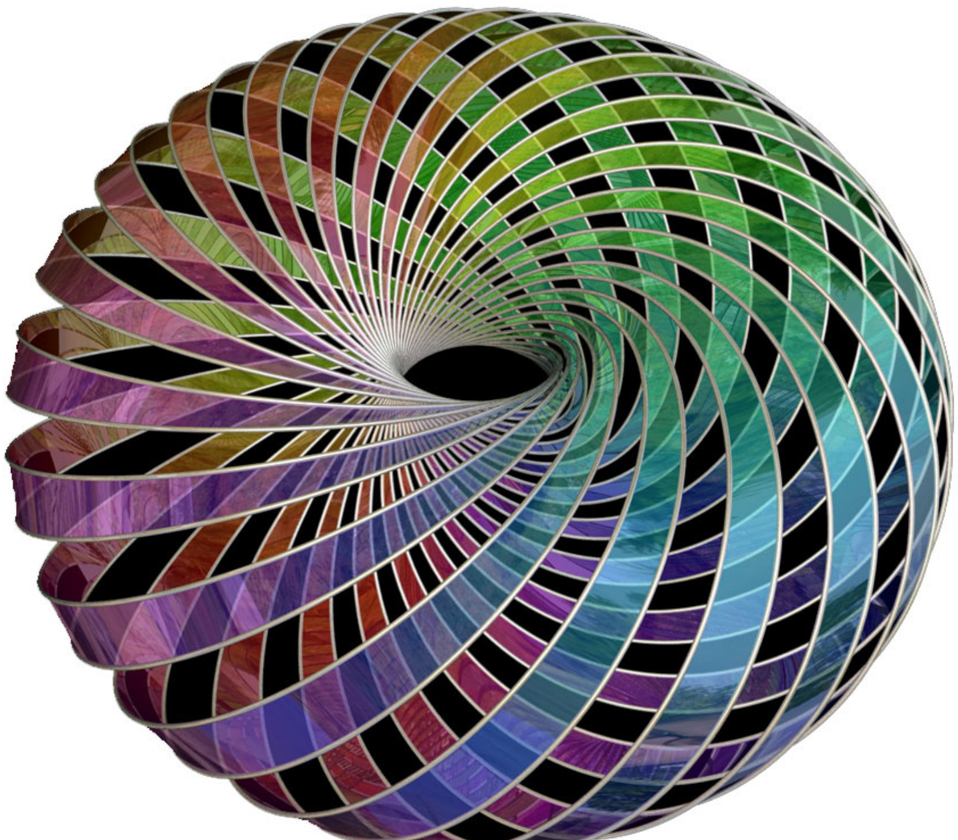
The Hopf Fibration and the Clifford Torus



http://nylander.wordpress.com/2008/08/12/clifford-torus/

The Hopf Fibration

The Hopf Fibration, also known as the Hopf Map, is a map of the 3-Sphere, a sphere in the 4th dimension, to the 3rd dimension.

It describes the 3-Sphere by using circles inside of an ordinary sphere, the 2-Sphere.

3 Sphere and 2 Sphere

If the 2 Sphere is a 3 dimensional shape, why is it not called the 3 sphere?

The 2 Sphere is the 2nd dimension all wrapped up into a ball, giving it the 3rd dimension, but it keeps its name. The 3 Sphere is the 3rd dimension pushed into a sphere, thereby giving it a 4th.

Dimensions

$$S^2 = \{(x_1,x_2,x_3):x_1{}^2+x_2{}^2+x_3{}^2=1\}$$

The equation for a 2-sphere is as shown above, with the **x**s being the distance from the center in the x, y, and z plane.

Dimensions

$$S^3 = \{(X_1,X_2,X_3,X_4):X_1{}^2+X_2{}^2+X_3{}^2+X_4{}^2=1\}$$

So the equation for a 3-Sphere, having only 1 more dimension, must be as shown above.

