Comp380

Student Name & ID: Cho In-Young (조인영, ciy405x@kaist.ac.kr) 20150720

Programming Assignment #6

Due May 30th (Tue) 11:59 PM

```
void normalize(glm::vec4 * v) { // normalize vector
	double div = pow( pow((*v)[0], 2) + pow((*v)[1], 2) + pow((*v)[2], 2), 0.5 );
	for (int i = 0; i < 3; i++) {
		(*v)[i] /= div;
	}
}
//-------------------------------------------------------------------------------------------------------
bool MyGL::ComputeLighting( vector<GLVertex> &verts ) {
	if( !_doLighting )
		return false;

	// Phong Illumination
	if( light.enabled ) {
		for (int i = 0; i < (int)verts.size(); i++) {

			double NL, VR; // 각각 N*L, V*R 내적
			glm::vec4 E, L, V;
			glm::vec3 N, R;

			//definition
			N = verts[i].normal;
			E = { 0,0,1,1 }; // eye position

			//calculation
			L = light.position - verts[i].position; normalize(&L); // 입사광 방향벡터. 정규화
			V = E - verts[i].position;              normalize(&V); // 시선벡터. 정규화
			NL = N[0] * L[0] + N[1] * L[1] + N[2] * L[2];

			for (int j = 0; j < 3; j++) {// R = 2*(NL)N - L, 이미 normalized
				R[j] = 2 * (NL) * verts[i].normal[j] - L[j];
			}
			VR = V[0] * R[0] + V[1] * R[1] + V[2] * R[2];

			glm::vec4 d = (light.diffuse * material.diffuse); // k_d * I_d
			glm::vec4 s = (light.specular * material.specular); // k_s * I_s

			// 벡터의 각 성분에 diffuse term 에는 max(NL,0)을, specular term 에는 max(VR,0)^exponent 을 각각
곱한다.
			d *= __max(0, NL);
			s *= pow(__max(0, VR), material.shininess);

			verts[i].color = light.ambient * material.ambient + d + s;  // Phong Illumination Model
		}
	}
	return true;
}
```

```
// someCalculation: colorZ, color0, colorT를 이용하여 attribute(color말고도 가능, tex, normal, ...) interpolate에 필요한 계수를 계산하여 mulOut에 저장.
void someCalculation(GLVertex verts[3], glm::vec4 colorZ, glm::vec4 color0, glm::vec4 colorT, glm_vec4 * mulOut) { // 저번 과제 제출 시, RasterizeTriangle 에서 썼던 코드를 함수로 묶은 것에 불과. 텍스쳐 좌표 interpolate에도 사용가능.
        // Solving for Linear Interpolation Equations
        // [Ar Br Cr 0]             [r0 r1 r2 0]       [(e0)^t 0]
        // [Ag Bg Cg 0]             [g0 g1 g2 0]       [(e1)^t 0]
        // [Ab Bb Cb 0]  =          [b0 b1 b2 0]  *    [(e2)^t 0]  /  (div)    ..............  by the Formula on
        // [Aa Ba Ca 0]             [a0 a1 a2 0]       [0 0 0  0]         p.22, Lecture08.pdf  (slightly modified)
        // mulOut       =           in1          *     in2  라고 쓰자.


        // in1 부터 구하자.
        glm::vec4 v0 = verts[0].position;
        glm::vec4 v1 = verts[1].position;
        glm::vec4 v2 = verts[2].position;
        double x0 = v0[0], x1 = v1[0], x2 = v2[0],
               y0 = v0[1], y1 = v1[1], y2 = v2[1],
               z0 = v0[2], z1 = v1[2], z2 = v2[2];

        glm_vec4 initial = { 0,0,0,0 };
        glm_vec4 color0 = { colorZ[0], colorZ[1], colorZ[2], colorZ[3] };
        glm_vec4 color1 = { color0[0], color0[1], color0[2], color0[3] };
        glm_vec4 color2 = { colorT[0], colorT[1], colorT[2], colorT[3] };

        glm_vec4 inT[4] = { color0,
                            color1,
                            color2,
                            initial };

        glm_vec4 in1[4];
        glm_mat4_transpose(inT, in1);

        double div = (x1*y2 - x2*y1) - (x0*y2 - x2*y0) + (x0*y1 - x1*y0);

        // in2 를 계산하자.
        glm_vec4 in2[4] = { { y1 - y2, x2 - x1, x1*y2 - x2*y1, 0 },
                            { y2 - y0, x0 - x2, x2*y0 - x0*y2, 0 },
