REPORT

과제 3: 조명과 관찰자 제어

과목명	컴퓨터그래픽스 (SW)-1분반	담당교수	송인식
학 번	32221741	전 공	소프트웨어
이 름	박성현	제 출 일	2023-11-19(일)

DKU 단국대학교 DANKOOK UNIVERSITY

내용

1. 개발환경

OS : Windows 10/11

Editor : Visual Studio Code

Browser : Microsoft Edge

---

2. 설계

1. 과제 2로 작성한 'rotatingPyramid.html', 'rotatingPyramid.js'를 기반으로 과제 3을 수행한다.
2. cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/SEVENTH_EDITION/CODE/06/shadedCube.html와 cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/SEVENTH_EDITION/CODE/06/shadedSphere3.html를 참고하여 코드를 수정한다.
    A. rotatingPyramid와 구조적으로 비슷한 shadedCube을 참고하여 조명을 추가한다.
    B. shadedSphere3를 참고하여 관찰자를 추가한다.
    C. 제어하는 버튼을 더 추가한다.

3. 구현

rotatingPyramid.html

```
<title>Rotating Pyramid</title>
<script id="vertex-shader" type="x-shader/x-vertex">

attribute  vec4 vPosition;
attribute  vec4 vColor;
attribute  vec3 vNormal;
varying vec4 fColor;

uniform vec4 ambientProduct, diffuseProduct, specularProduct;
uniform mat4 modelViewMatrix;
uniform mat4 projectionMatrix;
uniform vec4 lightPosition;
uniform float shininess;

void main()
{

    vec3 pos = -(modelViewMatrix * vPosition).xyz;

    vec3 light = lightPosition.xyz;
    vec3 L = normalize(light - pos);

    vec3 E = normalize(-pos);
    vec3 H = normalize(L + E);

    vec4 NN = vec4(vNormal, 0);

    vec3 N = normalize((modelViewMatrix*NN).xyz);

    vec4 ambient = ambientProduct;

    float Kd = max(dot(L, N), 0.0);
    vec4  diffuse = Kd * diffuseProduct;

    float Ks = pow(max(dot(N, H), 0.0), shininess);
    vec4  specular = Ks * specularProduct;

    if(dot(L, N) < 0.0) {
        specular = vec4(0.0, 0.0, 0.0, 1.0);
    }

    gl_Position = projectionMatrix * modelViewMatrix * vPosition;
    fColor = vColor * (ambient + diffuse + specular);
    fColor.a = 1.0;
}
</script>
```

shadedCube.html에서 참고하여 작성한 조명을 구현하는 vertex-shader 스크립트 부분이다. 원래 여기에서 회전 행렬으로 vPosition을 회전시켰으나 js의 render()에서 회전시키도록 변경하였다. 그래서 조명은 고정하고 피라미드만 회전시킬 수 있었다. 여기서 포인트는 gl_Position은 프로젝션 * 모델뷰 * vPosition 순으로 곱해야 한다는 것이다. 그리고 fColor를 ambient + diffuse + specular에 vColor를 곱하여 구하는 것이다.

```
<div>
    speed 0%
    <input id="slider" type="range" min="0" max="100" step="10" value="50" />
    100%
</div>
<button id="xButton">Rotate X</button>
<button id="yButton">Rotate Y</button>
<button id="zButton">Rotate Z</button>
<button id="tButton">Toggle Rotation</button>
<select id="Controls" size="2">
    <option value="0">Spin Faster</option>
    <option value="1">Spin Slower</option>
</select>
<p> </p>
<button id = "Button0">Increase R</button>
<button id = "Button1">Decrease R</button>

<p> </p>
<button id = "Button2">Increase theta</button>
<button id = "Button3">Decrease theta</button>
<button id = "Button4">Increase phi</button>
<button id = "Button5">Decrease phi</button>
```

기존 rotatingPyramid.html에 있던 버튼에서 회전을 토글하는 tButton, shadedSphere3.html에서 가져온 Button0~5를 추가하였다.

