Making Geometry Dynamic: Design Considerations in Mathematical Interactivity

Nicholas Jackiw
nicholasjackiw@gmail.com
Vancouver, Canada

Abstract: *This paper surveys situations in the early development of Dynamic Geometry Software in which designers had to "invent" plausible mathematical behaviors for specific dynamic configurations. It offers both a case study in the design of mathematical software and a reflection on the potential contribution dynamism makes to the history of mathematical representation.*

1. Introduction

How do mathematical representations—mathematical media, notations, and conceptions—arise, and how do they evolve? Sixty years into a growing diffusion of computational activity and software technology across mathematical practice, and half as long into an era in which various Dynamic Geometry packages have arisen as the most widely used such tools in the teaching and learning of mathematics, is it surprising that not one of these packages feature the most characteristic—even emblematic—representation of school geometry across the century before these new tools' arrival, the two-column proof? The nature of activity changes as the tools change through which activity is pursued. Adopting the position that emerging tools and representations shape, as much as are shaped by, standing ideas and conceptions of a domain, this paper surveys the origins of Dynamic Geometry software for moments where the gestural and visual paradigm for mathematical interaction introduced by such software had to extend or refigure practices and implications of "static" geometry to enable its new and dynamic approach.

This account is personal and to some degree subjective. My colleague Steven Rasmussen and I first coined the phrase "Dynamic Geometry," and I write about its design from my experience creating one of its first incarnations: *The Geometer's Sketchpad* [1]. My account focuses primarily on *Sketchpad,* but attempts to portray other perspectives in the fertile design dialog that occurred across software packages beginning in the early 1990s and continuing, to some extent, to date. However, my broad interest behind this account—in, through design, creating and clarifying the role of gesturally-mediated dynamic manipulations in mathematics software—features not one, but two, topics notably difficult to engage in print. Dynamic gesture, as I use the term, refers to physical motions and actions (hand, or hand/eye/body movements) that almost by definition constitute non-verbal and non-symbolic acts of communication, whether between people or between a person and a manipulated machine. I can describe such actions in print, but I cannot *communicate* them here effectively. And dynamic software is, of course, software whose content is not only graphical in nature but moving, animating, and evolving continuously in time and in response to gestured directions. It is decidedly *not* software that presents a text of symbols conveniently transcribable in print. Scholarship in both domains struggles with capturing the essential physicality of the gesture, and the inherent dynamics of the software, in words and static pictures. This paper accompanies an invited lecture I shall give at ATCM, but where within a few minutes in a talk I can give a half dozen compelling examples of dynamic gestures and dynamic software responses to them, in text one labors for paragraphs to evoke the physicality or dynamics critical to a single example. Thus, the presentations shall necessarily differ, as the media are so different in regard to critical fault-lines of my topic. (Fortuitously, this very fact—that choice of representational media is consequential! —is a fundamental conclusion of both my presentations.) My talk covers examples drawn not just from *Sketchpad* but also from my lesser-known works and from my work-in-progress and aims to highlight design comparisons and their implications. In this paper instead, I focus on the single example of dragging within Dynamic Geometry, to detail a single design in context, while hoping the reader has familiarity with the actual *act* of dragging in Dynamic Geometry sufficient to compensate for the deficiencies of print description. Even before we take up that subject, however, we must first ask "Why consider dynamics at all?" and "Why consider design?"

2. Why Dynamism?

Why focus on dynamic motions and gestures—on the often non-verbal waving of our hands—when we more typically consider mathematics as a set of concepts, skills, and procedures? Broadly, dynamic gestures and motions are a placeholder for an entire spectrum of embodied actions, intuitions, and understandings we might consider when we replace a perspective of mathematics as a knowledge domain in favor of a perspective on it as a human activity. Neither perspective invalidates the other: the first simply considers a set of propositions, symbolically expressed and organized by principles of abstraction, whereas the second considers the construction, communication, comprehension, and use of those propositions. And yet, a student of mathematics history rapidly learns that one of the social conventions of mathematical practice is to pretend mathematical practice has no social conventions: neither animate people nor animate—moving—objects appear to be acceptable in the empyrean of formal mathematics, in which (in the famous words of Nicolas Balacheff [2]) knowledge appears only and always "detemporalized, depersonalized, and decontextualized." Since this postured independence from all human action or motion makes mathematics a decidedly inhospitable one for many human students (who themselves are irrevocably temporalized, personalized, and contextualized!), it is worth briefly touring arguments for the importance and contribution of the dynamism, motion, and gesture to mathematical thinking.

