The centroid, $\bar{r}$, of a solid is defined as,

$$\bar{r} = \frac{1}{Volume} \iiint_V r \; dV$$

where, r is a position vector. The centroid has a few very interesting properties such as: 1. For a homogeneous solid the *center of mass* is located at the centroid 2. For any *linear function*, the average of the function inside the solid can be calculated by simply evaluating the function at the centroid point.

The calculation of the centroid of a solid can be done by using the divergence theorem to reduce the volume integral to area integral.

$$\bar{r} = \frac{1}{Volume} \iiint_V (x, y, z) \; dV = \frac{1}{Volume} \iiint_V \left(x\frac{\partial x}{\partial x}, y\frac{\partial y}{\partial y}, z\frac{\partial z}{\partial z} \right) \; dV$$

$$\bar{r} = \frac{1}{Volume} \frac{1}{2} \iiint_V \left(\frac{\partial x^2}{\partial x}, \frac{\partial y^2}{\partial y}, \frac{\partial z^2}{\partial z} \right) \; dV$$

$$\bar{x} = \frac{1}{Volume} \frac{1}{2} \iint_A x^2 \; n_x \; dA$$

$$\bar{y} = \frac{1}{Volume} \frac{1}{2} \iint_A y^2 \; n_y \; dA$$

$$\bar{z} = \frac{1}{Volume} \frac{1}{2} \iint_A z^2 \; n_z \; dA$$

where, $\bar{x}$, $\bar{y}$ and $\bar{z}$ are the components of the centroid in x , y and z directions respectively. The unit normal to the enclosing surface, $\hat{n} = (n_x, n_y, n_z)$, points outward of the solid. Thus, we can evaluate the centroid by calculation of second moment of area over the enclosing surface of the solid.

If we divide the enclosing surface into triangles, then for each of the triangle the normal remains constant and therefore can be moved out of the integral. Let us consider the integration of x -coordinate for an individual triangle in space.

$$I_x = \iint_A x^2 \; n_x \; dA$$

Since n_x is constant the integration becomes,

$$I_x = n_x \iint_A x^2 \; dA$$

The area integration needs to be carried out in the plane of the triangular element. Let us consider an arbitrary triangle with vertex 0, 1, 2. Let us also consider a new 2-dimensional $u - v$ co-ordinate system with origin at the vertex 0 of the triangle, the u-axis aligned to base of the triangle pointing towards vertex 1. The v-axis is perpendicular to the u-axis in the plane of the triangle. The integration therefore becomes,

$$I_x = n_x \iint_A x^2 \, du \, dv$$

The integration can be vastly simplified by using natural co-ordinate system, $\xi - \eta$, such that the ξ -axis is aligned to u-axis and scaled such that the vertex 0 is at $\xi = 0$ and vertex 1 is at $\xi = 1$. The η -axis is aligned to triangle edge 0-2 and scaled such that the vertex 0 is at $\eta = 0$ and vertex 2 is at $\eta = 1$. The region of integration in the $\xi - \eta$ co-ordinate system therefore simplifies to a right-angled triangle, which is easy to integrate. The integrand, $x^2 \, du \, dv$, needs to be transformed for the $\xi - \eta$ co-ordinate system. The linear transformation for x can be obtained as,

$$x = c_0 + c_1\xi + c_2\eta$$

Now, substituting the constraints on the co-ordinate system, at $\xi = 0, \eta = 0$; $x = x_0$, we get,

$$x_0 = c_0 + c_1 \times 0 + c_2 \times 0$$
$$c_0 = x_0$$

Substituting, at $\xi = 1, \eta = 0$; $x = x_1$, we get,

$$x_1 = x_0 + c_1 \times 1 + c_2 \times 0$$
$$c_1 = x_1 - x_0$$

Substituting, at $\xi = 0, \eta = 1$; $x = x_2$, we get,

$$x_2 = x_0 + c_1 \times 0 + c_2 \times 1$$
$$c_2 = x_2 - x_0$$

Therefore in the new co-ordinate system,

$$x = x_0 + (x_1 - x_0)\,\xi + (x_2 - x_0)\,\eta$$

similarly,

$$y = y_0 + (y_1 - y_0)\,\xi + (y_2 - y_0)\,\eta$$

and,

$$z = z_0 + (z_1 - z_0)\,\xi + (z_2 - z_0)\,\eta$$

The integration now becomes,

$$I_x = n_{xA} \left[x_0 + (x_1 - x_0)\,\xi + (x_2 - x_0)\,\eta \right]^2 \; du\, dv$$

Also elemental area,

$$du\, dv = \frac{1}{\left| \partial\left(\xi, \eta\right) / \partial\left(u, v\right) \right|} d\xi\, d\eta$$

where, $\left| \partial\left(\xi, \eta\right) / \partial\left(u, v\right) \right|$ is the determinant of the Jacobian. In this linear transformation it is just the ratio of area of the triangle in the $\xi - \eta$ co-ordinate system to area of triangle in $u - v$ co-ordinate system. Since the area of the triangle in $\xi - \eta$ co-ordinate system is 0.5, the scaling factor becomes,

$$\left| \vec{A}_T \right| = \frac{1}{\left| \partial\left(\xi, \eta\right) / \partial\left(u, v\right) \right|} = \left| \overrightarrow{0-1} \times \overrightarrow{0-2} \right|$$

where $\vec{A}_T = \overrightarrow{0-1} \times \overrightarrow{0-2}$ is the cross-product of the vectors made from edges 0-1 and 0-2. Therefore $\left| \vec{A}_T \right|$ is twice the area of the triangle. The integration therefore becomes,

$$I_x = \left| \vec{A}_T \right| \; n_{xA} \left[x_0 + (x_1 - x_0)\,\xi + (x_2 - x_0)\,\eta \right]^2 \; d\xi\, d\eta$$

with limits of the integration,

$$I_x = \left| \vec{A}_T \right| \; n_x \int_{\eta=0}^{\eta=1} \int_{\xi=0}^{\xi=1-\eta} \left[x_0 + (x_1 - x_0)\,\xi + (x_2 - x_0)\,\eta \right]^2 \; d\xi \; d\eta$$

