baseMaterialShader()

This API is experimental

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

Get the default shader used with lights, materials, and textures.

You can call `baseMaterialShader().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 must return a modified version.
`vec3 getLocalNormal`	Update the normal before transforms are applied. It takes in `vec3 normal` and must return a modified version.
`vec3 getWorldNormal`	Update the normal after transforms are applied. It takes in `vec3 normal` and must return a modified version.
`vec2 getUV`	Update the texture coordinates. It takes in `vec2 uv` 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 per-pixel inputs of the material. It takes in an `Inputs` struct, which includes: • `vec3 normal`, the direction pointing out of the surface • `vec2 texCoord`, a vector where `x` and `y` are between 0 and 1 describing the spot on a texture the pixel is mapped to, as a fraction of the texture size • `vec3 ambientLight`, the ambient light color on the vertex • `vec4 color`, the base material color of the pixel • `vec3 ambientMaterial`, the color of the pixel when affected by ambient light • `vec3 specularMaterial`, the color of the pixel when reflecting specular highlights • `vec3 emissiveMaterial`, the light color emitted by the pixel • `float shininess`, a number representing how sharp specular reflections should be, from 1 to infinity • `float metalness`, a number representing how mirrorlike the material should be, between 0 and 1 The struct can be modified and returned.
`vec4 combineColors`	Take in a `ColorComponents` struct containing all the different components of light, and combining them into a single final color. The struct contains: • `vec3 baseColor`, the base color of the pixel • `float opacity`, the opacity between 0 and 1 that it should be drawn at • `vec3 ambientColor`, the color of the pixel when affected by ambient light • `vec3 specularColor`, the color of the pixel when affected by specular reflections • `vec3 diffuse`, the amount of diffused light hitting the pixel • `vec3 ambient`, the amount of ambient light hitting the pixel • `vec3 specular`, the amount of specular reflection hitting the pixel • `vec3 emissive`, the amount of light emitted by the pixel
`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 `baseMaterialShader().inspectHooks()` to see all the possible hooks and their default implementations.

Examples

```
let myShader;

function setup() {
  createCanvas(200, 200, WEBGL);
  myShader = baseMaterialShader().modify({
    uniforms: {
      'float time': () => millis()
    },
    'vec3 getWorldPosition': `(vec3 pos) {
      pos.y += 20.0 * sin(time * 0.001 + pos.x * 0.05);
      return pos;
    }`
  });
}

function draw() {
  background(255);
  shader(myShader);
  lights();
  noStroke();
  fill('red');
  sphere(50);
}
```

```
let myShader;

function setup() {
  createCanvas(200, 200, WEBGL);
  myShader = baseMaterialShader().modify({
    declarations: 'vec3 myNormal;',
    'Inputs getPixelInputs': `(Inputs inputs) {
      myNormal = inputs.normal;
      return inputs;
    }`,
    'vec4 getFinalColor': `(vec4 color) {
      return mix(
        vec4(1.0, 1.0, 1.0, 1.0),
        color,
        abs(dot(myNormal, vec3(0.0, 0.0, 1.0)))
      );
    }`
  });
}

function draw() {
  background(255);
  rotateY(millis() * 0.001);
  shader(myShader);
  lights();
  noStroke();
  fill('red');
  torus(30);
}
```

```
let myShader;
let environment;

function preload() {
  environment =
loadImage('/assets/outdoor_spheremap.jpg');
}

function setup() {
  createCanvas(200, 200, WEBGL);
  myShader = baseMaterialShader().modify({
    'Inputs getPixelInputs': `(Inputs inputs) {
      float factor =
        sin(
          inputs.texCoord.x * ${TWO_PI} +
          inputs.texCoord.y * ${TWO_PI}
        ) * 0.4 + 0.5;
      inputs.shininess = mix(1., 100., factor);
      inputs.metalness = factor;
      return inputs;
    }`
  });
}

function draw() {
  panorama(environment);
  ambientLight(100);
  imageLight(environment);
  rotateY(millis() * 0.001);
```

```
let myShader;

function setup() {
  createCanvas(200, 200, WEBGL);
  myShader = baseMaterialShader().modify({
    'Inputs getPixelInputs': `(Inputs inputs) {
      vec3 newNormal = inputs.normal;
      // Simple bump mapping: adjust the normal based
on position
      newNormal.x += 0.2 * sin(
          sin(
            inputs.texCoord.y * ${TWO_PI} * 10.0 +
            inputs.texCoord.x * ${TWO_PI} * 25.0
          )
        );
      newNormal.y += 0.2 * sin(
        sin(
            inputs.texCoord.x * ${TWO_PI} * 10.0 +
            inputs.texCoord.y * ${TWO_PI} * 25.0
          )
        );
      inputs.normal = normalize(newNormal);
      return inputs;
```

Returns

p5.Shader: The material shader

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.