---

rotatingPyramid.js

```
var vertices = [
    vec4(-0.5, -0.5, 0.5, 1.0), // 좌측 앞
    vec4(0.5, -0.5, 0.5, 1.0), // 우측 앞
    vec4(0.0, 0.5, 0.0, 1.0), // 꼭대기
    vec4(0.5, -0.5, -0.5, 1.0), // 우측 뒤
    vec4(-0.5, -0.5, -0.5, 1.0), // 좌측 뒤
];

var vertexColors = [
    [0.0, 0.0, 0.0, 1.0], // black
    [1.0, 0.0, 0.0, 1.0], // red
    [1.0, 1.0, 0.0, 1.0], // yellow
    [0.0, 1.0, 0.0, 1.0], // green
    [0.0, 0.0, 1.0, 1.0], // blue
];
```

원래 함수 안에 있던 vertices와 vertexColors의 선언을 전역으로 끄집어내어 리팩토링하였다.

```
// 관찰자에 필요한 변수
var near = -10;
var far = 10;
var radius = 1.5;
var theta = 0.0;
var phi = 0.0;
var dr = (5.0 * Math.PI) / 180.0;
var left = -1.0;
var right = 1.0;
var ytop = 1.0;
var bottom = -1.0;
var eye;
var at = vec3(0.0, 0.0, 0.0);
var up = vec3(0.0, 1.0, 0.0);
```

관찰자에 관련한 변수를 예제에서 가져왔다. theta는 기존과 이름이 겹치기 때문에 기존 theta를 rotTheta로 이름을 변경하였다.

```
// 빛 관련 변수
var lightPosition = vec4(1.0, 1.0, 1.0, 0.0);
var lightAmbient = vec4(0.2, 0.2, 0.2, 1.0);
var lightDiffuse = vec4(1.0, 1.0, 1.0, 1.0);
var lightSpecular = vec4(1.0, 1.0, 1.0, 1.0);
var materialAmbient = vec4(1.0, 1.0, 1.0, 1.0);
var materialDiffuse = vec4(1.0, 0.8, 0.8, 1.0);
var materialSpecular = vec4(1.0, 0.8, 0.8, 1.0);
var materialShininess = 100.0;
var ambientColor, diffuseColor, specularColor;
```

조명 관련 변수도 예제에서 가져와 살짝 변경하였다. 내가 작성한 피라미드 코드의 경우 각 면마다 색상을 정하였으므로 벡터의 한 요소가 0.0이 되면 나중에 vColor와 ambientProduct, diffuseProduct 등을 결합하여 최종 색상을 결정할 때 어떤 색상만 조명 효과를 못받게된다. (0을 곱하면 무조건 0이되는 성질로 인하여)

```
// 모델-뷰 매트릭스, 프로젝션 매트릭스
var modelViewMatrix, projectionMatrix;
var modelViewMatrixLoc, projectionMatrixLoc;
```

모델-뷰 매트릭스와 프로젝션 매트릭스를 위한 변수도 만들어준다.

```
70  function triple(a, b, c) {
71      var t1 = subtract(vertices[b], vertices[a]);
72      var t2 = subtract(vertices[c], vertices[a]);
73      var normal = cross(t2, t1);
74      normal = vec3(normal);
75      normals.push(normal);
76      normals.push(normal);
77      normals.push(normal);
78      points.push(vertices[a]);
79      points.push(vertices[b]);
80      points.push(vertices[c]);
81      colors.push(vertexColors[a]);
82      colors.push(vertexColors[a]);
83      colors.push(vertexColors[a]);
84  }
```

colorPyramid() 함수는 그대로지만 triple() 함수는 수정이 필요하였다. 일단 법선 벡터를 계산하여 nomals 배열에 push해주었다. 그리고 정점 배열에 정점들을, 색상 배열에 색상을 push 해 주었다.

```
116     var nBuffer = gl.createBuffer();
117     gl.bindBuffer(gl.ARRAY_BUFFER, nBuffer);
118     gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
119
120     var vNormal = gl.getAttribLocation(program, "vNormal");
121     gl.vertexAttribPointer(vNormal, 3, gl.FLOAT, false, 0, 0);
122     gl.enableVertexAttribArray(vNormal);
```

법선 벡터에 관한 버퍼를 만들어준다. 이는 예제 코드를 참고하여 작성하였다.

```
140     modelViewMatrixLoc = gl.getUniformLocation(program, "modelViewMatrix");
141     projectionMatrixLoc = gl.getUniformLocation(program, "projectionMatrix");
```

모델-뷰 행렬, 프로젝션 행렬 uniform 변수들의 위치를 저장해준다.

```
144     var ambientProduct = mult(lightAmbient, materialAmbient);
145     var diffuseProduct = mult(lightDiffuse, materialDiffuse);
146     var specularProduct = mult(lightSpecular, materialSpecular);
```

