setAttributes()

Set attributes for the WebGL Drawing context. This is a way of adjusting how the WebGL renderer works to fine-tune the display and performance.

Note that this will reinitialize the drawing context if called after the WebGL canvas is made.

If an object is passed as the parameter, all attributes not declared in the object will be set to defaults.

The available attributes are:
alpha - indicates if the canvas contains an alpha buffer default is true

depth - indicates whether the drawing buffer has a depth buffer of at least 16 bits - default is true

stencil - indicates whether the drawing buffer has a stencil buffer of at least 8 bits

antialias - indicates whether or not to perform anti-aliasing default is false (true in Safari)

premultipliedAlpha - indicates that the page compositor will assume the drawing buffer contains colors with pre-multiplied alpha default is true

preserveDrawingBuffer - if true the buffers will not be cleared and and will preserve their values until cleared or overwritten by author (note that p5 clears automatically on draw loop) default is true

perPixelLighting - if true, per-pixel lighting will be used in the lighting shader otherwise per-vertex lighting is used. default is true.

version - either 1 or 2, to specify which WebGL version to ask for. By default, WebGL 2 will be requested. If WebGL2 is not available, it will fall back to WebGL 1. You can check what version is used with by looking at the global `webglVersion` property.

Examples

```
function setup() {
  createCanvas(100, 100, WEBGL);
}

function draw() {
  background(255);
  push();
  rotateZ(frameCount * 0.02);
  rotateX(frameCount * 0.02);
  rotateY(frameCount * 0.02);
  fill(0, 0, 0);
  box(50);
  pop();
}
```

```
<br>
Now with the antialias attribute set to true.
<br>


function setup() {
  setAttributes('antialias', true);
  createCanvas(100, 100, WEBGL);
}

function draw() {
  background(255);
  push();
  rotateZ(frameCount * 0.02);
  rotateX(frameCount * 0.02);
  rotateY(frameCount * 0.02);
  fill(0, 0, 0);
  box(50);
  pop();
}
```

```
// press the mouse button to disable perPixelLighting
function setup() {
  createCanvas(100, 100, WEBGL);
  noStroke();
  fill(255);
}

let lights = [
  { c: '#f00', t: 1.12, p: 1.91, r: 0.2 },
  { c: '#0f0', t: 1.21, p: 1.31, r: 0.2 },
  { c: '#00f', t: 1.37, p: 1.57, r: 0.2 },
  { c: '#ff0', t: 1.12, p: 1.91, r: 0.7 },
  { c: '#0ff', t: 1.21, p: 1.31, r: 0.7 },
  { c: '#f0f', t: 1.37, p: 1.57, r: 0.7 }
];

function draw() {
  let t = millis() / 1000 + 1000;
  background(0);
  directionalLight(color('#222'), 1, 1, 1);

  for (let i = 0; i < lights.length; i++) {
    let light = lights[i];
    pointLight(
      color(light.c),
      p5.Vector.fromAngles(t * light.t, t * light.p, width *
light.r)
```

Syntax

```
setAttributes(key, value)
```

```
setAttributes(obj)
```

Parameters

key	String: Name of attribute
value	Boolean: New value of named attribute
obj	Object: object with key-value pairs

This page is generated from the comments in src/webgl/p5.RendererGL.js . Please feel free to edit it and submit a pull request!

Related References

createFramebuffer
Creates a new p5.Framebuffer object with the same WebGL context as the graphics buffer.

remove
Removes the graphics buffer from the web page.

reset
Resets the graphics buffer's transformations and lighting.

blendMode
Sets the way colors blend when added to the canvas.