

APPROXIMATING SURFACE AREAS AND VOLUMES

Paul Bouthellier

Department of Computer Science and Mathematics

University of Pittsburgh-Titusville

Titusville, PA 16354

pbouthe@pitt.edu

The problems that we shall examine in this paper are that of approximating surface areas and volumes of 3D objects. In many applications, such as computer modeling, 3D objects are modeled as a series of vertices-basically a wireframe model. These models are seen in packages such as Maple and Mathematica. The new HTML5 canvas also creates 3D models by using a wireframe model. The wireframe sphere shown in this paper was created in the HTML5 canvas using JavaScript.

Given these vertices, basic formulas from calculus using cross and dot products can be used in a computer program, which iterates over all the vertices, to approximate the surface area and volume of an object. Given that any computer representation using vertices is limited by the maximum number of vertices used in the model of the object, the answers derived are approximations. We shall look at several examples in which the true answer is known and compare it to our approximations.

This paper is geared towards people teaching Calculus I-Calculus III.

Approximating the Surface Area

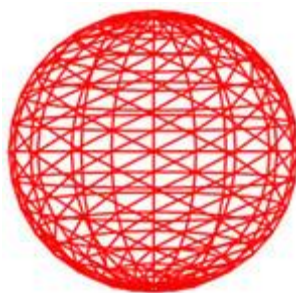


Figure 1-A wireframe sphere created in the HTML5 canvas

Logical questions to ask are:

- How do we get the vertices in the first place?

- Figure a way to group each set of three vertices
- Find the area of each such triangle

In general, the vertices are created using mathematical models created with Bezier curves and surfaces.

After one gets the vertices one must figure how to connect the points (generally using triangles or rectangles) to create the wireframe mesh. Doing this in an optimal way can be an interesting problem in graph theory.

To write a program approximating the surface areas and volumes of wireframes models one needs:

- An array of the given vertices
- A tables of which vertices are connected to which other vertices in order to form the mesh.

Say we are using a triangular mesh as shown in Figure 1 and we want to approximate the surface area using the triangular mesh. We just need to find the area of each triangle given its vertices.

Right out of Calc III: Given three points p_1 , p_2 , and p_3 :

$$p_1=(x_1,y_1,z_1)$$

$$p_2=(x_2,y_2,z_2)$$

$$p_3=(x_3,y_3,z_3)$$

the area is [1]:

$$A=1/2 \cdot |(p_1-p_2) \times (p_1-p_3)|$$

The program then has to loop through all the triangles of the mesh to approximate the surface area.

The sphere created in Figure 1 has a radius of $r=200$ with 16 partitions in both theta and phi. The approximation of the surface area was computed in the HTML5 page as follows:

This page says:

The approximate surface area is 463565.13113384385

Figure 2-using theta=phi=16 partitions

The true answer is 502,654.824 square units. By increasing the fineness of the mesh we can get more accurate results.

For example, using 256 partitions in theta and phi we get the much more accurate result:

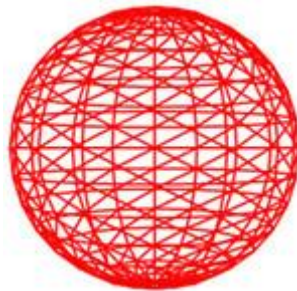
This page says:

The approximate surface area is 500656.91828922566

Figure 3-using theta=phi=256 partitions

It will be noted that if graphics packages such as Studio 3D Max are used, they in general have upper limits to the number of vertices that can be used in their wireframe models-hence there is an upper limit to the accuracy such approximations. Whereas, using the HTML5 canvas and JavaScript, the only limit to the number of vertices and triangles we may use is computer resources.

Approximating the Volume



- Translate the object to the origin via a translation matrix
- Figure a way to group each set of three vertices
- Find the volume of the tetrahedron using the origin as the fourth vertex.

Again from Calc III: Given three points p_1 , p_2 , and p_3

$$p_1=(x_1,y_1,z_1)$$

$$p_2=(x_2,y_2,z_2)$$

$$p_3=(x_3,y_3,z_3)$$

the volume of the tetrahedron is given by [1]:

$$V=1/6 \cdot |p_1 \cdot (p_2 \times p_3)|$$

The program then has to loop through all the tetrahedrons of the mesh to approximate the volume.

