

buildGeometry()

Creates a custom `p5.Geometry` object from simpler 3D shapes.

`buildGeometry()` helps with creating complex 3D shapes from simpler ones such as `sphere()`. It can help to make sketches more performant. For example, if a complex 3D shape doesn't change while a sketch runs, then it can be created with `buildGeometry()`. Creating a `p5.Geometry` object once and then drawing it will run faster than repeatedly drawing the individual pieces.

The parameter, `callback`, is a function with the drawing instructions for the new `p5.Geometry` object. It will be called once to create the new 3D shape.

See `beginGeometry()` and `endGeometry()` for another way to build 3D shapes.

Note: `buildGeometry()` can only be used in WebGL mode.

Examples



// Click and drag the mouse to view the scene from different angles.

```
let shape;

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

  // Create the p5.Geometry object.
  shape = buildGeometry(createShape);

  describe('A white cone drawn on a gray background.');
}

function draw() {
  background(50);

  // Enable orbiting with the mouse.
  orbitControl();

  // Turn on the lights.
  lights();

  // Style the p5.Geometry object.
  noStroke();

  // Draw the p5.Geometry object.
  model(shape);
}

// Create p5.Geometry object from a single cone.
function createShape() {
```



// Click and drag the mouse to view the scene from different angles.

```
let shape;

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

  // Create the arrow.
  shape = buildGeometry(createArrow);

  describe('A white arrow drawn on a gray background.');
}

function draw() {
  background(50);

  // Enable orbiting with the mouse.
  orbitControl();

  // Turn on the lights.
  lights();

  // Style the arrow.
  noStroke();

  // Draw the arrow.
  model(shape);
}

function createArrow() {
  // Add shapes to the p5.Geometry object.
  push();
  rotateX(PI);
```



// Click and drag the mouse to view the scene from different angles.

```
let button;
let particles;

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

  // Create the p5.Geometry object.
  shape = buildGeometry(createArrow);

  describe('Two white arrows drawn on a gray background. The arrow on the right rotates slowly.');
}

function draw() {
  background(50);

  // Enable orbiting with the mouse.
  orbitControl();

  // Turn on the lights.
  lights();

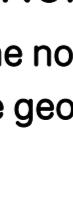
  // Style the arrows.
  noStroke();

  // Draw the p5.Geometry object.
  model(shape);

  // Translate and rotate the coordinate system.
  translate(30, 0, 0);
```

Syntax

```
buildGeometry(callback)
```



Parameters

`callback` Function: function that draws the shape.

This page is generated from the comments in `src/webgl/3d_primitives.js`. Please feel free to edit it and submit a pull request!

Related References

`calculateBoundingBox`

Calculates the position and size of the smallest box that contains the geometry.

`clearColors`

Removes the geometry's internal colors.

`computeFaces`

Computes the geometry's faces using its vertices.

`computeNormals`

Calculates the normal vector for each vertex on the geometry.

`p5.js`

`Resources`

[Reference](#)

[Tutorials](#)

[Examples](#)

[Contribute](#)

[Community](#)

[About](#)

[Start Coding](#)

[Donate](#)

`Information`

[Download](#)

[Contact](#)

[Copyright](#)

[Privacy Policy](#)

[Terms of Use](#)

`Socials`

[GitHub ↗](#)

[Instagram ↗](#)

[X ↗](#)

[YouTube ↗](#)

[Discord ↗](#)

[Forum ↗](#)

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

