Dynamic Geometry: New Math from Old?

June Lester
Centre for Experimental and Constructive Mathematics
Simon Fraser University, Canada
jalester@cecm.sfu.ca

Introduction. Dynamic geometry has its origins in the recent evolution of computer geometry software packages such as Geometer's Sketchpad [1], Cabri II [2] and, more recently, Cinderella [3]. Originally developed to support the teaching of school level euclidean geometry (in the role of "onscreen construction tools"), these packages have quickly become an integral part of geometric pedagogy and, increasingly, of geometric research.

As a topic of study, dynamic geometry (DG) concerns itself with the properties and behaviours of onscreen geometric objects which move or change over time. In this note, we contend that, though initially modeled on classical static euclidean geometry (SG), the mathematical nature of DG is fundamentally different from that of SG. We then explore some of the issues around trying to establish mathematical foundations for DG.

Caveat 1: The issue here is not the mathematical underpinnings of dynamic geometry software - the mathematical analysis and computer programming constructs that produce the onscreen pictures. The intent is to look at the feasibility of an *intrinsic* description of onscreen DG objects, in terms of both their inherited euclidean properties and the novel properties created by motion. The ultimate goal (towards which this note is only a very preliminary exploration) is to produce a formal description of DG in the style of classical axiomatizations of SG. Such a formalization (if indeed one can be found) could also be useful prescriptively to help resolve some of the inconsistencies within and among computer geometry software packages.

Caveat 2: What follows is by no means a precise exposition and is not meant to be - a precise formulation of the issues involved would be halfway to their resolution. Precision is part of the ultimate goal; here, there are more questions than answers (and unfortunately, more confusion than coherence). The issues tend to overlap or blur into each other, and may turn out to be as much philosophical as mathematical. The intent here is merely to identify some of the more salient ones.

How is dynamic geometry different? Onscreen geometrical objects (or indeed onscreen objects of any type) lie somewhere in the middle of the concrete-abstract continuum of objects in the universe. Though they aren't physically real, neither are they totally abstract: they possess a visual form that can move, change or interact with external "users". Furthermore, the nature of the user interaction - direct manipulation with a mouse - gives those users an almost tactile sense of interacting with physical objects. Our perception of onscreen points, lines and circles, etc. is thus fundamentally different from our perception of their more static counterparts (dots on a page): instead of fixed sets of points embedded in a plane, DG objects are perceived more as changeable objects "floating" on top of a plane, which can be seized, moved, resized or reformed.

The basic mathematical consequence of this "perceptual physicality" is that, unlike SG objects, a DG object can be moved or modified and still retain its identity. In SG, points at different positions are always different points; in DG, a point moved to a new position is still the same point - the same onscreen object. In SG, the circle with centre (1,2) and radius 3 is distinct from the one with centre (4,5) and radius 6. In DG, the circle with centre (1,2) and radius 3 can have its centre dragged to position (4,5) and its radius scaled to 6 and retain its identity - it remains the same circle. In DG, position, centre, radius, etc., instead of defining, labeling or identifying an object as they do in SG, are now variable attributes of that object.

But DG object behaviours are not completely physical either: unlike "real" objects, two or more DG objects can occupy the same position and remain separate objects. Dragging one point onto another at position (7,8) does not produce a single point at (7,8), merely two distinct points with (coincidentally) the same position, which can be separated at will. Distinct lines can coincide, as can distinct circles: in DG, coincidence is not identity.

Thus, if mathematics is to model perception (as it does for SG), DG is fundamentally different from SG. These basic differences should be reflected in any axiomatic formalization of DG.

What should a formalization of dynamic geometry do? To axiomatize DG, we must first choose some objects (e.g. points, lines ...) as "primitives" (basic objects not defined in terms of simpler component objects) and then define the fundamental relations among them; from these all else is to be derived. For DG, this is more difficult than for SG: if the DG mantra is to be "geometric truth is dynamically invariant", then even notions as basic as incidence are affected: a point does not become incident with a line merely by occupying a position on it, but by being defined, constructed or proven to lie on all positions of the line. Dynamic invariance is in fact not all that easy to pin down: the motion of a point whose only constraint is that it lie on a given line, for example (e.g. via Sketchpad's "construct point on object " menu choice) is not uniquely defined when that line moves. Onscreen, the point has to go somewhere specific, but exactly where is a programming choice, not a mathematical one: mathematically, the point need only end up somewhere on the line.

Even our choice of primitive objects is affected by motion. It would be unproductive to view a circle as a set of points, for example, since any movement of the circle would then have to prescribe that of each individual point on it. So at the very least, we need points, lines and circles as primitive objects. Then *which* points, lines and circles? All points or objects other than those explicitly defined, constructed or proven to exist within the context of a given configuration are irrelevant dynamically, since their movement with respect to the configuration is undefined and undefinable. (It may help to think of these configurations as something like linkages here - moveable configurations of objects.)