In the sphere created in Figure 1 a radius of $r=200$ with 256 partitions in both theta and phi, the approximation of the volume was computed in the HTML5 page as follows:

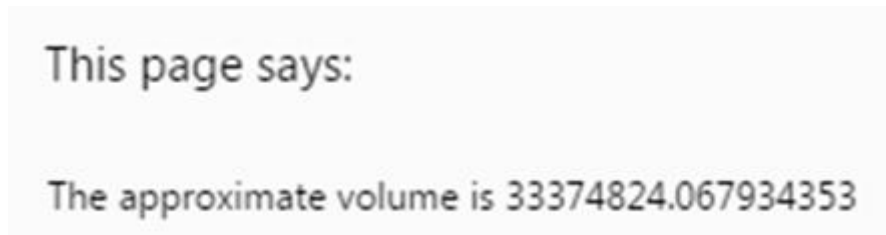


Figure 4-using theta=phi=256 partitions

The true volume is 33,510,321.6384 cubic units. Our approximation has an error of .4%.

Approximating Surface Areas and Volumes Using Graphics Packages

The following examples were done in the graphics package Studio 3D Max.

First let us approximate the volume of the great pyramid of Giza (Height=480.1' and length of base=756.1').

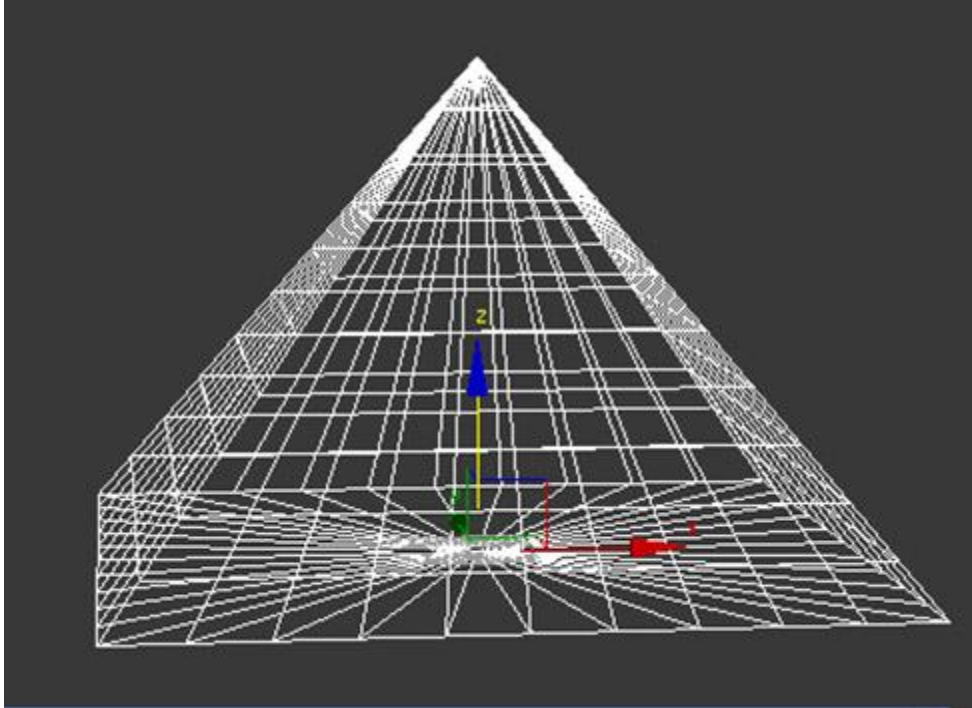


Figure 5-A pyramid created by Studio 3D Max

```

1 vol_use.ms 2 rdvol2.ms
1  fn volume1 =
2  (volume=0.0
3
4  theMesh=snapshotasmesh $
5  numFaces=theMesh.numfaces
6
7
8  for i=1 to numFaces do
9  (
10  Face=getFace theMesh i
11  vert2=getVert theMesh Face.z
12  vert1=getVert theMesh Face.y
13  vert0=getVert theMesh Face.x
14  dV=Dot (Cross (vert1-vert0) (vert2-vert0)) vert0
15
16  volume+=dV
17
18  )
19  delete theMesh
20
21  volume/6
22 )
23 volume1()

```

Figure 6-Looping through the tetrahedrons

```
Welcome to MAXScript.
```

```

volume1 ()
9.14458e+007
volume1 ()
9.1446e+007

```

Figure 7-Approximating the Volume (Not Bad)

Now we can approximate the surface area of a sphere of radius 1.