                            { y0 - y1, x1 - x0, x0*y1 - x1*y0, 0 },
                            initial,
                           };

        // mulOut를 계산하자.
        glm_mat4_mul(in2, in1, mulOut);
}
```

```
someCalculation(verts, verts[0].color, verts[1].color, verts[2].color, mulOut); // mulOut 에 color의 interpolation 계수들을 계산해서 넣는다.
        someCalculation(verts, texCoordZ, texCoord0, texCoordT, texMulOut); // texMulOut 에 texture의 interpolation 계수들을 계산해서 넣는다.

        glm_vec4 rgba; // glm_vec4 -> glm::vec4로의 변환을 위해 임시저장용.
        glm_vec4 t; // glm_vec4 -> glm::vec4로의 변환을 위해 임시저장용.

        glm::vec4 color;
        glm::vec2 texCoord;

        for (int x = x_min; x <= x_max; x++) { // total bounding box를 설정.
                for (int y = y_min; y <= y_max; y++) {
                        if (insideTriangle(x, y, v0, v1, v2) || insideTriangle(x, y, v0, v2, v1)) { // 삼각형 버텍스들의 orientation에 독립적으로 내.외부 판별.

                                glm_vec4 vecMulIn2 = { x, y, 1, 0 };

                                // edit
                                // color를 계산하자.
                                if (textureEnabled) { // 만약 텍스쳐를 입혀야 한다면,
                                        if (texture.id != 0) { // texMulOut를 이용해 interpolate 한다.
                                                t = glm_vec4_mul_mat4(vecMulIn2, texMulOut);
                                                texCoord = { t.m128_f32[0] / div, t.m128_f32[1] / div };  // 내삽완료.
                                                int x = (int)(texCoord[0] * (texture.width - 1) + 0.5); // 소숫점 첫째 자리 반올림으로 Nearest Neighbor Filtering 구현
                                                int y = (int)(texCoord[1] * (texture.height - 1) + 0.5);
                                                color = texture.GetPixel(x, y, 0);
                                        }
                                }
                                else {  // 텍스쳐가 필요없다면, PA5와 같다.
                                        rgba = glm_vec4_mul_mat4(vecMulIn2, mulOut);
                                        color = { rgba.m128_f32[0] / div, rgba.m128_f32[1] / div,
                                                        rgba.m128_f32[2] / div, rgba.m128_f32[3] / div };
                                }



                                // interpolating for Z-buffering
                                z = z0 + A * (x - x0) + B * (y - y0);

                                //z_buffering
                                if (depthTestEnabled){
                                        // d를 누르면 depthTest를 해서 앞쪽 색을 출력한다.(앞쪽 색만 저장한다)
                                        // 구의 Illumination에서, 각 폴리곤을 출력하는 순서에 depend하게 색이 결정되버려서 뒷면의 그림자 색이 출력되는 경우가 발생한다.
                                        // 이때 depthTest를 켜주면 문제가 해결된다.
                                        if (frameBuffer.GetDepth(x, y) > z) {
                                                frameBuffer.SetDepth(x, y, z);
                                                frameBuffer.SetPixel(x, y, color);
                                        }
                                }
                                else if (!depthTestEnabled) {
                                        frameBuffer.SetPixel(x, y, color);
                                }
                                //end
                        }
                }

        }
```