

ellipsoid()

Draws an ellipsoid.

An ellipsoid is a 3D shape with triangular faces that connect to form a round surface. Ellipsoids with few faces look like crystals. Ellipsoids with many faces have smooth surfaces and look like eggs. `ellipsoid()` defines a shape by its radii. This is different from `ellipse()` which uses diameters (width and height).

The first parameter, `radiusX`, is optional. If a `Number` is passed, as in `ellipsoid(20)`, it sets the radius of the ellipsoid along the x-axis. By default, `radiusX` is 50.

The second parameter, `radiusY`, is also optional. If a `Number` is passed, as in `ellipsoid(20, 30)`, it sets the ellipsoid's radius along the y-axis. By default, `radiusY` is set to the ellipsoid's `radiusX`.

The third parameter, `radiusZ`, is also optional. If a `Number` is passed, as in `ellipsoid(20, 30, 40)`, it sets the ellipsoid's radius along the z-axis. By default, `radiusZ` is set to the ellipsoid's `radiusY`.

The fourth parameter, `detailX`, is also optional. If a `Number` is passed, as in `ellipsoid(20, 30, 40, 5)`, it sets the number of triangle subdivisions to use along the x-axis. All 3D shapes are made by connecting triangles to form their surfaces. By default, `detailX` is 24.

The fifth parameter, `detailY`, is also optional. If a `Number` is passed, as in `ellipsoid(20, 30, 40, 5, 7)`, it sets the number of triangle subdivisions to use along the y-axis. All 3D shapes are made by connecting triangles to form their surfaces. By default, `detailY` is 16.

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

Examples

```

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

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

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

function draw() {
  background(200);

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

  // Draw the ellipsoid.
  // Set its radiusX to 30.
  ellipsoid(30);
}

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

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

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

function draw() {
  background(200);

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

  // Draw the ellipsoid.
  // Set its radiusX to 30.
  // Set its radiusY to 40.
  ellipsoid(30, 40);
}

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

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

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

function draw() {
  background(200);

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

  // Draw the ellipsoid.
  // Set its radiusX to 30.
  // Set its radiusY to 40.
  // Set its radiusZ to 50.
  ellipsoid(30, 40, 50);
}

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

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

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

function draw() {
  background(200);

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

  // Draw the ellipsoid.
  // Set its radiusX to 30.
  // Set its radiusY to 40.
  // Set its radiusZ to 50.
  // Set its detailX to 4.
  // Set its detailY to 3.
  ellipsoid(30, 40, 50, 4, 3);
}

```

Syntax

`ellipsoid([radiusX], [radiusY], [radiusZ], [detailX], [detailY])`

Parameters

`radiusX` Number: radius of the ellipsoid along the x-axis. Defaults to 50.
`radiusY` Number: radius of the ellipsoid along the y-axis. Defaults to `radiusX`.
`radiusZ` Number: radius of the ellipsoid along the z-axis. Defaults to 40.
`detailX` Integer: number of triangle subdivisions along the x-axis. Defaults to 24.
`detailY` Integer: number of triangle subdivisions along the y-axis. Defaults to 16.

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 bounding box that contains the size of 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.

`detailX`

Number: detailX

`detailY`

Number: detailY

`detailZ`

Number: detailZ

`detailW`

Number: detailW

`detailH`

Number: detailH

`detailA`

Number: detailA

`detailB`

Number: detailB

`detailC`

Number: detailC

`detailD`

Number: detailD

`detailE`

Number: detailE

`detailF`

Number: detailF

`detailG`

Number: detailG

`detailH`

Number: detailH

`detailI`

Number: detailI

`detailJ`

Number: detailJ

`detailK`

Number: detailK

`detailL`

Number: detailL

`detailM`

Number: detailM

`detailN`

Number: detailN

`detailO`

Number: detailO

`detailP`

Number: detailP

`detailQ`

Number: detailQ

`detailR`

Number: detailR

`detailS`

Number: detailS

`detailT`

Number: detailT

`detailU`

Number: detailU

`detailV`

Number: detailV

`detailW`

Number: detailW

`detailX`

Number: detailX

`detailY`

Number: detailY

`detailZ`

Number: detailZ

`detailA`

Number: detailA

`detailB`

Number: detailB

`detailC`

Number: detailC

`detailD`

Number: detailD

`detailE`

Number: detailE

`detailF`

Number: detailF

`detailG`

Number: detailG

`detailH`

Number: detailH

`detailI`

Number: detailI

`detailJ`

Number: detailJ

`detailK`

Number: detailK

`detailL`

Number: detailL

`detailM`

Number: detailM

`detailN`

Number: detailN

`detailO`

Number: detailO

`detailP`

Number: detailP

`detailQ`

Number: detailQ

`detailR`

Number: detailR

`detailS`

Number: detailS

`detailT`

Number: detailT

`detailU`

Number: detailU

`detailV`

Number: detailV

`detailW`

Number: detailW

`detailX`

Number: detailX

`detailY`

Number: detailY

`detailZ`

Number: detailZ

`detailA`

Number: detailA

`detailB`

Number: detailB

`detailC`

Number: detailC

`detailD`

Number: detailD

`detailE`

Number: detailE

`detailF`

Number: detailF

`detailG`

Number: detailG

`detailH`

Number: detailH

</