WebGL 2D 평행 이동

3D로 넘어가기 전에 잠시 2D를 사용해봅시다. 끝까지 함께 해주세요. 어떤 분들에게는 이 글이 굉장히 당연하겠지만 몇 가지 요점을 써보려고 합니다.

이 글은 WebGL 기초로 시작한 WebGL 관련 시리즈에서 이어집니다. 아직 읽지 않았다면 적어도 첫 번째 글을 먼저 읽고 다시 여기로 돌아오는 게 좋습니다.

평행 이동은 기본적으로 무언가를 "움직이는 것"을 의미하는 멋진 수학적 명칭인데요. 문장을 영어에서 일본어로 옮기는 것도 맞지만 이 경우에는 기하학적 이동을 말합니다. 첫 번째 포스트에서 끝낸 샘플 코드를 사용하면 `setRectangle`에 전달되는 값을 변경하는 것만으로 쉽게 사각형을 이동할 수 있었죠? 다음은 이전 샘플에 기반한 샘플입니다.

먼저 사각형의 `translation`, `width`, `height`, `color` 변수를 만듭니다.

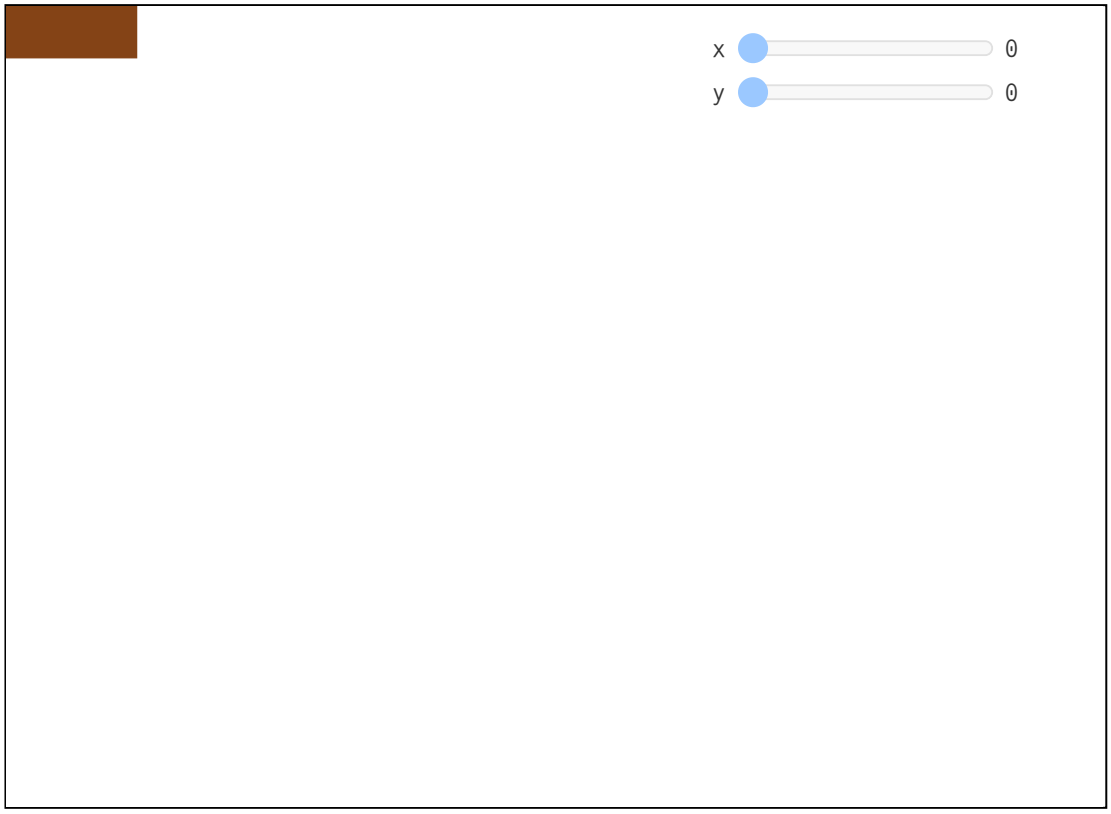
```
var translation = [0, 0];
var width = 100;
var height = 30;
var color = [Math.random(), Math.random(), Math.random(), 1];
```

