push_back(3);
```

```
    // 중간 생략…
    return vao;
}

GLuint setCylinderData(float radius=1.0f, float height=2.0f, int slices=16)
{
    glm::vec3 vertex;
    float xTexCoord = 0.0f;
    float theta = (float) (2*M_PI/slices);
    for (int i=0; i<=slices; i++) {
        vertex[0] = radius * cosf(theta * i);
        vertex[1] = -height/2.0f;
        vertex[2] = radius * sinf(theta * i);
        cylinderVertices.push_back(vertex);
        cylinderNormals.push_back(_____1_____);
        cylinderTextureCoords.push_back(glm::vec2(xTexCoord, 0.0f));
        numCylinderVertices++;
        vertex[0] = radius * cosf(theta * i);
        vertex[1] = height/2.0f;
        vertex[2] = radius * sinf(theta * i);
        cylinderVertices.push_back(vertex);
        cylinderNormals.push_back(_____2_____);
        cylinderTextureCoords.push_back(glm::vec2(xTexCoord, 1.0f));
        numCylinderVertices++;
        xTexCoord += 1.0f/slices;
    }
    // 중간 생략…
    return vao;
}

void init()
{  // 중간 생략…
    texture[0] = initTexture("opengl.jpg", GL_REPEAT, GL_NEAREST, GL_NEAREST);
    texture[1] = initTexture("opengl.jpg", GL_REPEAT, GL_LINEAR, GL_LINEAR_MIPMAP_LINEAR);
    texture[2] = initTexture("opengl.jpg", GL_REPEAT, GL_LINEAR, GL_LINEAR);
    texture[3] = initTexture("opengl.jpg", GL_MIRRORED_REPEAT, GL_LINEAR, GL_LINEAR);
    texture[4] = initTexture("opengl.jpg", GL_CLAMP_TO_EDGE, GL_LINEAR, GL_LINEAR);
    texture[5] = initTexture("rock.jpg", GL_REPEAT, GL_LINEAR, GL_LINEAR);
    texture[6] = initTexture("lightmap.png", GL_REPEAT, GL_LINEAR, GL_LINEAR);
    squareVAO = setSquareData();
}

void drawTextureQuad(int textureID)
{
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, textureID);
    glBindVertexArray(squareVAO);
    glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);
    glBindVertexArray(0);
    glBindTexture(GL_TEXTURE_2D, 0);
}

void draw()
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  projection = glm::perspective(45.0f, 1.0f, 1.0f, 100.0f);
  view = glm::lookAt(glm::vec3(4, 0, 0), glm::vec3(0, 0, 0), glm::vec3(0, 1, 0));
```

```
    // 중간 생략…
    drawTextureQuad(texture[0]); // 중간 생략…
    drawTextureQuad(texture[1]); // 중간 생략…
    drawTextureQuad(texture[2]); // 중간 생략…
    drawTextureQuad(texture[3]); // 중간 생략…
    drawTextureQuad(texture[4]); // 중간 생략…
    // 중간 생략…
    drawTextureQuad(texture[5]);
    glDepthFunc(GL_LEQUAL);
    glEnable(GL_BLEND);
    glBlendFunc(GL_ONE, GL_ONE);
    drawTextureQuad(texture[6]);
    glDepthFunc(GL_LESS);
    glDisable(GL_BLEND);
}
```

1) 위의 코드에서 view matrix와 projection matrix가 무엇인지 자세히 설명하라. (5점)

2) 위의 코드에서 *glm::lookAt(glm::vec3(4, 0, 0), glm::vec3(0, 0, 0), glm::vec3(0, 1, 0));* 함수가 생성한 view matrix를 유도하라. (10점)

peudo code: n = eye – at; n.normalize(); u = up x n; u.normalize(); v = n x u; v.normalize();

View[0][0] = u[0]; View[1][0] = u[1]; View[2][0] = u[2]; View[3][0] = - u ・eye;

View[0][1] = v[0]; View[1][1] = v[1]; View[2][1] = v[2]; View[3][1] = - v ・eye;

View[0][2] = n[0]; View[1][2] = n[1]; View[2][2] = n[2]; View[3][2] = - n ・eye;

View[0][3] = 0; View[1][3] = 0; View[2][3] = 0; View[3][3] = 1;

$$View = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

3) 위에 코드에서 setCylinderData(…) 함수 내부에 cylinderNormals.push_back(__1__)과 (__2__) 안에 들어갈 내용은 무엇인가? (5점)

4) 위의 코드에서 glDrawElements(...) 함수가 어떻게 사각형을 그리는지 (정점 위치, 법선 벡터, 텍스쳐 좌표, 인덱스를 표시할 것)을 설명하라. (5점)

5) 텍스쳐 맵핑의 밉맵 필터 (Mipmap filter)를 자세히 설명하라. 그리고 위 코드에서 밉맵과 관련된 부분을 모두 찾아서 적어라. (5점)

6) 위 코드에서 텍스쳐 매핑의 확대, 축소 필터링(Filtering)이 무엇인지, texture[0], texture[1], texture[2]로 바인딩하여 그렸을 때 나타나는 차이점이 무엇인지 자세히 설명하라. (10점)

7) 위 코드에서 texture[2], texture[3], texture[4]로 텍스쳐 바인딩하여 그렸을 때 나타나는 차이점을 그림으로 그려서 자세히 설명하라. (10점)

8) 위 코드에서 setSquareData() 함수 내에 다음 부분이 아래와 같이 변했을 때, drawTextureQuad(texture[3]); 출력결과를 그려라. (5점)

