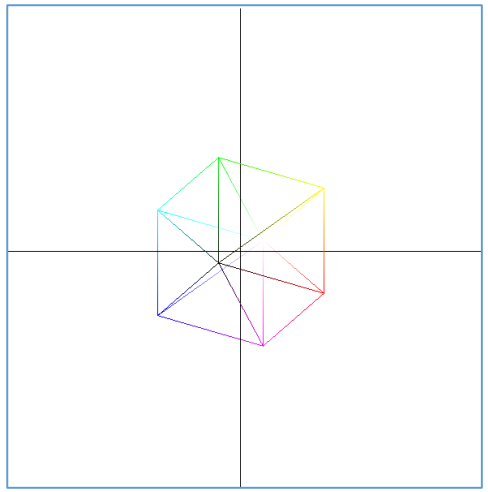
openGL 세이더 만들기 2

2020-2 컴퓨터 그래픽스

세이더 사용하여 도형 그리기 2 (실습 추가)

셰이더 이해하기 (셰이더 구현한 사람은 skip 해도 괜찮습니다.)



원하는 결과물

객체1: 육면체
객체2: 좌표계

객체의 속성

응용 프로그램

버텍스 셰이더

프래그먼트 셰이더

셰이더 프로그램

Vertex Array Object

Vertex Buffer Object

Vertex Buffer Object

Vertex Buffer Object

```
glBindVertexArray                  //VAO 바인드
glBindBuffer                       // 버퍼 오브젝트 바인드
glBufferData                       // 버퍼 오브젝트의 데이터를 생성
glVertexAttribPointer              // 버텍스 속성 데이터의 배열 정의
glEnableVertexAttribArray          // 버텍스 속성 배열 사용가능하게 함
```

셰이더 이해하기 (셰이더 구현한 사람은 skip 해도 괜찮습니다.)

- 속성 데이터

```
// 육면체 데이터:  면 6개에 대한 속성 → 면 6개 → 삼각형 12개
float CubeData [] =
{
        0.5, 0.5, 0.0,          1.0, 0.0, 0.0,          //앞면: 삼각형 1
        -0.5, -0.5, 0.0,        0.0, 0.0, 1.0,          // 포지션, 색상
        0.5, -0.5, 0.0,         1.0, 0.0, 1.0,

        0.5, 0.5, 0.0,          1.0, 0.0, 0.0,          // 앞면: 삼각형 2
        -0.5, 0.5, 0.5,         0.0, 1.0, 1.0,          // 포지션, 색상
        -0.5, -0.5, 0.0,        0.0, 0.0, 1.0
        …
};
```

```
// 좌표계 데이터: 선 2개에 대한 속성
float LineData [] = {
        -1.0f, 0.0f, 0.0f,      0.0, 0.0, 0.0,
        1.0f, 0.0f, 0.0f,       0.0, 0.0, 0.0,

        0.0f, -1.0f, 0.0f,      0.0, 0.0, 0.0,
        0.0f, 1.0f, 0.0f        0.0, 0.0, 0.0
};
```

셰이더 이해하기 (셰이더 구현한 사람은 skip 해도 괜찮습니다.)

- Vertex Shader

```
#version 330 core

layout (location = 0) in vec3 vPos;      //--- 위치 변수: attribute position 0
layout (location = 1) in vec3 vColor;    //--- 컬러 변수: attribute position 1

out vec3 outColor;                       //--- 컬러를 fragment shader로 출력

void main()
{
   gl_Position = vec4 (vPos, 1.0);
   outColor = vColor;                     //--- vertex data로부터 가져온 컬러 입력을 ourColor에 설정
}
```

- Fragment Shader

```
#version 330 core

out vec4 FragColor;
in vec3 outColor;

void main()
{
        FragColor = vec4 (outColor, 1.0f);


}
```

셰이더 이해하기 (셰이더 구현한 사람은 skip 해도 괜찮습니다.)

