

# baseNormalShader()

This API is experimental

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

Get the shader used by `normalMaterial()`.

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

Hook	Description
<code>void beforeVertex</code>	Called at the start of the vertex shader.
<code>vec3 getLocalPosition</code>	Update the position of vertices before transforms are applied. It takes in <code>vec3 position</code> and must return a modified version.
<code>vec3 getWorldPosition</code>	Update the position of vertices after transforms are applied. It takes in <code>vec3 position</code> and must return a modified version.
<code>vec3 getLocalNormal</code>	Update the normal before transforms are applied. It takes in <code>vec3 normal</code> and must return a modified version.
<code>vec3 getWorldNormal</code>	Update the normal after transforms are applied. It takes in <code>vec3 normal</code> and must return a modified version.
<code>vec2 getUV</code>	Update the texture coordinates. It takes in <code>vec2 uv</code> and must return a modified version.
<code>vec4 getVertexColor</code>	Update the color of each vertex. It takes in a <code>vec4 color</code> and must return a modified version.
<code>void afterVertex</code>	Called at the end of the vertex shader.
<code>void beforeFragment</code>	Called at the start of the fragment shader.
<code>vec4 getFinalColor</code>	Update the final color after mixing. It takes in a <code>vec4 color</code> and must return a modified version.
<code>void afterFragment</code>	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 `baseNormalShader().inspectHooks()` to see all the possible hooks and their default implementations.

## Examples

```
▶ let myShader;

▶ function setup() {
  createCanvas(200, 200, WEBGL);
  myShader = baseNormalShader().modify({
    uniforms: {
      'float time': () => millis()
    },
    'vec3 getWorldPosition': `(vec3 pos) {
      pos.y += 20. * sin(time * 0.001 + pos.x * 0.05);
      return pos;
    }`;
  });
}

function draw() {
  background(255);
  shader(myShader);
  noStroke();
  sphere(50);
}
```

```
▶ let myShader;

▶ function setup() {
  createCanvas(200, 200, WEBGL);
  myShader = baseNormalShader().modify({
    'vec3 getWorldNormal': `(vec3 normal) { return
      abs(normal); }`,
    'vec4 getFinalColor': `(vec4 color) {
      // Map the r, g, and b values of the old normal
      // to new colors
      // instead of just red, green, and blue:
      vec3 newColor =
        color.r * vec3(89.0, 240.0, 232.0) / 255.0 +
        color.g * vec3(240.0, 237.0, 89.0) / 255.0 +
        color.b * vec3(205.0, 55.0, 222.0) / 255.0;
      newColor = newColor / (color.r + color.g +
      color.b);
      return vec4(newColor, 1.0) * color.a;
    }`;
  });
}

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

## Returns

p5.Shader: The `normalMaterial` 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.