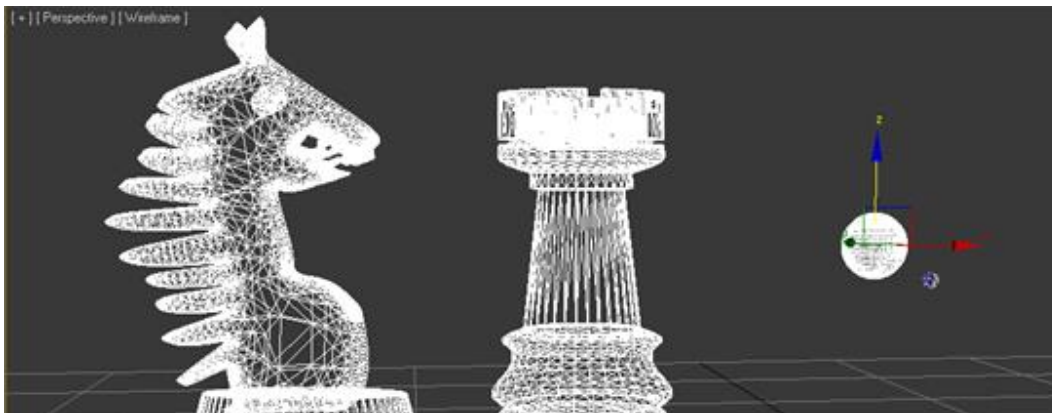


Figure 8-Approximating the Surface Area of a Sphere of Radius 1

```

1 vol_use.ms 2 surfacearea_v4.ms * 3 surface4.ms *
9
10 theMesh=snapshotasmesh someobj
11 numFaces=theMesh.numfaces
12
13 for i=1 to numFaces do
14 (
15
16     Face=getFace theMesh i
17     vert2=getVert theMesh Face.z
18     vert1=getVert theMesh Face.y
19     vert0=getVert theMesh Face.x
20     da=sqrt(Dot(Cross (vert1-vert0) (vert2-vert0)) (Cross (vert1-vert0) (vert2-vert0)))/2.0
21     area+=da*(1.0/n)
22 )
23 delete theMesh
24
25 )
26 area
27 )
28
29 surfacearea1(500)
30

```

Figure 9-Looping through the triangles of the mesh for the sphere

```
Welcome to MAXScript.
|
surfacearea()
4.17342
OK
```

Figure 10-The approximate surface area of the sphere of radius 1

As shown in Figure 8, we can create far more complex wireframe models in graphics packages than is practical by hand and use our techniques to approximate their surface areas and volumes.

By the table of vertices which make up the mesh one could create physical models of such objects using 3D printers as examples in our classes. (I would like to thank the member of the audience of this talk for suggesting this.)

Another interesting application is that of deforming objects and showing how their volumes and surface area change (or don't change). An example is that of the volume of a cylinder shown below:

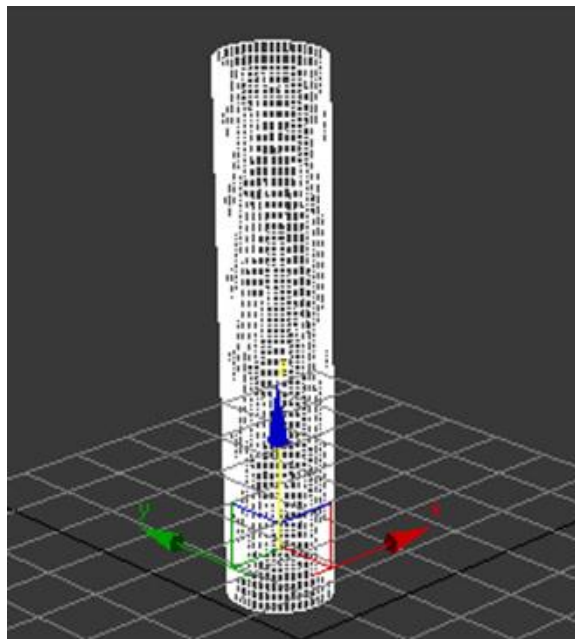


Figure 11-Wireframe cylinder created by Studio 3D Max

The code for looping through the tetrahedrons is the same as before so will not be repeated.

```
Welcome to MAXScript.  
  
volume1()  
31333.3
```

Figure 12-Approximating the volume of the above cylinder using tetrahedrons
Deforming the above cylinder as below should not change its volume.