Such a tour could usefully begin at the dawn of mathematics. Where Western accounts often trace math's origin to the work of the ancient Greeks—Plato coins the term "mathematics"—of course at that time arithmetic and geometry were already well-developed practices, so we look earlier. Drawing on the anthropological literature and Indian, Chinese, Babylonian, and Egyptian documents as well as Greek ones, Seidenberg ([3], [4]) argues compellingly that both disciplines have their earliest origin in religious ritual, and that—specifically—ritualistic chanting and religious procession are the forebears of number sequence, of cardinality and ordinality. Thus the moving body, and the social concert of moving bodies in song and in parade, is the very font of mathematics. Jumping 5,000 years ahead to the opposite end of the timeline, Lakoff and Nunéz [5] trace an arc of influence to present-day mathematics, which they argue, from cognitive science perspectives, is constructed through a small but powerful set of embodied understandings in which mathematical abstractions are metaphors based on concrete physical actions and sense perceptions of the physical body. Inside these two historical bookends, the Platonic move *away* from embodied and enactive (motion-based) understandings seems almost contrarian. Plato arises in an intellectual milieu in which Zeno's "paradoxes" have only recently challenged arguments based on temporality and temporal continuity, so there is philosophical motivation to such a disposition. But present-day scholars like James Kaput or Brian Rotman might argue such a turn arises as much as a consequence of the available infrastructure for mathematical representation and communication in the time of the Greeks (a move from the diagrams of "sand reckoning" to the symbols of writing and "print") as from any explicit epistemological rejection of embodiment, temporality, or dynamism. (Recall that Archimedes—a dynamic geometer! —was more influential in his day than was his rough contemporary Euclid, whose eventually important *book* established the next 2,000 years' pursuit of geometry as a static enterprise.) In either case, the Platonic conceit is clearly situated within a larger narrative in which dynamics and motion-based reasoning are grounded, and continue to ground, mathematical experience and comprehension.

3. And Why Design?

And why talk about design? Is the art, mechanics, and practice of design in a particular area, say mathematics education technology, relevant to anyone besides the small community of practitioners working in that area? Yes. We are all users of designed objects; an awareness of, sensitivity to, and ability to think critically about design makes us better-educated consumers of the products or technologies we use. But more importantly, we are all also designers. In mathematics education, we are designers in our choices of approaches to problems. There, design influences how we assemble whatever physical or conceptual toolkit we bring to a given task, and how we orchestrate those elements (what scripts we follow, what existing skills we rehearse) in accomplishing that task. We are also designers of activities, lessons, and classes for our students, sometimes merely in the sensibilities we bring to selecting and sequencing existing curricular resources, other times more aggressively in creating our own new resources out of rawer ingredients. Finally, if or where we use mathematical technologies that are themselves authoring systems—systems where we can produce ideas rather than merely consume them—we are designers of interactive technologies ourselves. In environments like Computer Algebra Systems or Dynamic Geometry Systems, a digital object we create for ourselves, to answer our own private curiosity, is never far from one that explains that same curiosity to another, and a frequent turn in the development of one's work in an environment like *Sketchpad*, comes at the moment one asks "now how can I make this useful for someone else?" In all these cases, we are engaged in design.

And yet, effective design often resists "strategic" or "paradigmatic" thinking—solutions or design strategies that elegantly solve problems in one context often become either simplistic clichés or arcane prescriptions, when reapplied in an unrelated context. We have all seen "effective design principles" stripped of, and promoted outside, any specific application—"put the important thing first!", say, or "keep it simple, stupid!"—and we can easily summon specific problems in which these principles would be not just ineffective but even, perhaps, counterproductive. (Mathematics itself is ripe with such examples: in a deductive argument, we usually need to state the important thing *last!* And we cannot motivate introducing tools of mathematical simplification—factoring, or parentheses, or outsourcing a claim to a lemma or corollary—unless we are willing to tolerate the emergence of sufficient complexity to justify them!) So rather than spend too much effort considering abstract *principles* of design, we must consider design in application, design in specific instances and practices rather than in generalizations and abstractions. In this, getting good at design is like getting good at sport: a theory of volleyball only takes you so far. Eventually, you have to play the game.

