scale()

Scales the coordinate system.

By default, shapes are drawn at their original scale. A rectangle that's 50 pixels wide appears to take up half the width of a 100 pixel-wide canvas. The `scale()` function can shrink or stretch the coordinate system so that shapes appear at different sizes. There are two ways to call `scale()` with parameters that set the scale factor(s).

The first way to call `scale()` uses numbers to set the amount of scaling. The first parameter, `s`, sets the amount to scale each axis. For example, calling `scale(2)` stretches the x-, y-, and z-axes by a factor of 2. The next two parameters, `y` and `z`, are optional. They set the amount to scale the y- and z-axes. For example, calling `scale(2, 0.5, 1)` stretches the x-axis by a factor of 2, shrinks the y-axis by a factor of 0.5, and leaves the z-axis unchanged.

The second way to call `scale()` uses a p5.Vector object to set the scale factors. For example, calling `scale(myVector)` uses the x-, y-, and z-components of `myVector` to set the amount of scaling along the x-, y-, and z-axes. Doing so is the same as calling `scale(myVector.x, myVector.y, myVector.z)`.

By default, transformations accumulate. For example, calling `scale(1)` twice has the same effect as calling `scale(2)` once. The push() and pop() functions can be used to isolate transformations within distinct drawing groups.

Note: Transformations are reset at the beginning of the draw loop. Calling `scale(2)` inside the draw() function won't cause shapes to grow continuously.

Examples

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

  describe(
    'Two white squares on a gray background. The larger square appears at the top-center. The smaller square appears at the top-left.'
  );
}

function draw() {
  background(200);

  // Draw a square at (30, 20).
  square(30, 20, 40);

  // Scale the coordinate system by a factor of 0.5.
  scale(0.5);

  // Draw a square at (30, 20).
  // It appears at (15, 10) after scaling.
  square(30, 20, 40);
}
```

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

  describe('A rectangle and a square drawn in white on a gray background.');
}

function draw() {
  background(200);

  // Draw a square at (30, 20).
  square(30, 20, 40);

  // Scale the coordinate system by factors of
  // 0.5 along the x-axis and
  // 1.3 along the y-axis.
  scale(0.5, 1.3);

  // Draw a square at (30, 20).
  // It appears as a rectangle at (15, 26) after scaling.
  square(30, 20, 40);
}
```

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

  describe('A rectangle and a square drawn in white on a gray background.');
}

function draw() {
  background(200);

  // Draw a square at (30, 20).
  square(30, 20, 40);

  // Create a p5.Vector object.
  let v = createVector(0.5, 1.3);

  // Scale the coordinate system by factors of
  // 0.5 along the x-axis and
  // 1.3 along the y-axis.
  scale(v);

  // Draw a square at (30, 20).
  // It appears as a rectangle at (15, 26) after scaling.
  square(30, 20, 40);
}
```

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

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

  describe(
    'A red box and a blue box drawn on a gray background. The red box appears embedded in the blue box.'
  );
}

function draw() {
  background(200);

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

  // Turn on the lights.
  lights();

  // Style the spheres.
  noStroke();

  // Draw the red sphere.
  fill('red');
  box();

  // Scale the coordinate system by factors of
  // 0.5 along the x-axis and
  // 1.3 along the y-axis and
  // 2 along the z-axis.
```

Syntax

```
scale(s, [y], [z])
```

```
scale(scales)
```

Parameters

s	Number|p5.Vector|Number[]: amount to scale along the positive x-axis.
y	Number: amount to scale along the positive y-axis. Defaults to `s`.
z	Number: amount to scale along the positive z-axis. Defaults to `y`.
scales	p5.Vector|Number[]: vector whose components should be used to scale.

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

Related References

applyMatrix
Applies a transformation matrix to the coordinate system.

resetMatrix
Clears all transformations applied to the coordinate system.

rotate
Rotates the coordinate system.

rotateX
Rotates the coordinate system about the x-axis in WebGL mode.