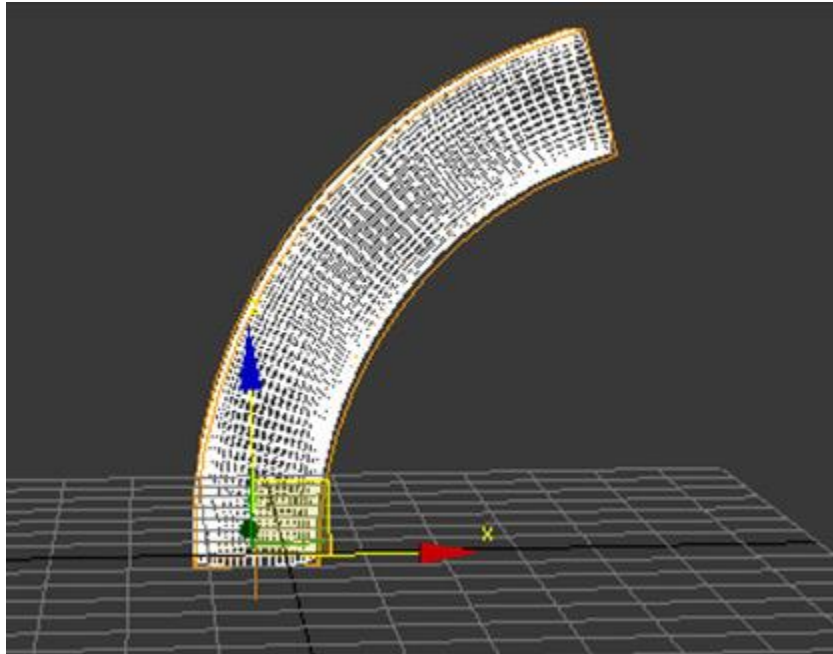
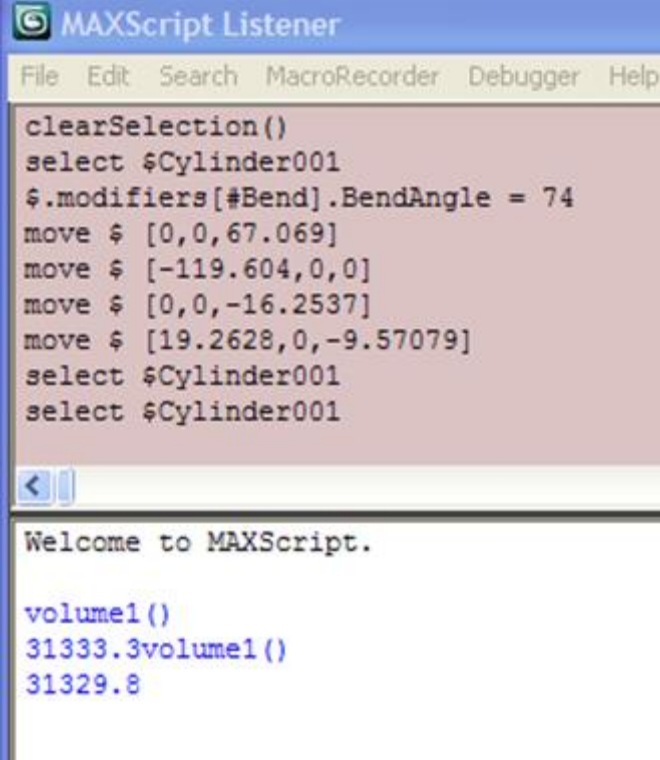


Figure 13-Deforming a piece of tubing



```

MAXScript Listener
File Edit Search MacroRecorder Debugger Help

clearSelection()
select $Cylinder001
$.modifiers[#Bend].BendAngle = 74
move $ [0,0,67.069]
move $ [-119.604,0,0]
move $ [0,0,-16.2537]
move $ [19.2628,0,-9.57079]
select $Cylinder001
select $Cylinder001

Welcome to MAXScript.

volume1()
31333.3volume1()
31329.8

```

Figure 14-Showing how the volume of the deformed object is “almost” the same.

An interesting question we can ask our students is why there is a small difference. Basically, as the vertices will be different the approximated volume will be slightly different as well.

By using graphics packages and the HTML5 canvas we can create mathematical models of 3D objects and approximate properties such as: volumes, surface areas, areas of cross sections, volumes and surface areas of deformations and do it on the fly in classes.

References:

[1] James Stewart, *Calculus* 7th Edition, Brooks-Cole, Belmont, CA, 2012.