그런 다음 전부 다시 그리는 함수를 만들어봅시다. `translation`을 업데이트한 후에 이 함수를 호출할 수 있습니다.

```
// 장면 그리기
function drawScene() {
  webglUtils.resizeCanvasToDisplaySize(gl.canvas);

  // 클립 공간에서 픽셀로 변환하는 방법을 WebGL에 지시
  gl.viewport(0, 0, gl.canvas.width, gl.canvas.height);

  // 캔버스 지우기
  gl.clear(gl.COLOR_BUFFER_BIT);

  // 프로그램(셰이더 쌍) 사용 지시
  gl.useProgram(program);

  // 위치 속성 활성화
  gl.enableVertexAttribArray(positionLocation);

  // 위치 버퍼 할당
  gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);

  // 사각형 설정
  setRectangle(gl, translation[0], translation[1], width, height);
```

```
    // positionBuffer(ARRAY_BUFFER)에서 데이터 가져오는 방법을 속성에 지시
    var size = 2;          // 반복마다 2개의 컴포넌트
    var type = gl.FLOAT;   // 데이터는 32비트 부동 소수점
    var normalize = false; // 데이터 정규화 안 함
    var stride = 0;        // 0 = 다음 위치를 가져오기 위해 반복마다 size * sizeof(type) 만큼 앞으로 이동
    var offset = 0;        // 버퍼의 처음부터 시작
    gl.vertexAttribPointer(positionLocation, size, type, normalize, stride, offset);

    // 해상도 설정
    gl.uniform2f(resolutionLocation, gl.canvas.width, gl.canvas.height);

    // 색상 설정
    gl.uniform4fv(colorLocation, color);

    // 사각형 그리기
    var primitiveType = gl.TRIANGLES;
    var offset = 0;
    var count = 6;
    gl.drawArrays(primitiveType, offset, count);
  }
```

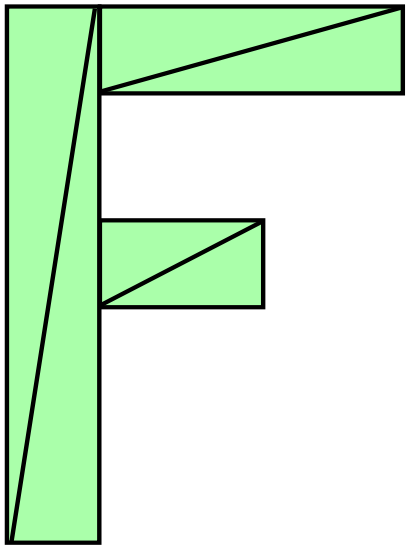
아래 예제에 translation[0] 과 translation[1] 을 업데이트하고 변경할 때마다 drawScene 을 호출하는 슬라이더 두 개를 첨부했습니다. 사각형을 움직이려면 슬라이더를 드래그해보세요.



새 창을 열려면 여기를 클릭

지금까지는 그럭저럭 잘 됐습니다. 하지만 더 복잡한 모양으로 똑같이 하고 싶다고 상상해보세요.

이처럼 삼각형 6개로 구성된 'F'를 그리고 싶다고 가정해봅시다.



우선 현재 코드에 따라 `setRectangle`을 이런 식으로 변경해야 합니다.

```
// 문자 'F'를 정의하는 값들로 버퍼 채우기
function setGeometry(gl, x, y) {
  var width = 100;
  var height = 150;
  var thickness = 30;
  gl.bufferData(
    gl.ARRAY_BUFFER,
    new Float32Array([
      // 왼쪽 열
      x, y,
      x + thickness, y,
      x, y + height,
      x, y + height,
      x + thickness, y,
      x + thickness, y + height,

      // 상단 가로 획
      x + thickness, y,
      x + width, y,
      x + thickness, y + thickness,
      x + thickness, y + thickness,
      x + width, y,
      x + width, y + thickness,

      // 중간 가로 획
      x + thickness, y + thickness * 2,
      x + width * 2 / 3, y + thickness * 2,
      x + thickness, y + thickness * 3,
      x + thickness, y + thickness * 3,
      x + width * 2 / 3, y + thickness * 2,
      x + width * 2 / 3, y + thickness * 3,
    ]),
    gl.STATIC_DRAW
  );
}
```

확장하기에 좋지 않겠다는 걸 눈치채셨을 겁니다. 수백 수천 개의 선으로 이루어진 아주 복잡한 지오메트리를 그리려면 꽤 복잡한 코드를 작성해야 합니다. 또한 자바스크립트는 그릴 때마다 모든 점들을 업데이트해야 하는데요.

더 간단한 방법이 있습니다. 지오메트리를 업로드하고 셰이더에서 평행 이동을 수행하면 됩니다.

여기 새로운 셰이더 입니다.

```
<script id="vertex-shader-2d" type="x-shader/x-vertex">
attribute vec2 a_position;

uniform vec2 u_resolution;
uniform vec2 u_translation;

void main() {
  // 평행 이동 추가
  vec2 position = a_position + u_translation;

  // 사각형을 픽셀에서 0.0에서 1.0사이로 변환
  vec2 zeroToOne = position / u_resolution;
  ...
```

그리고 코드를 조금 재구성할 겁니다. 우선 지오메트리는 한 번만 설정하면 됩니다.