Dimensions

The Hopf Fibration takes (X_1,X_2,X_3,X_4) to (x_1,x_2,x_3) with the following equations.

$$x_1= 2(X_1X_2+X_3X_4)$$

$$x_2=2(X_1X_4-X_2X_3)$$

$$x_3=(X_1^2+X_3^2)-(X_2^2+X_4^2)$$

The Hopf Fibration

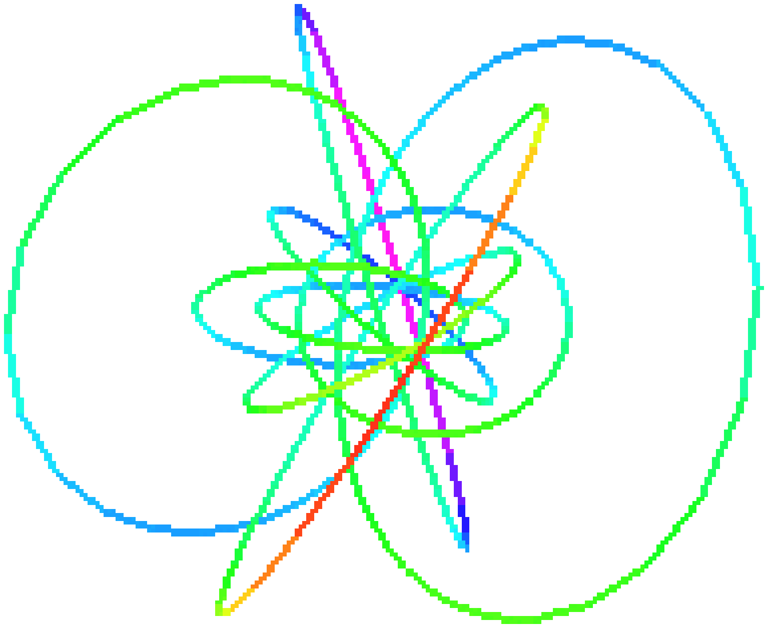
The Hopf Map is a fibration.

A fibration is a "map between topological spaces" that satisfies a certain "homotopy lifting property."

As we mentioned earlier, the Hopf Fibration is the map of the 3-Sphere to the 3rd dimension. Its fibers are great circles, so if we remove the poles, we now have a sphere composed of fibers.

The Hopf Fibration

Each fiber is linked with each other fiber exactly once.