```
squareVertices.push_back(glm::vec3(-0.5f, -0.5f, 0.0f));
squareVertices.push_back(glm::vec3( 0.5f, -0.5f, 0.0f));
squareVertices.push_back(glm::vec3( 0.5f, 0.5f, 0.0f));
squareVertices.push_back(glm::vec3(-0.5f, 0.5f, 0.0f));
squareTextureCoords.push_back(glm::vec2(-1.0f, -1.0f));
squareTextureCoords.push_back(glm::vec2( 0.0f, -1.0f));
squareTextureCoords.push_back(glm::vec2( 0.0f, 0.0f));
squareTextureCoords.push_back(glm::vec2(-1.0f, 0.0f));
```

9) 위 코드에서 draw() 함수 내에 *이탤릭체 부분*이 어떻게 동작하는지 자세히 설명하라. 이 부분에서 *glDepthFunc(GL_LEQUAL);*는 왜 필요한 지? (10점)

10) 위 코드에서 draw() 함수 내에 *이탤릭체 부분*에서 *glBlendFunc(GL_ONE, GL_ONE);*는 어떤 블렌딩 함수인가? 만약 배경 이미지만 나타나는 블렌딩 (background only)을 원한다면 어떻게 바꿔야 하는가? (5점)

2. 다음은 조명 (lighting)에 관한 문제이다. 아래의 질문에 답하시오. (30점)

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

1) 퐁 직접 조명 모델 (Phong Reflection Model)에서 환경반사 (ambient reflection), 난반사 (diffuse reflection), 정반사 (specular reflection)가 무엇인지, 위의 공식에서 어느 부분인지 설명하라. (5점)

2) 정반사(specular reflection)에 사용하는 반사 벡터가 무엇인지 자세히 설명하라. (5점)

3) Per-pixel lighting(일명, Phong shading)이 무엇인지 자세히 설명하라. 이것이 per-vertex lighting과의 차이점은 무언인가? (10점)

4) 다음은 프래그먼트 쉐이더(fragment shader)코드의 일부를 보여주고 있다. 밑줄 친 부분의 코드를 자세히 설명하라. (10점)

```
void main()
{
    vec3 MaterialDiffuseColor = texture2D(gTextureSampler, TexCoordPass).rgb;
    vec3 MaterialAmbientColor = vec3(0.1,0.1,0.1) * MaterialDiffuseColor;
    vec3 MaterialSpecularColor = vec3(0.3,0.3,0.3);
    float distance = length(gLightPosition - PositionWorldPass);
    vec3 N = normalize(NormalViewPass);
    vec3 L = normalize(LightDirectionViewPass);
    float cosTheta = clamp(dot(N, L), 0, 1);
    vec3 V = normalize(EyeDirectionViewPass);
    vec3 R = reflect(-L, N);
    float cosAlpha = clamp(dot(V,R), 0, 1);
    Color = MaterialAmbientColor +
        MaterialDiffuseColor * gLightColor * cosTheta / (distance*distance) +
        MaterialSpecularColor * gLightColor * pow(cosAlpha,5) / (distance*distance);
}
```

3. 다음은 moglclass의 SimpleMobile 클래스의 변형인 SimpleObject 클래스이다. 이 클래스가 동작하는 그 출력 결과를 그림으로 그려라. (extra 10점).

```
void SimpleObject::init()
{
  part1 = Cube(4.0f);
  part2 = Cylinder(2.0f, 4.0f, 16);
  part3 = Torus(1.5f, 0.5f, 32, 16);
  part4 = Sphere(2.0f, 16, 16);
}
```

```
void SimpleObject::draw(Program* p, glm::mat4 projection, glm::mat4 view, glm::mat4
model)
{
   p->useProgram();
   p->setUniform("gProjection", projection);
   p->setUniform("gView", view);

   glm::mat4 m1 = model * bodyTransform;
   p->setUniform("gModel", m1);
   part1.draw();

   glm::mat4 m2 = m1 * partTransform[0];
   p->setUniform("gModel", m2);
   part2.draw();

   glm::mat4 m3 = m2 * partTransform[1] * partTransform[3];
   p->setUniform("gModel", m3);
   part3.draw();

   glm::mat4 m4 = m2 * partTransform[2] * partTransform[3];
   p->setUniform("gModel", m4);
   part4.draw();
}

bool SimpleObject::update(float deltaTime)
{
   angle = angle - 180.0f * (float) (deltaTime) * 0.0001f;
   bodyTransform = glm::rotate(glm::mat4(1.0f), angle, glm::vec3(0, 1, 0));
   partTransform[0] = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, -4.0f, 0.0f));
   partTransform[1] = glm::translate(glm::mat4(1.0f), glm::vec3(-2.0f, -4.0f, 0.0f));
   partTransform[2] = glm::translate(glm::mat4(1.0f), glm::vec3(2.0f, -4.0f, 0.0f));
   partTransform[3] = glm::rotate(glm::mat4(1.0f), angle, glm::vec3(0.0f, 0.0f, 1.0f));
   return true;
}
```

-끝-