4. Design and dynamics, in Dynamic Geometry

4.1 The origin of dragging

Now let us explore in detail an actual instance of a designed and gestural form of mathematical interactivity, dragging, as it appears in Dynamic Geometry software such as *The Geometer's Sketchpad* or *Cabri* [6]. While these environments are mathematically rich and find diverse curricular applications, the act of dragging is at their heart and was widely celebrated at their inception as their most important contribution to the evolving corpus of geometry technologies ([7], [8]). Broadly, dragging allows you physically to move one or more elements of a graphical construction or diagram, while preserving all the stated definitions that relate that element to other elements of the figure. In the single gesture of moving your computer mouse across the screen, you tour dozens if not hundreds of continuously-related examples of your construction, its fundamental invariances intact while inessential properties melt away. A diagram is no longer an *example* of a construction, it instead appears almost as the *general case*. It is a powerful tool of mathematical generalization, and since you yourself are guiding the mouse dragging the vertex through all these limitless examples, you find yourself a tremendously empowered agent of mathematical causation and generalization. Thirty years after the invention of Dynamic Geometry dragging, the idea has spread so widely through software and mathematics that it is hard to imagine the sensation of its original impact. And even in that moment, many mathematicians felt an uncanny sense of familiarity upon first encountering Dynamic Geometry or a welcome inevitability ([9]): Dynamic Geometry Software had taken a fantasy of continuously evolving geometric configurations out of the closed garden of mathematicians' daydreams and put it into the literal hands of mathematical explorers everywhere. It is almost as if Dynamic Geometry was not a *designed* idea but rather an essential or pre-existing one that software simply made "real."

In one sense, this claim of inevitability is, in fact, correct. The same powerful idea—dynamic dragging of geometric elements with responsive diagrams adjusting to that motion—emerged simultaneously and completely independently in the United States with *Sketchpad* and in France with *Cabri Géomètre*. Both programs were already well under development by their respective research clusters when they first encountered each other at a NATO Advanced Research Workshop in Grenoble in 1989, and the initial similarities of their approaches to this novel idea was breathtaking. Since the name *Cabri* is a play on the French term for "electronic notebook," even their titles were similar! Yet this is perhaps less a case of raw serendipity than it first appears. The research projects separately giving rise to these programs trace back to the mid-1980s, shortly after the 1984 debut of the Apple Macintosh. The Macintosh introduced bitmap display screens, mouse-based interactions, and graphical user interfaces to the computing public, and these were technologies that individually and in concert disrupted many paradigms of software interaction that preceded them. The Mac's novel hardware and software capabilities forced software professionals in all domains to ask: "what are the implications of graphical visualization and direct manipulation on the objects and workflows of *my* work?" *Sketchpad* and *Cabri* both asked this question, and answered it similarly.

Yet it is by no means given that something as "new" as Dynamic Geometry would arise from such reflection. In many domains—including within the reigning geometry technologies of the day (that is; of *before* the advent of the fully graphical user interface)—the answer was some form of "we shall continue as before, but now use a mouse to choose from menus and icons directing our previous workflows," in a pattern designers have come to call *skeuomorphism* , wherein new objects or technologies are intentionally designed to resemble older technologies they may seek to replace. The ornamental triglyphs of Doric stone columns in classical Greek temple architecture resemble structural features of the wooden temples they replace. While some dismiss the skeuomorphic tendency in design as a form of nostalgia, I believe instead it reflects the difficulty of envisioning what one cannot already, in some sense, see. The skeuomorph signals less the present's longing for the past than its stranglehold on potential futures, in other words. Comically, in the case of *Sketchpad*, the very earliest design motivation for draggable configurations had nothing to do with mathematical exploration or geometric generalization and was instead itself an instance of skeuomorphism. I originally imagined the main purpose of the software was to enable mathematicians to produce print illustrations—perhaps for publication in print journals! In that context, the ability to rearrange their constructions on-screen to fit the available page space in print would be welcome compared to an alternative of rebuilding them, repeatedly, to meet the layout requirements. It was only through a process of considering consequences, and through experiment with some of the first actual prototypes of moveable—dynamic—geometry, that its profound mathematical novelty, and its vast potential impact on visualization and learning, became apparent.