- VAO와 VBO 만들기

```
void InitBuffer ()
{
        GLuint VAO[2], VBO[2];

        //--- VAO 객체 생성 및 바인딩
        glGenVertexArrays (2, VAO);     // 2개 객체를 위한 VAO
        glGenBuffers (2, VBO);          // 2개 객체에서 각각의 속성을 위한 VBO

        //---큐브데이터: 육면체를 그리기 위한 VAO
        glBindVertexArray (VAO[0]);
        glBindBuffer (GL_ARRAY_BUFFER, VBO[0]);

        //--- vertex data 데이터 입력.
        glBufferData (GL_ARRAY_BUFFER, sizeof (CubeData), CubeData, GL_STATIC_DRAW);

        //--- 위치 속성: 속성 인덱스 0
        glVertexAttribPointer (0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)0);
        glEnableVertexAttribArray (0);

        //--- 색상 속성: 속성 인덱스 1
        glVertexAttribPointer  (1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float),
                                                (void*)(3* sizeof(float)));
        glEnableVertexAttribArray (1);

        //---라인데이터: 좌표계를 그리기 위한 VAO
        glBindVertexArray (VAO[1]);
        glBindBuffer (GL_ARRAY_BUFFER, VBO[1]);

        //--- vertex data 데이터 입력.
        glBufferData (GL_ARRAY_BUFFER, sizeof (LineData), LineData, GL_STATIC_DRAW);

        //--- 위치 속성: 속성 인덱스 0
        glVertexAttribPointer (0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (void*)0);
        glEnableVertexAttribArray (0);

        //--- 색상 속성: 속성 인덱스 1
        glVertexAttribPointer  (1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float),
                                                (void*)(3* sizeof(float)));
        glEnableVertexAttribArray (1);
}
```

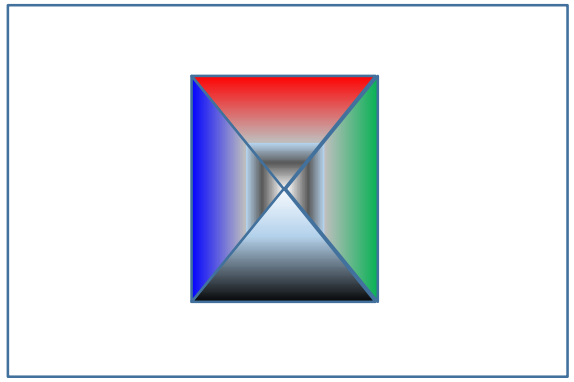
셰이더 이해하기 (셰이더 구현한 사람은 skip 해도 괜찮습니다.)

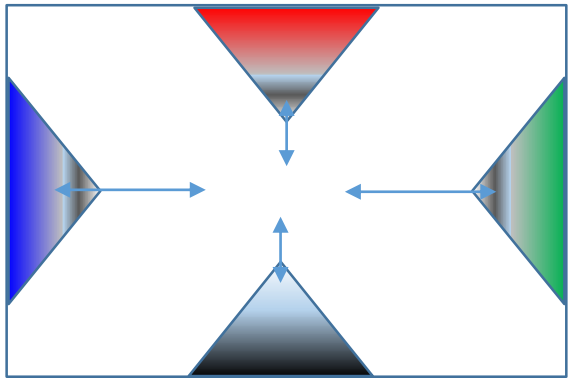
- 그리기 콜백 함수

```
void drawScene ()
{
	glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
	glClear(GL_COLOR_BUFFER_BIT| GL_COLOR_BUFFER_BIT);

	glUseProgram (s_Program );

	glBindVertexArray(VAO[0]);		//--- 육면체에 대한 속성이 연결되어 있는 VAO[0] 사용
	glDrawArrays(GL_TRIANGLES, 0, 36);		// 36개의 버텍스를 사용하여 삼각형 그리기 → 12개 삼각형

	glBindVertexArray(VAO[1]);		//--- 선에 대한 속성이 연결되어 있는 VAO[1] 사용
	glDrawArrays(GL_LINES, 0, 4);		// 4개의 버텍스를 사용하여 선 그리기 → 2개 선 (좌표계)

	glutSwapBuffers();

}
```