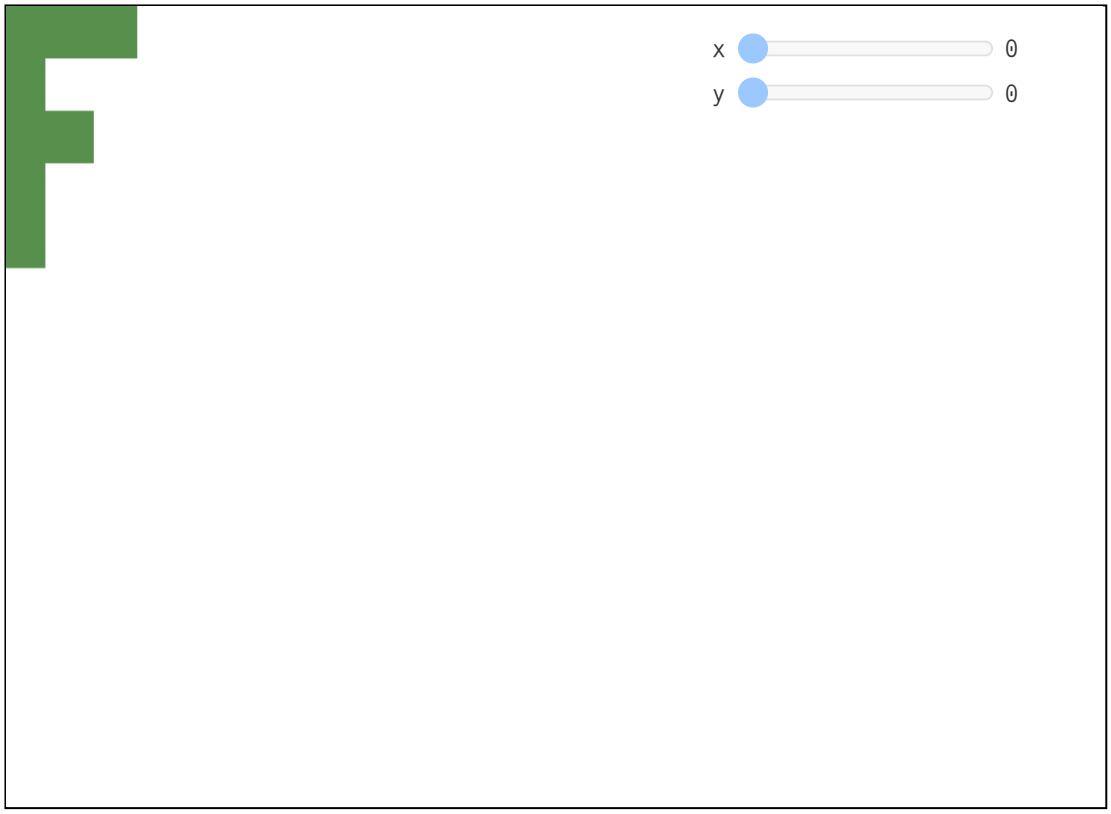
```
// 문자 'F'를 정의하는 값들로 버퍼 채우기
function setGeometry(gl) {
  gl.bufferData(
    gl.ARRAY_BUFFER,
    new Float32Array([
      // 왼쪽 열
      0, 0,
      30, 0,
      0, 150,
      0, 150,
      30, 0,
      30, 150,

      // 상단 가로 획
      30, 0,
      100, 0,
      30, 30,
      30, 30,
      100, 0,
      100, 30,

      // 중간 가로 획
      30, 60,
      67, 60,
      30, 90,
      30, 90,
      67, 60,
      67, 90,
    ]),
    gl.STATIC_DRAW
  );
}
```

그런 다음 평행 이동하여 그리기 전에 `u_translation` 을 업데이트하면 됩니다.

```
...

var translationLocation = gl.getUniformLocation(program, "u_translation");

...

// 위치를 넣을 버퍼 생성
var positionBuffer = gl.createBuffer();
// ARRAY_BUFFER에 바인딩 (ARRAY_BUFFER = positionBuffer)
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
// 버퍼에 지오메트리 데이터 넣기
setGeometry(gl);

...

// 장면 그리기
function drawScene() {

  ...

  // 평행 이동 설정
  gl.uniform2fv(translationLocation, translation);

  // 사각형 그리기
  var primitiveType = gl.TRIANGLES;
  var offset = 0;
  var count = 18;
  gl.drawArrays(primitiveType, offset, count);
}
```

`setGeometry` 가 한 번만 호출된다는 점에 주목하세요. 더 이상 `drawScene` 안에 있지 않습니다.

그리고 여기 해당 예제입니다. 다시 한 번 `translation` 을 업데이트하기 위해 슬라이더를 드래그해보세요.



새 창을 열려면 여기를 클릭

이제 그릴 때 WebGL이 거의 모든 걸 수행하고 있습니다. `translation` 을 설정하고 그려달라고 요청하는 게 우리가 하는 전부죠. 심지어 지오메트리에 수만 개의 점들이 있더라도 주요 코드는 그대로 유지됩니다.

원한다면 모든 점들을 업데이트하는 복잡한 자바스크립트 버전과 비교할 수 있습니다.

너무 뻔한 예제가 아니었기를 바랍니다. 다른 한편으로 이걸 수행하는 훨씬 더 좋은 방법을 다룰 것이므로 계속 읽어주세요. 다음 글에서는 회전으로 넘어가겠습니다.