

createShader()

Creates a new `p5.Shader` object.

Shaders are programs that run on the graphics processing unit (GPU). They can process many pixels at the same time, making them fast for many graphics tasks. They're written in a language called `GLSL` and run along with the rest of the code in a sketch.

Once the `p5.Shader` object is created, it can be used with the `shader()` function, as in `shader(myShader)`. A shader program consists of two parts, a vertex shader and a fragment shader. The vertex shader affects where 3D geometry is drawn on the screen and the fragment shader affects color.

The first parameter, `vertSrc`, sets the vertex shader. It's a string that contains the vertex shader program written in GLSL.

The second parameter, `fragSrc`, sets the fragment shader. It's a string that contains the fragment shader program written in GLSL.

A shader can optionally describe `hooks`, which are functions in GLSL that users may choose to provide to customize the behavior of the shader using the `modify()` method of `p5.Shader`. These are added by describing the hooks in a third parameter, `options`, and referencing the hooks in your `vertSrc` or `fragSrc`. Hooks for the vertex or fragment shader are described under the `vertex` and `fragment` keys of `options`. Each one is an object, where each key is the type and name of a hook function, and each value is a string with the parameter list and default implementation of the hook. For example, to let users optionally run code at the start of the vertex shader, the `options` object could include:

```
{
  vertex: {
    'void beforeVertex': '() {}'
  }
}
```

Then, in your vertex shader source, you can run a hook by calling a function with the same name prefixed by `HOOK_`. If you want to check if the default hook has been replaced, maybe to avoid extra overhead, you can check if the same name prefixed by `AUGMENTED_HOOK_` has been defined:

```
void main() {
  // In most cases, just calling the hook is fine:
  HOOK_beforeVertex();

  // Alternatively, for more efficiency:
  #ifdef AUGMENTED_HOOK_beforeVertex
  HOOK_beforeVertex();
  #endif

  // Add the rest of your shader code here!
}
```

Note: Only filter shaders can be used in 2D mode. All shaders can be used in WebGL mode.

Examples

```
// Note: A "uniform" is a global variable within a shader program.

// Create a string with the vertex shader program.
// The vertex shader is called for each vertex.
let vertSrc = `
precision highp float;
uniform mat4 uModelViewMatrix;
uniform mat4 uProjectionMatrix;
attribute vec3 aPosition;
attribute vec2 aTexCoord;
varying vec2 vTexCoord;

void main() {
  vTexCoord = aTexCoord;
  vec4 positionVec4 = vec4(aPosition, 1.0);
  gl_Position = uProjectionMatrix * uModelviewMatrix *
positionVec4;
}
`;

// Create a string with the fragment shader program.
// The fragment shader is called for each pixel.
let fragSrc = `
precision highp float;

void main() {
```

```
// Note: A "uniform" is a global variable within a shader program.

// Create a string with the vertex shader program.
// The vertex shader is called for each vertex.
let vertSrc = `
precision highp float;
uniform mat4 uModelViewMatrix;
uniform mat4 uProjectionMatrix;
attribute vec3 aPosition;
attribute vec2 aTexCoord;
varying vec2 vTexCoord;

void main() {
  vTexCoord = aTexCoord;
  vec4 positionVec4 = vec4(aPosition, 1.0);
  gl_Position = uProjectionMatrix * uModelViewMatrix *
positionVec4;
}
`;

// Create a string with the fragment shader program.
// The fragment shader is called for each pixel.
let fragSrc = `
precision highp float;
uniform vec2 p;
```

Syntax

```
createShader(vertSrc, fragSrc, [options])
```

Parameters

`vertSrc`: String: source code for the vertex shader.
`fragSrc`: String: source code for the fragment shader.
`options`: Object: An optional object describing how this shader can be augmented with hooks. It can include:
`vertex`: An object describing the available vertex shader hooks.
`fragment`: An object describing the available fragment shader hooks.

Returns

`p5.Shader`: new shader object created from the vertex and fragment shaders.

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.

p5.js	Resources	Information	Socials
	Reference Tutorials Examples Contribute Community About Start Coding Donate	Download Contact Copyright Privacy Policy Terms of Use	GitHub ↗ Instagram ↗ X ↗ YouTube ↗ Discord ↗ Forum ↗

Donate Today! Support p5.js and the Processing Foundation.

