Lab05. Viewport

DoHoon Lee Ph.D

Visual & Biomedical Computing(VisBiC) Lab.
School of Computer Science & Engineering
Pusan National University

http://visbic.pusan.ac.kr/

실습 : Camera 설정 및 Multiviewport

Sample Code

- Sample file lab05.html을 실행해 보자
  - 필요한 파일 : lab05.html, lab05.js, lab05-1.html, lab05-1.js Axis.js, initShaders.js, MV.js, webgl-utils.js
- Sample code lab05.html에서 각 버튼의 기능을 설명하시오.
  - 각 버튼의 기능이 어떤 방식으로 구현되었는지 source code를 보고 설명해 보시오.
  - 각 객체의 회전축을 그려보자.
  - drawscene()에서 perspective 함수의 fov를 조절해보자. 가장 크게 나타나는 각도와 가장 작게 나타나는 각도를 찾아보자.
- Sample code lab05-1.html과 그 부속된 파일을 활용하여 실행시키고 show axis의 구현 방법에 대해 설명하시오.

Sample Code

- lookAt()를 사용하여 카메라 위치를 설정해보자. 카메라는 mvMatrix에서 가장 먼저 설정한다.
- In lab05-1.js 에서

  lookAtMatrix 변수 선언

```
var lookAtMatrix;
```

카메라 설정하기

- In drawScene()

```
function drawScene() {
    gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    pMatrix = perspective(45, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0);

    // --- camera setting ----
    lookAtMatrix = lookAt( vec3(0,10,0), vec3(0,0,-10), vec3(0,1,0) );

    // mvMatrix for pyramid
    mvMatrix = lookAtMatrix;
    //mvMatrix = translate(0.0, 0.0, -10.0);
    mvMatrix = mult( mvMatrix, translate(0.0, 0.0, -10.0));
    // --- end of modification

    // --- camera setting ----
    lookAtMatrix = lookAt( vec3(0,10,0), vec3(0,0,-10), vec3(0,1,0) );

    // mvMatrix for pyramid
    mvMatrix = lookAtMatrix;
    //mvMatrix = translate(0.0, 0.0, -10.0);
    mvMatrix = mult( mvMatrix, translate(0.0, 0.0, -10.0));
    // --- end of modification
```

Cube 에 카메라 설정

```
gl.drawArrays(gl.TRIANGLES, 0,
pyramidVertexPositionBuffer.numItems);


 // --- drawing cube
 mvMatrix = lookAtMatrix;
 //mvMatrix = translate(0.0, 0.0, -10.0);
 mvMatrix = mult(mvMatrix, translate(0.0, 0.0, -10.0));


 mvMatrix = mult(mvMatrix, rotate(rTotal, [0, 1, 0]));
 mvMatrix = mult(mvMatrix, translate(-2, 0.0, 0.0));
 mvMatrix = mult(mvMatrix, rotate(rCube, [1, 1, 1]));
```

Cube 회전축을 위한 mvMatrix 수정

```
gl.drawElements(gl.TRIANGLES,
cubeVertexIndexBuffer.numItems, gl.UNSIGNED_SHORT, 0);

 //--drawing axis
 mvMatrix = lookAtMatrix;
//mvMatrix = translate(0.0, 0.0, -10.0);
 mvMatrix = mult(mvMatrix, translate(0.0, 0.0, -10.0));
 mvMatrix = mult(mvMatrix, rotate(rTotal, [0, 1, 0]));
 mvMatrix = mult(mvMatrix, translate(-2, 0.0, 0.0));
 mvMatrix = mult(mvMatrix, rotate(45, [1,  0, 0]));
 mvMatrix = mult(mvMatrix, rotate(-35, [0,  0, 1]));
 if(axis_show) rotateAxisObject(1.9);
```

Camera 위치 이동

```
lookAtMatrix = lookAt( vec3(0,10,0), vec3(0,0,-10), vec3(0,1,0) );
```

- 위에서 카메라 위치를 이동시켜 보자.
  - (0,10,0)을 (0,-5,0)으로 변경하여 실행해 보자.
- 카메라가 보고 있는 지점도 같이 수정해 보자.
  - 값을 조금씩 변경해서 어떤 현상이 일어나는지 고찰해보자.