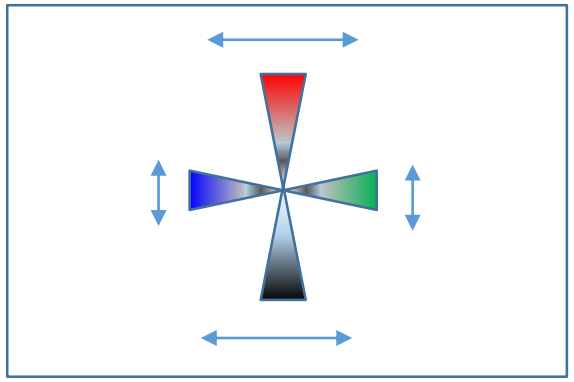
실습 9

- 화면에 삼각형 4개를 사용하여 사각형을 그리고 키보드 명령에 의해 이동하기
  - 화면에 800x600 크기의 윈도우 띄우기
  - 4개의 삼각형을 사각형 모양이 되게 그린다.
    - 꼭지점 한 개가 일치된다.
    - 꼭지점 마다 다른 색을 설정하여 그라데이션으로 그린다.
  - 키보드 명령어
    - t: 삼각형이 각각 위/아래/좌/우 방향으로 이동하고 가장자리에 닿으면 다시 안쪽으로 이동한다.
    - S:좌우로 움직이는 삼각형들은 y값이 축소/제자리로 변동, 상하로 움직이는 삼각형들은 x값이 축소/제자리로 변동되는 애니메이션
    - t: 애니메이션 멈추기





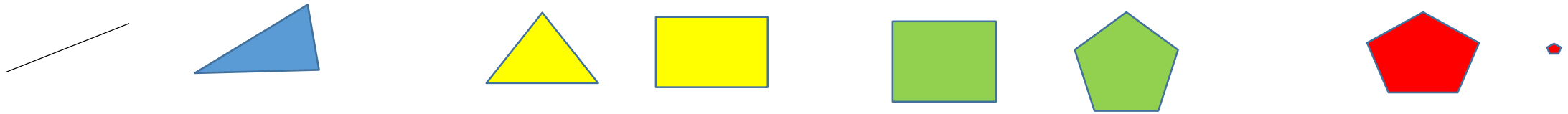
t: 좌우 또는 상하로 이동

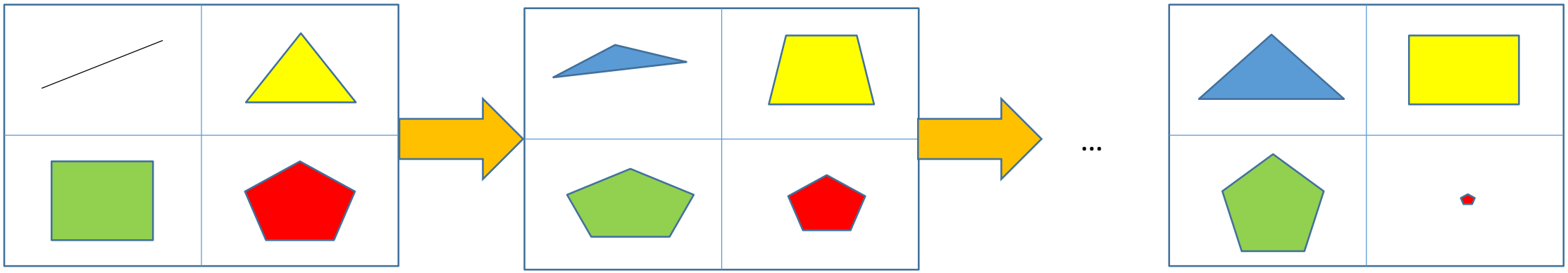


s: x축 또는 y축 값이 축소

실습 10

- 여러 도형 그리기
  - 화면을 가로, 세로 각각 이등분한 위치에 각각 선, 삼각형, 사각형(삼각형 사용), 오각형(삼각형 사용)을 그린다.
  - 네 개의 도형을 단계별로 다른 도형으로 변환시킨다. (애니메이션 적용)
    - 1) 선 → 삼각형
    - 2) 삼각형 → 사각형
    - 3) 사각형 → 오각형
    - 4) 오각형 → 점





실습 11

이번주에는

- 셰이더 사용한 추가 실습문제
  - 버텍스 셰이더와 프래그먼트 셰이더 구현하여 객체 그리기

- 다음 시간에는 좌표계 변환

- 실습 시간에 봅시다!!