앞서 전역으로 만들어준 조명의 요소들로 행렬곱을 해 ambient, diffuse, specular의 product 값을 저장한다.

```
177   document.getElementById("tButton").onclick = function () {flag = !flag;};
178   document.getElementById("Button0").onclick = function () {radius *= 2.0;};
179   document.getElementById("Button1").onclick = function () {radius *= 0.5;};
180   document.getElementById("Button2").onclick = function () {theta += dr;};
181   document.getElementById("Button3").onclick = function () {theta -= dr;};
182   document.getElementById("Button4").onclick = function () {phi += dr;};
183   document.getElementById("Button5").onclick = function () {phi -= dr;};
```

추가한 버튼들의 onclick 이벤트 리스너를 달아준다. 이도 역시 예제에서 가지고 왔다.

```
185   gl.uniform4fv(gl.getUniformLocation(program, "ambientProduct"),flatten(ambientProduct));
186   gl.uniform4fv(gl.getUniformLocation(program, "diffuseProduct"),flatten(diffuseProduct));
187   gl.uniform4fv(gl.getUniformLocation(program, "specularProduct"),flatten(specularProduct));
188   gl.uniform4fv(gl.getUniformLocation(program, "lightPosition"),flatten(lightPosition));
189   gl.uniform1f(gl.getUniformLocation(program, "shininess"), materialShininess);
```

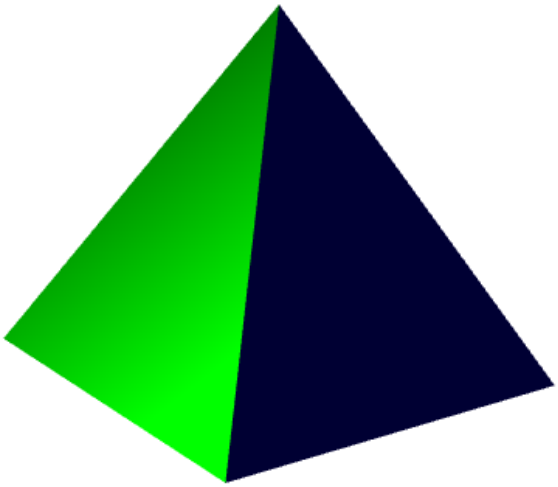
앞서 행렬곱을 마친 ambient, diffuse, specular와 lightPosition, materialShininess를 uniform 변수에 전달한다.

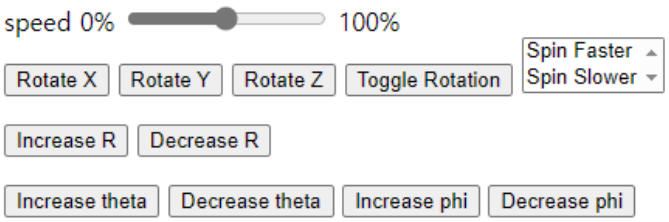
```
194   if (flag) rotTheta[axis] += 2.0;
195   eye = vec3(
196     radius * Math.sin(theta) * Math.cos(phi),
197     radius * Math.sin(theta) * Math.sin(phi),
198     radius * Math.cos(theta)
199   );
200   modelViewMatrix = lookAt(eye, at, up);
201   modelViewMatrix = mult(modelViewMatrix, rotate(rotTheta[xAxis], [1, 0, 0]));
202   modelViewMatrix = mult(modelViewMatrix, rotate(rotTheta[yAxis], [0, 1, 0]));
203   modelViewMatrix = mult(modelViewMatrix, rotate(rotTheta[zAxis], [0, 0, 1]));
204   projectionMatrix = ortho(left, right, bottom, ytop, near, far);
205   gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
206   gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
```

render() 내부에 추가한 코드들이다. eye를 계산하고 앞서 저장한 at, up과 같이 lookAt()으로 카메라 행렬을 만들어 모델-뷰 매트릭스에 저장한다. 그리고 앞서 말했듯 행렬 곱을 통해 원래 vertex shader에서 수행했던 회전을 수행한다. 프로젝션 매트릭스에는 ortho()로 직교투영하도록 하였다. left, right 등의 인자는 앞서 전역으로 만들어 둔 것이다. 그리고 모델-뷰 매트릭스와 프로젝션 매트릭스를 uniform 변수로 전달한다.

---

4. 실행 결과 캡쳐





조명이 제대로 적용되었고, 관측자를 제어하는 버튼도 잘 작동함을 알 수 있었다.

---

5. 소스 코드

소스코드는 GitHub 레포지토리에 있습니다.

https://github.com/ParkSeonghyeon2003/DKU_ComputerGraphics