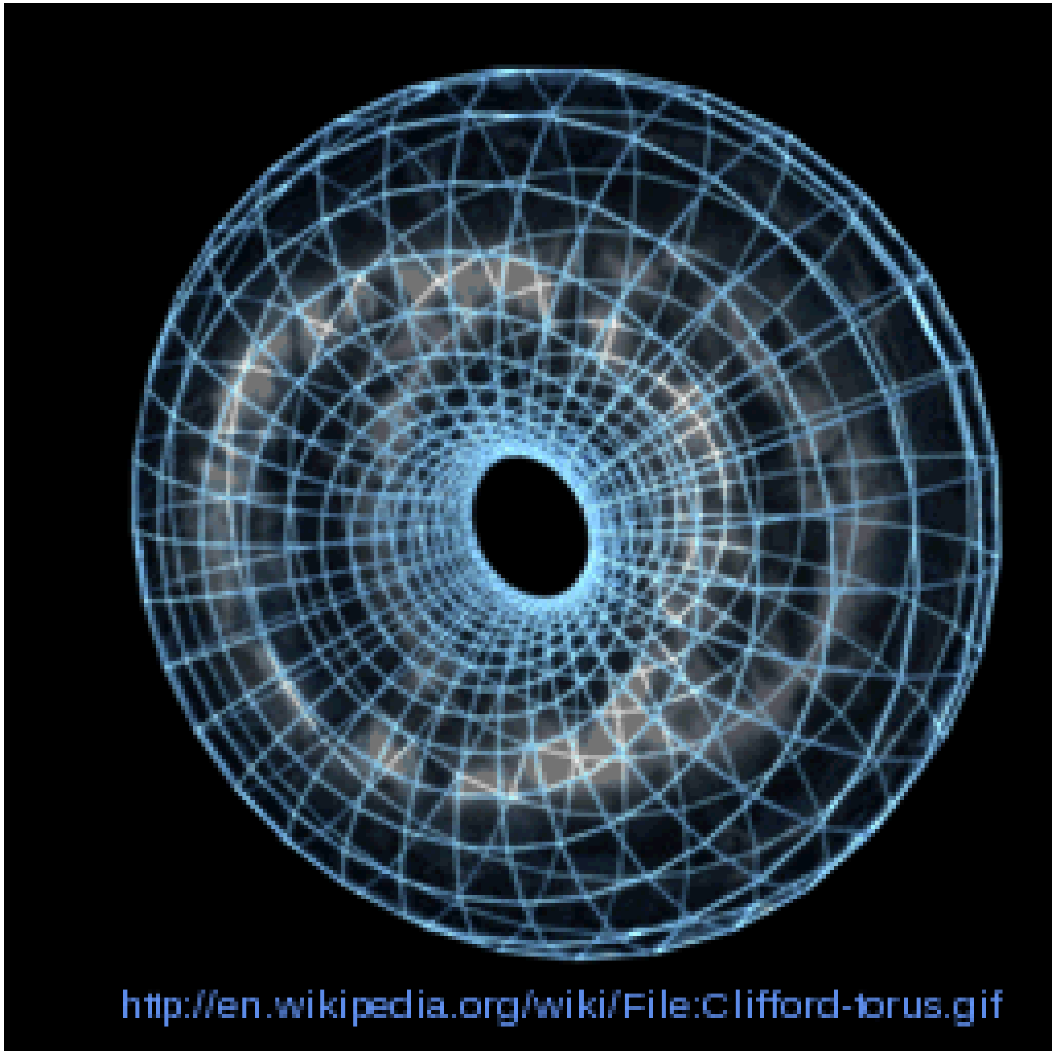


http://math.scu.edu/~ffarris/

The Clifford Torus

When stereographically projected from the fourth dimension to the 3rd dimension, the Hopf Fibration produces what is called a “Clifford Torus.”

Clifford Torus



Equations

Torus: $S^1 x S^1 = \{(\theta, \tau), 0 \leq \theta, \tau < 2\pi\}$

A torus is the product of 2 circles.

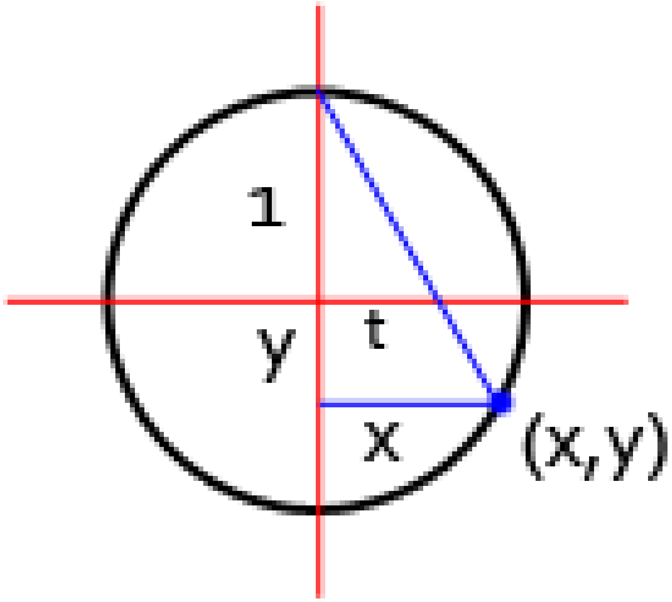
S^3=[((1/sqrt2)(cosθ))

((1/sqrt2)(sinθ))

((1/sqrt2)(cosτ))

