baseStrokeShader()

This API is experimental

Its behavior may change in a future version of p5.js.

Get the shader used when drawing the strokes of shapes.

You can call `baseStrokeShader().modify()` and change any of the following hooks:

Hook	Description
`void beforeVertex`	Called at the start of the vertex shader.
`vec3 getLocalPosition`	Update the position of vertices before transforms are applied. It takes in `vec3 position` and must return a modified version.
`vec3 getWorldPosition`	Update the position of vertices after transforms are applied. It takes in `vec3 position` and pust return a modified version.
`float getStrokeWeight`	Update the stroke weight. It takes in `float weight` and pust return a modified version.
`vec2 getLineCenter`	Update the center of the line. It takes in `vec2 center` and must return a modified version.
`vec2 getLinePosition`	Update the position of each vertex on the edge of the line. It takes in `vec2 position` and must return a modified version.
`vec4 getVertexColor`	Update the color of each vertex. It takes in a `vec4 color` and must return a modified version.
`void afterVertex`	Called at the end of the vertex shader.
`void beforeFragment`	Called at the start of the fragment shader.
`Inputs getPixelInputs`	Update the inputs to the shader. It takes in a struct `Inputs inputs`, which includes: • `vec4 color`, the color of the stroke • `vec2 tangent`, the direction of the stroke in screen space • `vec2 center`, the coordinate of the center of the stroke in screen space p5.js pixels • `vec2 position`, the coordinate of the current pixel in screen space p5.js pixels • `float strokeWeight`, the thickness of the stroke in p5.js pixels
`bool shouldDiscard`	Caps and joins are made by discarded pixels in the fragment shader to carve away unwanted areas. Use this to change this logic. It takes in a `bool willDiscard` and must return a modified version.
`vec4 getFinalColor`	Update the final color after mixing. It takes in a `vec4 color` and must return a modified version.
`void afterFragment`	Called at the end of the fragment shader.

Most of the time, you will need to write your hooks in GLSL ES version 300. If you are using WebGL 1 instead of 2, write your hooks in GLSL ES 100 instead.

Call `baseStrokeShader().inspectHooks()` to see all the possible hooks and their default implementations.

Examples

```
let myShader;

function setup() {
  createCanvas(200, 200, WEBGL);
  myShader = baseStrokeShader().modify({
    'Inputs getPixelInputs': `(Inputs inputs) {
      float opacity = 1.0 - smoothstep(
        0.0,
        15.0,
        length(inputs.position - inputs.center)
      );
      inputs.color *= opacity;
      return inputs;
    }`
  });
}

function draw() {
  background(255);
  shader(myShader);
  strokeWeight(30);
  line(
    -width/3,
    sin(millis()*0.001) * height/4,
    width/3,
    sin(millis()*0.001 + 1) * height/4
  );
}
```

```
let myShader;

function setup() {
  createCanvas(200, 200, WEBGL);
  myShader = baseStrokeShader().modify({
    uniforms: {
      'float time': () => millis()
    },
    declarations: 'vec3 myPosition;',
    'vec3 getWorldPosition': `(vec3 pos) {
      myPosition = pos;
      return pos;
    }`,
    'float getStrokeWeight': `(float w) {
      // Add a somewhat random offset to the weight
      // that varies based on position and time
      float scale = 0.8 + 0.2*sin(10.0 * sin(
        floor(time/250.) +
        myPosition.x*0.01 +
        myPosition.y*0.01
      ));
      return w * scale;
    }`
  });
}

function draw() {
  background(255);
}
```

```
let myShader;

function setup() {
  createCanvas(200, 200, WEBGL);
  myShader = baseStrokeShader().modify({
    'float random': `(vec2 p) {
      vec3 p3  = fract(vec3(p.xyx) * .1031);
      p3 += dot(p3, p3.yzx + 33.33);
      return fract((p3.x + p3.y) * p3.z);
    }`,
    'Inputs getPixelInputs': `(Inputs inputs) {
      // Replace alpha in the color with dithering by
      // randomly setting pixel colors to 0 based on
opacity
      float a = inputs.color.a;
      inputs.color.a = 1.0;
      inputs.color *= random(inputs.position.xy) > a ?
0.0 : 1.0;
      return inputs;
    }`
  });
}

function draw() {
  background(255);
  shader(myShader);
```

Returns

p5.Shader: The stroke shader

This page is generated from the comments in src/webgl/material.js . Please feel free to edit it and submit a pull request!

Related References

copyToContext
Copies the shader from one drawing context to another.

inspectHooks
Logs the hooks available in this shader, and their current implementation.

modify
Returns a new shader, based on the original, but with custom snippets of shader code replacing default behaviour.

setUniform
Sets the shader's uniform (global) variables.