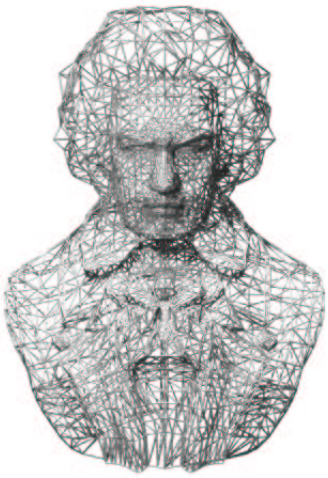


In computer graphics, surfaces are usually represented as unions of triangles. A few points on the surface are given and triples of points form triangles. This illustrates the concept of **linear approximation**. The fact that every triangle is close to the actual surface illustrates the concept of linear approximation. The normal vectors to the triangles are close to the normal vectors of the surface itself.

HOW TO PRODUCE THE BETHOVEN SURFACE.

A file **beethoven.dat** which contains data

```
1.417 -2.541 1.493
1.259 -2.426 1.563
1.236 -2.711 1.686
1.417 -2.541 1.493
1.236 -2.711 1.686
1.409 -2.865 1.613
...
...
```

encoding points in space: every line is a point, three points in a row represent a triangle in space.

So, the first 3 lines from the datafile represent the triangle $\{P, Q, R\} = \{(1.42, -2.54, 1.50), (1.26, -2.43, 1.57), (1.24, -2.71, 1.69)\ \}$. In total, the file contains about 15000 points encoding about 5000 triangles.

PLOTTING THE SURFACE.

To the right is a small Mathematica program which produces a surface from these data. The data are first read, then packed into groups of 3 to get the points, then packed into groups of three to get the triangles, then each triple of points is made into a polygon. All these polygons are finally displayed.

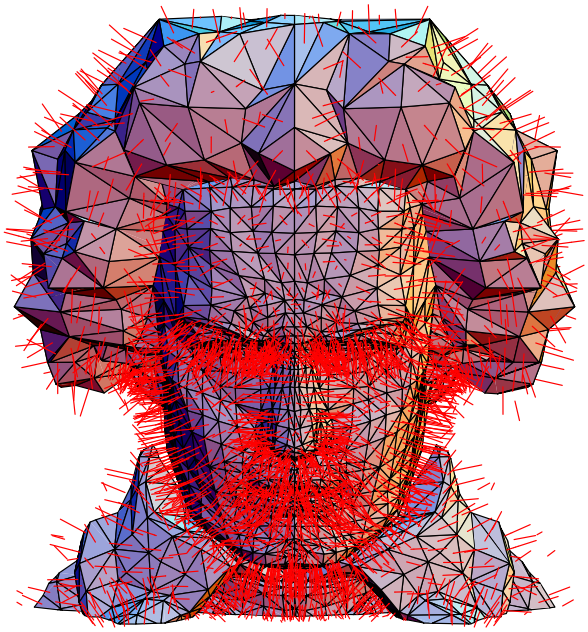
```
A=ReadList["beethoven.dat",Number];
B=Partition[A,3];
P=Partition[B,3];
U=Map[Polygon,P];
Show[Graphics3D[U],Boxed -> False, ViewPoint -> {0,-3,0}]
```

PLOTTING THE GRADIENTS.

We can get the gradient vector on each triangle $\{P_1, P_2, P_3\}$ by taking the cross product $n = (P_1 - P_2) \times (P_3 - P_2)$ and adding lines connecting $Q = (P_1 + P_2 + P_3)$ with the point $Q + n$. We used that

Gradients are orthogonal to the level surfaces.

```
A=ReadList["beethoven.dat",Number];
B=Partition[A,3];
P=Partition[B,3];
U=Map[Polygon,P];
NormedCross[a_,b_]:=Module[{},u=Cross[a,b]; u/Sqrt[u.u]];
NormalVector[{ a_,b_,c_ }]:=Module[{},P1=(a+b+c)/3;
P2=P1-NormedCross[b-a,c-a]/3;Line[{P1,P2}]];
V=Map[NormalVector,P];
Show[Graphics3D[{U,V}],Boxed-> False,ViewPoint-> {0,-3,0}]
```



Here is the output of the above routine. Fortunately, the points of each triangle were given in such a way that we also know the direction of the normal vector. We have chosen the direction pointing outwards. Through every triangle, one gradient vector is displayed.

Once we have the data in Mathematica, we can also export a file, which a raytracer can read. We could also export to a VRML (Virtual Reality Markup Language) and fly around or through it.