computeFaces()

Computes the geometry's faces using its vertices.

All 3D shapes are made by connecting sets of points called *vertices*. A geometry's surface is formed by connecting vertices to form triangles that are stitched together. Each triangular patch on the geometry's surface is called a *face*. `myGeometry.computeFaces()` performs the math needed to define each face based on the distances between vertices.

The geometry's vertices are stored as p5.Vector objects in the myGeometry.vertices array. The geometry's first vertex is the p5.Vector object at `myGeometry.vertices[0]`, its second vertex is `myGeometry.vertices[1]`, its third vertex is `myGeometry.vertices[2]`, and so on.

Calling `myGeometry.computeFaces()` fills the myGeometry.faces array with three-element arrays that list the vertices that form each face. For example, a geometry made from a rectangle has two faces because a rectangle is made by joining two triangles. myGeometry.faces for a rectangle would be the two-dimensional array `[[0, 1, 2], [2, 1, 3]]`. The first face, `myGeometry.faces[0]`, is the array `[0, 1, 2]` because it's formed by connecting `myGeometry.vertices[0]`, `myGeometry.vertices[1]`,and `myGeometry.vertices[2]`. The second face, `myGeometry.faces[1]`, is the array `[2, 1, 3]` because it's formed by connecting `myGeometry.vertices[2]`, `myGeometry.vertices[1]`, and `myGeometry.vertices[3]`.

Note: `myGeometry.computeFaces()` only works when geometries have four or more vertices.

Examples

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

let myGeometry;

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

  // Create a p5.Geometry object.
  myGeometry = new p5.Geometry();

  // Create p5.Vector objects to position the vertices.
  let v0 = createVector(-40, 0, 0);
  let v1 = createVector(0, -40, 0);
  let v2 = createVector(0, 40, 0);
  let v3 = createVector(40, 0, 0);

  // Add the vertices to myGeometry's vertices array.
  myGeometry.vertices.push(v0, v1, v2, v3);

  // Compute myGeometry's faces array.
  myGeometry.computeFaces();

  describe('A red square drawn on a gray background.');
```

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

let myGeometry;

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

  // Create a p5.Geometry object using a callback function.
  myGeometry = new p5.Geometry(1, 1, createShape);

  describe('A red square drawn on a gray background.');
}

function draw() {
  background(200);

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

  // Turn on the lights.
  lights();

  // Style the shape.
  noStroke();
  fill(255, 0, 0);

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

function createShape() {
  // Create p5.Vector objects to position the vertices.
```

This page is generated from the comments in src/webgl/p5.Geometry.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.