Using Maxima symbolic manipulator we can simplify this as,

$$I_x = \left| \vec{A}_T \right| \; n_x \left(\frac{x_0^2 + x_1^2 + x_2^2 + x_0 x_1 + x_0 x_2 + x_1 x_2}{12} \right)$$

Therefore the formula for centroid of a solid with N enclosing triangles becomes,

$$\bar{x} = \frac{1}{Volume} \frac{1}{2} \sum_{i=0}^{N-1} \left| \vec{A}_{Ti} \right| \; n_{xi} \left(\frac{x_0^2 + x_1^2 + x_2^2 + x_0 x_1 + x_0 x_2 + x_1 x_2}{12} \right)_i$$

$$\bar{x} = \frac{1}{24} \frac{1}{Volume} \sum_{i=0}^{N-1} \left| \vec{A}_{Ti} \right| \; n_{xi} \left(x_0^2 + x_1^2 + x_2^2 + x_0 x_1 + x_0 x_2 + x_1 x_2 \right)_i$$

Similarly,

$$\bar{y} = \frac{1}{24} \frac{1}{Volume} \sum_{i=0}^{N-1} \left| \vec{A}_{Ti} \right| \; n_{yi} \left(y_0^2 + y_1^2 + y_2^2 + y_0 y_1 + y_0 y_2 + y_1 y_2 \right)_i$$

and

$$\bar{z} = \frac{1}{24}\frac{1}{Volume}\sum_{i=0}^{N-1}\left|\vec{A}_{Ti}\right| n_{zi}\left(z_0^2 + z_1^2 + z_2^2 + z_0z_1 + z_0z_2 + z_1z_2\right)_i$$

The above expressions can be further simplified by realizing that $\left|\vec{A}_T\right| (n_x, n_y, n_z) = \overrightarrow{0-1} \times \overrightarrow{0-2}$, which can be calculated only once and the x- y- and z-components can be used in respective equations. It is advisable to translate the solid close to the origin for the numerical calculations so that the numerical errors due to squaring of large numbers is reduced. The actual centroid can be obtained by translating back by the same amount.

A sample program written in Java to implement the algorithm is given below:

```
/**
 * @author Sourabh Bhat (heySourabh@gmail.com)
 * Calculates the centroid point of a solid formed by a set of triangles.
 * The triangles must have points either all-clockwise or
 * all-anti-clockwise, looking from outside of the solid.
 *
 * @param triangles array of triangles
 * @return centroid point of solid region enclosed by the set of triangles
 */
public static Point centroid(TriGeom[] triangles) {
    Point translateBy = boundingBox(triangles).midPoint();
    TriGeom[] newTriangles = translateTriangles(triangles, -translateBy.x,
-translateBy.y, -translateBy.z);

    double vol = signedVolume(newTriangles);
    Vector centroidPos = new Vector(0, 0, 0);

    for (TriGeom t : newTriangles) {
        centroidPos = centroidPos.add(scaledCentroidUnder(t));
    }
    centroidPos = centroidPos.mult(1.0 / vol / 24.0);
    centroidPos = centroidPos.add(new Vector(translateBy.x, translateBy.y,
translateBy.z));

    return new Point(centroidPos.x, centroidPos.y, centroidPos.z);
}
```

```
/**
* @author Sourabh Bhat (heySourabh@gmail.com)
**/
private static Vector scaledCentroidUnder(TriGeom tri) {
    Point[] p = tri.points();
    Point p0 = p[0];
    Point p1 = p[1];
    Point p2 = p[2];

    Vector v1 = new Vector(p0, p1);
    Vector v2 = new Vector(p0, p2);

    Vector areaVector = v1.cross(v2);
```

```
    double xc = areaVector.x * (p0.x * p0.x + p1.x * p1.x + p2.x * p2.x + p0.x *
p1.x + p0.x * p2.x + p1.x * p2.x);
    double yc = areaVector.y * (p0.y * p0.y + p1.y * p1.y + p2.y * p2.y + p0.y *
p1.y + p0.y * p2.y + p1.y * p2.y);
    double zc = areaVector.z * (p0.z * p0.z + p1.z * p1.z + p2.z * p2.z + p0.z *
p1.z + p0.z * p2.z + p1.z * p2.z);

    return new Vector(xc, yc, zc);
}
```

The `Vector` and `Point` classes can be implemented with basic required functionality. The `boundingBox()` `translateTriangles()` methods can be implemented to reduce error, or they can be omitted and the corresponding lines in the code can be removed. The `signedVolume()` method can be implemented as given in the Volume page. For more elaborate implementations have a look at the GeometryHelper class in my CFDSolver repository.