Forever Knot —a non-harmonic torus helix (not-knot)

Main Page ·⋱· Project References

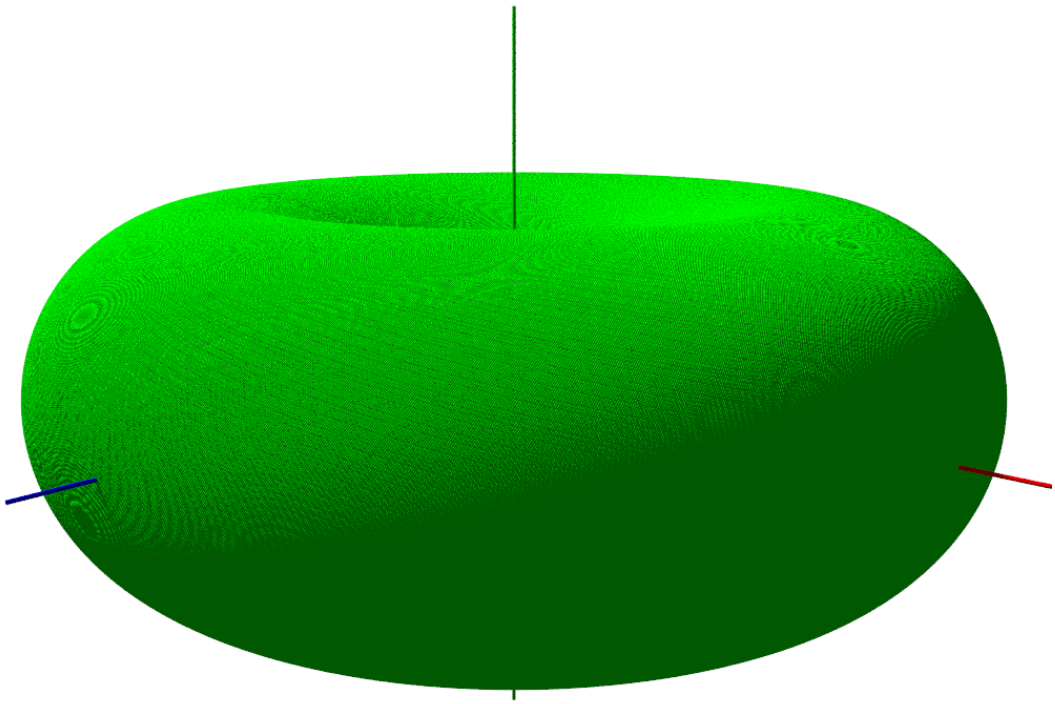
Topics: Geometry, Torus, Knot Windings, Golden Ratio

The 'Forever Knot'

Categories:torus knot, irrational, golden ratio, golden torus, golden sope, povray, code

This is not a real knot, because it does not connect with itself, and continues to trace a non-harmonic path. If it could continue, it would yet never reconnect with its beginning ever ever. The illustration has 888 helical windings on the donut.

The not-knot illustrated above has a slope of Phi:1, and the donut (torus) is golden, which places the slope of the winding helix as Phi at a very irrational number, and therefore there can be no integer harmonic condition where the beginning and end of the knot meet at the same point to repeat the same winding path.



Knot trivia: A 'golden torus' has a minor radius that is four powers of the golden ratio (1.6180339887...) less than th radius. In terms of the radius of the torus hole of a golden donut, the hole radius from the torus center point is Phi^4 les the major radius —scalable with a multiplier on the major:hole proportion.

