Look, Ma, No Trigonometry!
Motor orbits for the working graphician.

Steven De Keninck
University of Amsterdam

1 Abstract

In this talk we consider the use of PGA motor orbits for a number of practical constructive geometry tasks in computer graphics. Specifically, we consider the creation of parametric surfaces and wonder if the eloquent language of geometric algebra can help clean up the abundance of coordinate manipulations and trigonometry functions that this task typically involves.

As a guiding example, we will be constructing the geometry of a rectified torus knot, including vertex positions, normal, tangent and bitangent vectors. We compare the two quite different algebraic approaches, and see how they translate into actual code.

We'll demonstrate substantial savings in code size, precision, readibility, clarity and maintainability of the PGA approach, and hope this presents a compelling argument for the (initial) learning curve of geometric algebra.

2 Teaser Results

The construction of standard geometries as the sphere, torus, cylinder etc are a default part of both the graphics programmers curriculum and every 3D tool and engine. It is not a task that is generally considered hard, and so perhaps not the most obvious candidate for reworking. We hope to show however that a substantial unification and simplification is being overlooked, and will remedy that through the use of PGA motor orbits.

The PGA algebra models the Euclidean group of distance preserving transformations, and its even subalgebra, the important subgroup of rigid body transformations. This allows us to write translations, rotations and their combinations as exponential functions, which in turn enables a very eloquent method of describing parametric geometries.

The resulting objects, parametric functions that produce a motor, are called motor orbits and allow us to take a completely topological approach to geometry. This approach will effectively make a cylinder a product of a line segment with a circle, a torus the product of two circles, etc.

We'll show how these motor orbits encode not just the points on the surface parametrically, but also the complete local orientation. The final approach is contrasted against an industry standard implementation (three.js), where some teaser results are available in the table below:

	Classic LOC	**Orbit LOC**
Shared	-	20
Line Segment	-	1
Circle	60	1
Sphere	82	1
Cylinder	167	1
Cone	20	1
Torus	74	1
Torus Knot	105	11

Table 1: Comparison of Lines of Code for both approaches.

As table 1 shows, we can get substantial savings on code size, and are able to isolate all of the shared code, as it is independent of the specific orbit.

The talk ends with the full glsl implementation of the torus knot geometry, in a completely coordinate and trigonometry free manner:

```
/***** TORUS KNOT CURVE **********/

// Circle Orbit. Rotation after Translation
motor circle ( float t, float radius, line axis ) {
  return gp_rt( exp_r(t * PI, axis), exp_t(radius, e01) );
}

// Torus Knot. Product of two circles.
motor knot ( float t ) {
  return gp_mm( circle( t * P, r2, e12 ), circle( t * Q, r1, e31 ) );
}

/***** DERIVATIVE OF TORUS KNOT CURVE **********/

// Derivative of circle. Axis * PI * Circle.
motor dCircle ( float t, float radius, line axis ) {
  return gp_rm( motor( 0.0, axis[0] * PI, 0.0, 0.0, 0.0, 0.0), circle(t, radius, axis) );
}

// Derivative of torus knot.
motor dKnot ( float t ) {
  return gp_mm( P * dCircle( t * P, r2, e12 ), circle( t * Q, r1, e31 ) )
       + gp_mm( circle( t * P, r2, e12 ), Q * dCircle( t * Q, r1, e31 ) );
}

// Derivative of origin as moved by torus knot.
// k' * e123 * ~k  +  k * e123 * ~k'
direction dOrig ( float t ) {
  return gp_mom_mom( dKnot(t), knot(t) );
}

/***** RECTIFIED TORUS KNOT CURVE **********/

// Rectified torus curve.
motor rKnot( float t ) {
  motor     M    = knot(t);
  direction to   = sw_md(reverse_m(M), dOrig(t));
  motor     corr = sqrt_m(gp(normalize(to), e021));
  return gp(M, corr);
}

// Final extruded torus knot.
motor tube( float s, float t ) {
  return gp_mm( rKnot(s), circle(t - s + 0.125, r3, e12) );
}
```

3 Talk Overview

1. **5m - The PGA Exponential function.**

   A minimal introduction to outline the PGA concepts used in the talk, briefly summarizing the notation used, and the geometry of the group and algebra.

2. **5m - Products of circles and lines.**

   We'll explore how parametric PGA motor orbits provide a topologically eloquent method of describing a wide range of geometries.

3. **5m - Derivatives of circles and lines.**

   We show how the exponential form make the calculation of explicit derivatives easy, improving the precision of the reference implementation we're comparing against.

4. **5m - The torus knot**

   We'll put all concepts together and construct several versions of the rectified torus knot geometry.

5. **5m - Conclusions and Results.**

   We summarize the advantages of this unified representation, and highlight some of the gains we uncovered.

A For the reviewers

The open-source 'Look, Ma, No Trigonometry!' project provides an example implementation that uses PGA motor orbits for the construction of parametric curves and surfaces. It is accompanied with a detailed writeup that will be used as the source for this talk.

It is available at `https://enki.ws/LookMaNoTrigonometry/index.html`, and contains a large number of interactive animations that will be used to illustrate the concepts used in this talk. The planned presentation is very hands-on and practical, and targets engineers and students working in computer graphics.