WebGL 2D 스케일

이 포스트는 WebGL 관련 시리즈에서 이어집니다. 첫 번째는 기초로 시작했고, 이전에는 지오메트리 회전에 관한 것이었습니다.

스케일은 평행 이동만큼이나 쉽습니다.

원하는 스케일로 위치를 곱하면 되는데요. 이전 예제에서 변경된 사항들은 다음과 같습니다.

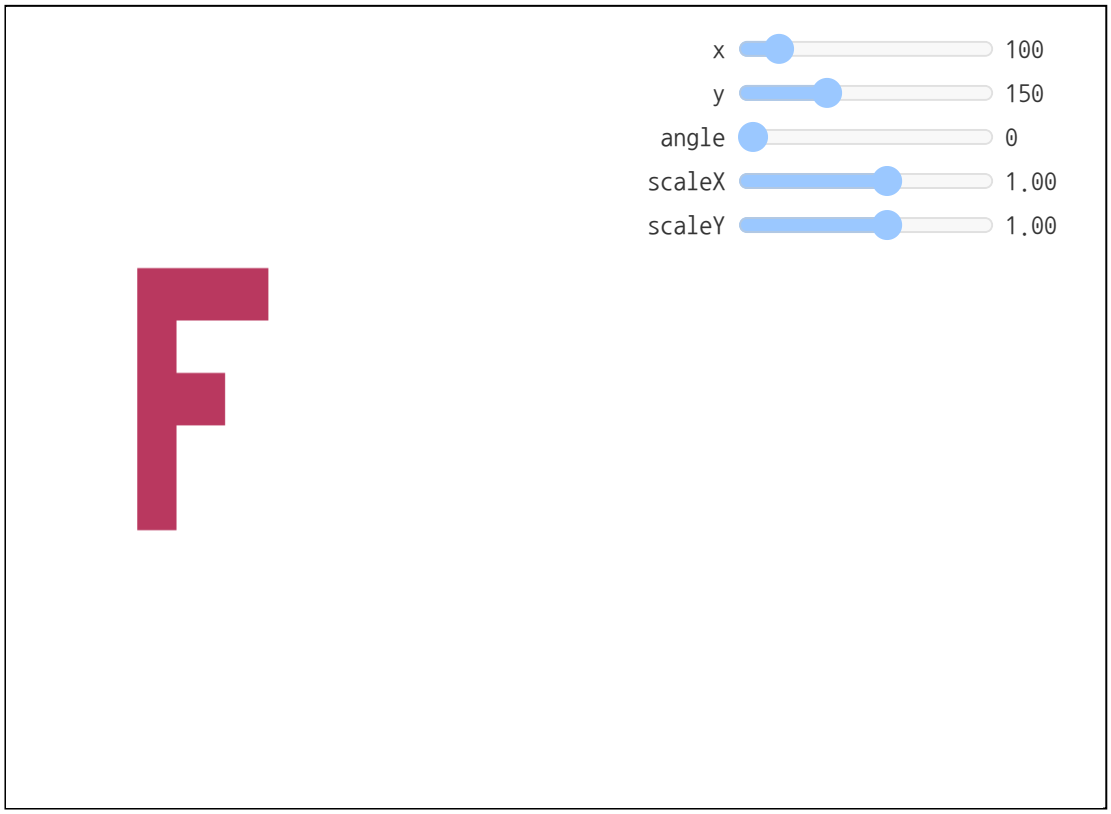
```
<script id="vertex-shader-2d" type="x-shader/x-vertex">
attribute vec2 a_position;

uniform vec2 u_resolution;
uniform vec2 u_translation;
uniform vec2 u_rotation;
uniform vec2 u_scale;

void main() {
  // 위치 스케일링
  vec2 scaledPosition = a_position * u_scale;

  // 위치 회전
  vec2 rotatedPosition = vec2(
    scaledPosition.x * u_rotation.y + scaledPosition.y * u_rotation.x,
    scaledPosition.y * u_rotation.y - scaledPosition.x * u_rotation.x);

  // 평행 이동 추가
  vec2 position = rotatedPosition + u_translation;
```

그리고 그릴 때 스케일을 설정하기 위해 필요한 자바스크립트를 추가합니다.

```
  ...

  var scaleLocation = gl.getUniformLocation(program, "u_scale");

  ...

  var scale = [1, 1];

  ...

  // 장면 그리기
  function drawScene() {

    ...

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

```
    // 회전 설정
    gl.uniform2fv(rotationLocation, rotation);

    // 스케일 설정
    gl.uniform2fv(scaleLocation, scale);

    // 지오메트리 그리기
    var primitiveType = gl.TRIANGLES;
    var offset = 0;
    var count = 18;  // 'F'의 삼각형 6개, 삼각형마다 점 3개
    gl.drawArrays(primitiveType, offset, count);
  }
```

그러면 이제 스케일을 가집니다. 슬라이더를 드래그해보세요.



새 창을 열려면 여기를 클릭

한 가지 주목할만한 점은 음수로 스케일링하면 지오메트리가 뒤집힌다는 겁니다.

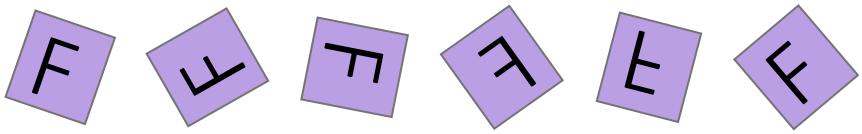
지난 3개의 글이 평행 이동, 회전, 스케일을 이해하는데 도움이 되셨기를 바랍니다.

다음에는 이 3가지 모두를 훨씬 간단하고 더 유용한 형태로 결합하는 행렬을 살펴보겠습니다.

왜 'F'인가요?

처음에 누군가가 'F'를 텍스처로 쓰는 걸 봤는데요. 'F' 자체는 중요하지 않습니다. 중요한 건 어느 방향에서든 오리엔테이션을 알 수 있다는 겁니다. 예를 들어 하트 ❤나 삼각형 △을 사용한다면 그게 수평으로 뒤집혔는지 알 수 없습니다. 원 ◯은 더 안 좋겠죠.

색상이 있는 사각형은 각 모서리를 다른 색으로 할 수 있겠지만 각 모서리가 어떤 것인지 기억하고 있어야 합니다. F의 오리엔테이션은 바로 인지할 수 있습니다.



오리엔테이션을 알 수 있는 어떤 모양이라도 괜찮으며, 저는 처음 이 아이디어를 접했을 때부터 'F'를 사용해왔습니다.