POVRay code for the not-knot as illustrated

```
  // POVRay scene file: groupwave.pov        //

//  Copyright Don@groupKOS.com
//  +SF0 +EF99 +KFI0 +KFF99 +W800 +H600 +A

#version 3.7;
global_settings{ assumed_gamma 1.0 }
#default{ finish{ ambient 0.1 diffuse 0.9 }}

#include "colors.inc"
```

```
// An area light (creates soft shadows)
// WARNING: This special light can significantly slow down rendering times!
light_source {
  <0,0,0>            // light's position (translated below)
  color rgb 1.0      // light's color
  area_light
  <8, 0, 0> <0, 0, 8> // lights spread out across this distance (x * z)
  4, 4               // total number of lights in grid (4x*4z = 16 lights)
  adaptive 0         // 0,1,2,3...
  jitter             // adds random softening of light
  circular           // make the shape of the light circular
  orient             // orient light
  translate <40, 80, -40>   //  position of light
}

// sky (tilt) camera
camera {
  //orthographic
  location  <.3,.1,.3>*84
//  location  y*66
  angle 45
  sky       y //<1.0, 0.5,  0.0>
  look_at   <0.0, 0.0,  0.0>
  right     x*image_width/image_height
}

// Set a color of the background (sky)
background { color rgb< 1, 1, 1> }

// Render Count number of spheres at radius around Y axis.
#macro circle_sphere_array(Count, arrayRad, sphereRadius, txtr)
    // Radial pattern centered on the Y axis of count divided into 360.

#end // circle_sphere_array().



#macro plotTorusKnotUnitQDensity(R, r, p, q, hand, plotRadius, plotResolution, lugRadius, densityHigh, densityLow, txtr)
    // Plot 1/q-th of a whole p:q knot //
    // Vary the radius of the plotting sphere to abstract the       //
    // radius of a certain magnetic flux density about that point   //
    //
    // hand = chirality of the knot windings; a clockwise or a counter-clockwise helix around the donut. //

    union{

        #local unitEndAngle  = ( 360*p/q );
        #local plotAngleDelta = unitEndAngle / plotResolution;

        #if(showEquatorConnectorLugs)
            // Draw connector lug spheres //
            union{
                // Draw the D.C. termination lugs at the ends of the unit segment windings //
                sphere{ 0, lugRadius
                        scale x*3
                        translate x*(R+r)
                        rotate y*0

                        texture{txtr}
                        }
                // Draw the D.C. termination lugs at the ends of the unit segment windings //
                sphere{ 0, lugRadius
                        scale x*3
                        translate x*(R+r)
                        rotate y*180

                        texture{txtr}
                        }
            } // end unioN

        #end // if()

        union{ // of knot winding plot points //

            //sphere{0, .3 translate x*(radiusMajor+radiusMinor) rotate y*(000*p/q)}    // test cursor
            //sphere{0, .3 translate x*(radiusMajor+radiusMinor) rotate y*unitEndAngle} // test cursor

            #local plotAngle = 0;          // While-loop index.
            #while (plotAngle < unitEndAngle*222*2*2 )

                sphere{ 0,  plotRadius
                    // Move objPlot from the origin to the X axis by the minor radius plus major radius.
                    translate  x*(r)

                    // Rotate objectLug in the XY plane around the z axis, for either hand.
                    rotate z*(hand*plotAngle*q/p)  // Extract q degrees from (360*p/q).

                    translate x*R

                    // Rotate objPlot in the XZ plane around the Y axis of the torus hole, for either hand.
```

```
                rotate y*(hand*plotAngle)

                texture{pigment{color Green}}
            }

            // Rinse and repeat.
            #local plotAngle = plotAngle + plotAngleDelta;

        #end // while

    } // end union
  } // end union
#end // plotTorusKnotUnitQDensity()


#local txtrGreen = texture{pigment{color Green}};
#local txtrRed   = texture{pigment{color Red}};
#local txtrBlue  = texture{pigment{color Blue}};

union{ // scene union

  #local phaseCount       = 1;
  #local unitQResolution  = 3000;
  #local patternLength    = phaseCount/2;

  #local plotRad          = .025;
  #local lugRad           = plotRad*2;
  #local AxesDia          = .1;
  #local showEquatorConnectorLugs = +1;

  // A 4th-degree golden donut //
  #local R = 6.8541;
  #local r = 5.8541;

  // Knot winding-ratio parameters (p:q)
  #local p = 8;
  #local q = 5;

  // axes //
  cylinder{-x*15, +x*15, AxesDia/2 texture{txtrRed   } }
  cylinder{-y*10, +y*10, AxesDia/2 texture{txtrGreen } }
  cylinder{-z*15, +z*15, AxesDia/2 texture{txtrBlue  } }
  #local phi = pow(5,.5)*.5+.5;

  #local phi0 = pow(phi,0);
  #local phi1 = pow(phi,1);
  #local phi2 = pow(phi,2);
  #local phi3 = pow(phi,3);
  #local phi4 = pow(phi,4);
  #local phi5 = pow(phi,5);


  #local j=0;
  union{ // scene union

    #local phaseCount       = 1;
    #local unitQResolution  = 3000;
    #local patternLength    = phaseCount/2;

    #local plotRad          = .025;
    #local lugRad           = plotRad*2;
    #local AxesDia          = .1;
    #local showEquatorConnectorLugs = +1;

    // A 4th-degree golden donut //
    #local R = 6.8541;
    #local r = 5.8541;

    // Knot winding-ratio parameters (p:q)
    #local p = 8;
    #local q = 5;

    // axes //
    cylinder{-x*15, +x*15, AxesDia/2 texture{txtrRed   } }
    cylinder{-y*10, +y*10, AxesDia/2 texture{txtrGreen } }
    cylinder{-z*15, +z*15, AxesDia/2 texture{txtrBlue  } }
    #local phi = pow(5,.5)*.5+.5;

    #local phi0 = pow(phi,0);
    #local phi1 = pow(phi,1);
    #local phi2 = pow(phi,2);
    #local phi3 = pow(phi,3);
    #local phi4 = pow(phi,4);
    #local phi5 = pow(phi,5);


    #local j=0;
    union{
      #while(j < phaseCount)
        union{
          //plotTorusKnotUnitQDensity( p, q, 12.9787137637477918123,  8, +1,  plotRad,  unitQResolution, lugRad   2,       1,       txtrR
```

```
            plotTorusKnotUnitQDensity( p, q, phi1,   phi0, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )
            //plotTorusKnotUnitQDensity( p, q, phi3,   phi2, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )

            /*
            //plotTorusKnotUnitQDensity( R, r,  p,  q, hand, plotRadius, plotResolution,  lugRadius, densityHigh, densityLow, txtr)    */
            //plotTorusKnotUnitQDensity( p, q,  1,  1, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )
            //plotTorusKnotUnitQDensity( p, q,  2,  1, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )

            //plotTorusKnotUnitQDensity( p, q,  3,  2, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )
            //plotTorusKnotUnitQDensity( p, q,  5,  3, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )
            //plotTorusKnotUnitQDensity( p, q,  8,  5, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )
            //plotTorusKnotUnitQDensity( p, q, 13,  8, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )
            //plotTorusKnotUnitQDensity( p, q, 21, 13, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )
            //plotTorusKnotUnitQDensity( p, q, 34, 21, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )
            //plotTorusKnotUnitQDensity( p, q, 55, 34, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )
            //plotTorusKnotUnitQDensity( p, q, 89, 55, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )
            //plotTorusKnotUnitQDensity( p, q, 144, 89, +1,   plotRad,    unitQResolution, lugRad     2,          1,         txtrRed  )

            rotate y*(360*j/phaseCount)

          } // end object

          #local j = j + 1;
        #end // while

        rotate y*(360*j/phaseCount)
    }
    rotate y*(360 * frame_number/final_frame)
} // end scene union
```