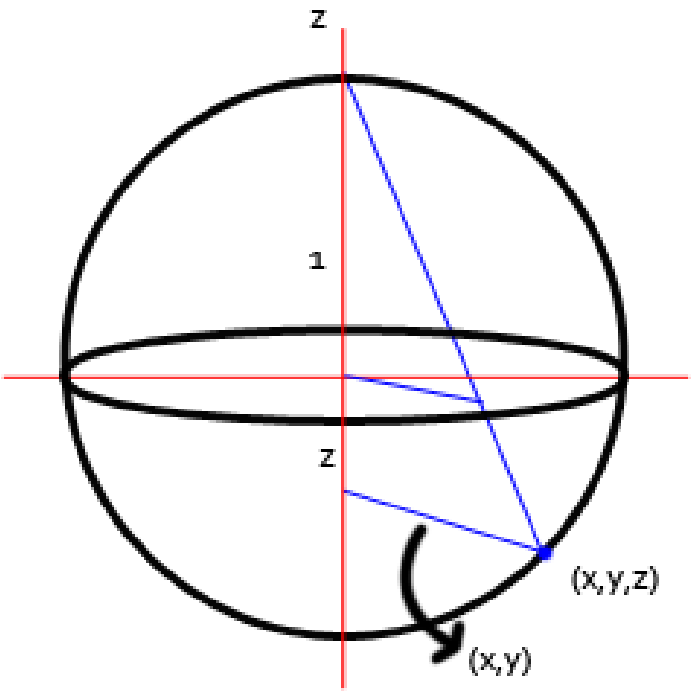
((1/sqrt2)(sinτ))]

Equations (2D)



$t=(x/(1-y))$

Equations (3D)



$$(x,y,z) \rightarrow ((x/(1-z)),(y/(1-z)))$$

Modeling the Clifford Torus

This project started in DPGraph.

Using the equations from the previous slide, we plug those into DPGraph to get:

```
graph3d(rectangular(
      .707*cos(u+v)/(1.0-.707*sin(u-v)),
      .707*sin(u+v)/(1.0-.707*sin(u-v)),
      .707*cos(u-v)/(1.0-.707*sin(u-v)))
      )
```

OpenGL

From DPGraph, the Torus went to PyOpenGL.

The first attempt did not rotate so well, so, with help from Matt Hoffman, numpy matrix multiplication was implemented.

```
n0= .707*C(th+ta)
n1= .707*S(th+ta)
n2= .707* C(th-ta)
n3= .707*S(th-ta)
pointvector=numpy.array([n0,n1,n2,n3])

pointvector=numpy.dot(mat4,pointvector)
pointvector=numpy.dot(mat3,pointvector)
pointvector=numpy.dot(mat2,pointvector)
pointvector=numpy.dot(mat1,pointvector)
```

banclif12test_handmatr

OpenGL

However, that wasn't quite right either.
But a different similar method has seemed to have worked.

```
n0= .5*(C(th) + S(th))
n1= .5*(C(th) - S(th))
n2= .5*(C(ta) + S(ta))
n3= .5*(C(ta) - S(ta))
vert=numpy.array([n0,n1,n2,n3])

vert=numpy.dot(xmat,vert)
vert=numpy.dot(ymat,vert)
vert=numpy.dot(zmat,vert)
```

```
denominator = 1-vert[3]
glVertex3f(vert[0]/denominator, vert[1]/denominator, vert[2]/denominator)
```

Chaplin.py

Chaplin2.py

Syzygy

Fortunately, Chaplin.py managed to get into the cube.

SZGChaplin.py

Sources

http://nylander.wordpress.com/2008/08/12/clifford-torus/
http://mathworld.wolfram.com/HopfMap.html
http://www.math.brown.edu/~banchoff/art/PAC-9603/tour/torus/torus-math.html
http://www.math.brown.edu/~banchoff/art/PAC-9603/tour/torus/torus-movies.html
http://www.math.union.edu/~dpvc/math/4D/stereo-projection/welcome.html
http://theory.org/geotopo/3-sphere/html/node4.html
http://www.cutoutfoldup.com/980-clifford-torus.php