render() 함수 수정

- 카메라 설정을 render()함수에서 해보자.
- drawScene()에서

```
function drawScene() {
    //gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
    //gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    pMatrix = perspective(45, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0);
```

render() 수정

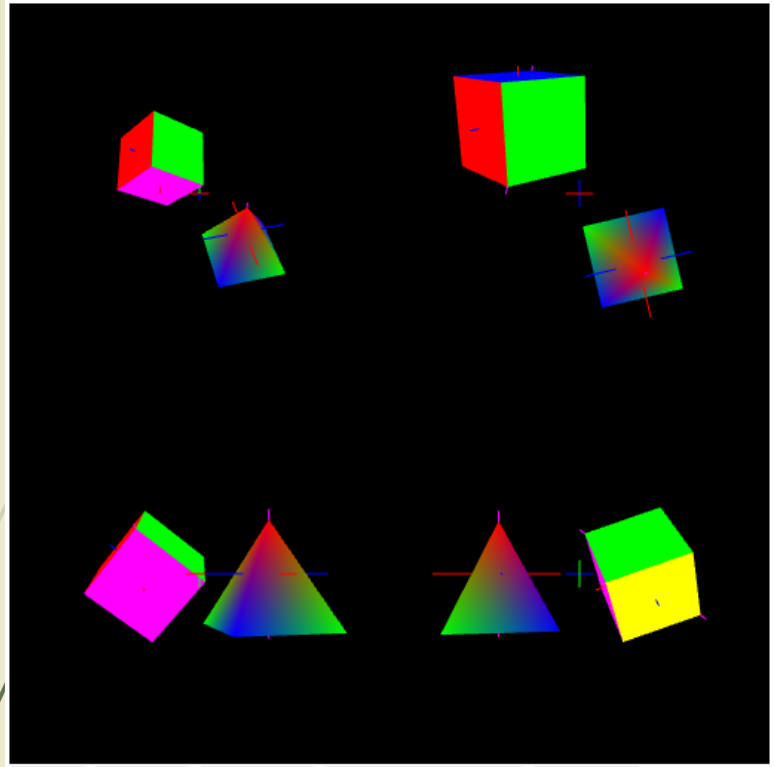
```
function render() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
    // --- camera setting ----
    // 45도
    lookAtMatrix = lookAt( vec3(0, 10, 0), vec3(0, 0, -10), vec3(0, 1, 0));
    gl.viewport(0, gl.viewportHeight/2, gl.viewportWidth/2, gl.viewportHeight/2);
    drawScene();

    // 정면
    lookAtMatrix = lookAt( vec3(0, 0, 0), vec3(0, 0, -10), vec3(0,1,0));
    gl.viewport(gl.viewportWidth/2, gl.viewportHeight/2, gl.viewportWidth/2, gl.viewportHeight/2);
    drawScene();

    setNextTimeScene();
    window.requestAnimFrame(render);
}
```

실습문제-lab05

- 실습문제
  - 화면은 4등분하여 정면, 평면(위->아래), 측면, 평명(아래->우)을 볼 수 있도록 카메라를 설정해보자. (힌트 : viewport() 이용. lookAt()함수로 카메라 설정)
- 도전문제
  - 회전의 속도를 각각 조절(버튼:화면수 만큼)하는 기능을 추가해 보자.
- 참고문제
  - 4등분한 화면의 mvMatrix와 pMatrix의 변화를 보여주는 기능을 추가해 보자.



STOP | Rotation | Revolution | Rotation+Revolution | Hide Axis

pMatrix

2.414	0.000	0.000	0.000
0.000	2.414	0.000	0.000
0.000	0.000	-1.002	-0.200
0.000	0.000	-1.000	0.000

pyramidVp1mvMatrix

-0.228	0.000	0.974	1.281
0.688	0.707	0.161	-1.086
-0.688	0.707	-0.161	-13.056
0.000	0.000	0.000	1.000

cubeVp1mvMatrix

0.010	0.635	-0.773	-1.281
-0.900	0.342	0.269	1.086
0.435	0.693	0.575	-15.228
0.000	0.000	0.000	1.000

pyramidVp2mvMatrix

-0.228	0.000	0.974	1.281
0.000	1.000	0.000	0.000
-0.974	0.000	-0.228	-8.464
0.000	0.000	0.000	1.000

cubeVp2mvMatrix

0.010	0.635	-0.773	-1.281
-0.329	0.732	0.597	0.000
0.944	0.248	0.216	-11.536
0.000	0.000	0.000	1.000