Rather than the SG infinite plane of points and objects formed from subsets of points, the appropriate universe for DG may perhaps be more usefully taken to be a collection of "templates" for primitive objects (e.g. a generic "line" with line-like properties and behaviors) and rules for constructing dynamically invariant configurations from a finite (or countable) number of instances of these types. There are programming constructs for this sort of concept, but they lead in a more complicated direction than this simple axiomatic first approach.

Assuming there is a coherent way of dealing with all of the above, how do we check the consistency of the axioms? Traditionally in geometry, consistency has been verified by reference to a concrete model of the geometry in terms of some other consistent system like the real numbers. But if motion is involved, the most likely source of models appears to be the physical world, and as we've seen, DG is not completely physical. Some other method will have to be used.

Some other issues.

***Parent-child relations*.** This is a significant DG issue that has only an insignificant or non-existent SG counterpart. Parent-child relations describe the sequence of defined or constructed objects in a configuration – the line joining two points is a child of both parent points, for example. For DG, this distinction is important: parental motion normally determines that of the child but not vice versa (though some software implementations of DG allow child-initiated movement, usually translation of the whole configuration). So we would expect the visually identical constructions "two lines, then their intersection point" and "a point on a line and then another line through the point" to behave differently when the point is dragged (the former possibly allowing translation of the two lines; the latter moving only the second line). Any axiomatization of DG must capture such parent-child relations, and also account for any variances when one or more of the ancestors is "pinned" (fixed).

Existence of intersections. This issue, the bane of DG software developers, doesn't occur in SG: what becomes of object intersections when the objects are pulled apart (i.e. to non-intersecting positions)? If two intersecting DG circles are dragged apart, do their intersection points just disappear? This is axiomatically and epistemologically unpalatable: if we are treating points "physically" as objects instead of positions, then either they exist or they don't, irrespective of position. And then, if the circles are dragged together again, are the intersection points the same ones as before? and if so, which is which? Computationally, different DG software packages deal with such intersections in various and often incompatible ways, sometimes using relatively sophisticated mathematical constructs such as continuity [3]. But such constructs are chosen more to avoid visual "jumping" than to adhere to any elementary mathematical principles, and even then, their application may be problematic.

For example, suppose a moving circle passes a stationary circle of equal radius with its centre constrained to a (horizontal) line through the centre of the stationary circle. What should happen to their points of intersection as the moving circle passes by the other? In Sketchpad, the upper and lower intersection points exchange positions after the circles coincide - a result that seems somewhat counterintuitive and undesirable: continuous motion should ideally produce continuous consequences. In Cinderella, the points don't exchange positions, a more “natural” behaviour - or is it? Suppose instead that the moving circle's centre passes only very near the centre of the fixed circle (so the circles never exactly coincide). Simple experimentation (in either program) shows that the intersection points then do a rapid flip around the stationary circle as the other passes by. So if small perturbations in position should cause only small changes in behaviour, perhaps the flipping behaviour is indeed more appropriate? In fact, without a mathematical framework to judge by, either behaviour is arguably better than the other.

Conclusion. There are other issues affecting a potential DG axiomatization, probably many more than identified here. Transformations, for example: if the motion of objects is already part of the formalization, what role if any do rotations, translations and reflections play? If, like Hilbert, we identify a geometry by its group of transformations, where does DG fit in? And so on.

From the foregoing discussion, it seems likely that the most intractable challenges to this type of formalization of DG will be coincidence and existence/non-existence - the "non-physical" relations of DG. Coincidence of objects allows all sorts of singular situations: even the basic "two points determine a line" axiom of SG breaks down for coincident points in DG. Triangles with coincident vertices are degenerate; circles with coincident centres have no radical axis, and so on. Parallelism causes similar effects, as may any "borderline" relation. The problems caused by existent/non-existent intersection points were described above; similar problems occur with common tangents to pairs of circles, for example. And to complicate matters, the two issues are interrelated: situations where the coincidence of some objects can cause non-existence of others are easy to find.

In light of these and other as yet unforeseen difficulties, then, it may well be that a simple axiomatic formalization of DG is impossible, and that the appropriate way to formalize DG is in terms of some other more complicated mathematical system.

References

1. **Geometer’s Sketchpad**, Key Curriculum Press, Emeryville, California, <http://www.keypress.com/catalog/products/software/Prod_GSP.html>

2. **Cabri II**, Institut d'Informatiqe et de Mathematiques Appliquees in Université Joseph Fourier de Grenoble, France <http://www.cabri.net/>

3. **The Interactive Geometry Software Cinderella**, by Jürgen Richter Gebert and Ulrich H. Kortenkamp. Springer, 1999 <http://www.cinderella.de>