In this preliminary account we gain insight into an essential characteristic of designs' relationship to the domains in which they appear. In the case of software design, the affordances of available hardware not only dictate the limits of what is *possible* in software (an obvious point), but also what is *conceivable* in software (a less obvious one). Thus at any moment within a domain where software finds application, the affordances of contemporary hardware assert an influence not just on our thinking about software but on our thinking about the domain itself. In the 1960s, pre-graphical computers acted entirely as symbol processors; hardware supported text-based input and text-based output. The quest of geometry software in that era—and therefore, of considerable work both in geometry and in computer science—was to automate deduction, a task which conceives of geometric reasoning as a symbol-processing activity. In the 1970s and early 1980s, hardware developed limited support for graphical output but remained symbol-based (keyboard driven) in their input register. The geometry software of that era—Logo ruled the day—involved students' authoring symbolic procedures that produced entertaining or attractive images. Here geometry was acknowledged as pertaining to the structure of space or the plane—the graphical output was relevant—but reasoning and problem-solving was still propositional and linguistic. With the debut of the Macintosh, hardware evolved to feature a ubiquitous bitmapped screen (an always-available 2D graphical output, which the metaphor of "scrolling" makes infinite) and a ubiquitous mouse (an always-available 2D graphical input, which the action of continuous "rolling" makes infinite). It is hard to conceive a better set of physical affordances on which to model the mathematical idea of an unbounded two-dimensional Euclidean plane! And thus in that moment Dynamic Geometry became possible, and the discourse of contemporary school geometry shifted again to emphasize construction and visualization of planar objects rather than their symbolic manipulation. Uncritically, we might like to think of mathematics as Platonically ideal, as impervious to transitory issues of representation or affordance; but in transitions such as these, we see it is constructed by them.

4.2 Deterministic designs for dynamic indeterminacies

Even if the *possibility* of Dynamic Geometry is a direct implication of the emergent computer hardware of the 1980s, its inevitability or mathematical obviousness is, in fact, an illusion. On close inspection, the manipulation of even simple geometric diagrams in space and time opens numerous mathematically unresolvable questions. Thus the "inevitability" of the consequences of dragging owes as much to design as to deduction or to the certainties of mathematical definition—of *static* mathematical definition. Let us briefly consider several examples of situations where dynamics needed to be newly designed for, rather than was defined by pre-existing, geometric purposes, in order to survey the variety of considerations contributing to such design.

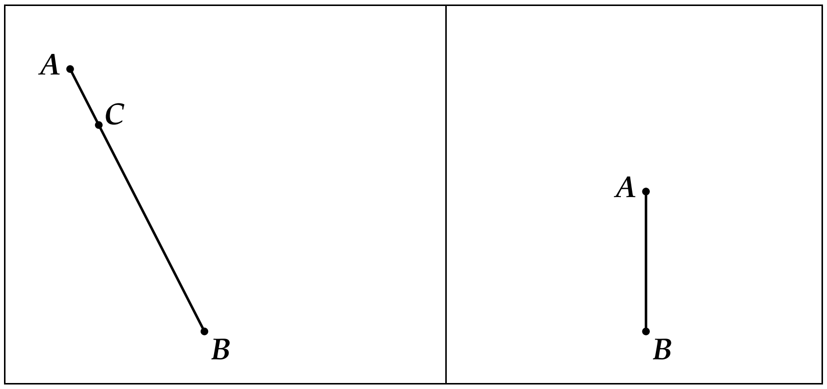


Figure 1. Membership indeterminacy: as *A* moves where is *C*?

In the simple example of Figure 1 (left), point *C* is defined as a given point on segment *AB*. The user drags *A* closer to *B*, resulting in *A*, *B*, and the segment *AB* as shown on the right. Where should *C* appear in this segment? Should it maintain its original distance from *A*, though the segment is now shorter? Or is its original distance from *B*? Since all we know is that *C* is defined *somewhere* on *AB*, does that mean *anywhere*? Should it move to a random location? Or perhaps should it stick as close as possible to its original planar position, i.e. to *C* (left), minimizing the distance it itself moves?

Sketchpad, *Cabri*, and every other Dynamic Geometry tool I know "answer" this question identically: under changes to *AB*, *C* maintains a constant ratio of division of *AB*. Thus since *C* (left) is roughly 30% of the distance from *A* to *B*, we can locate *C* (right) similarly. This design has the elegant effect of moving *C* in linear proportion to the overall motion of either endpoint and thus maintains a conceptual symmetry between endpoints while avoiding the stochastic chaos of *C* hopping about at random even when the segment is minimally perturbed. And yet, by preserving a constant ratio across dynamic manipulation, the solution introduces a mathematical artifact—a barycentric invariance—to the generalization that is absent from the construction's stated definition. We expect students to be staggered by facts like the concurrence of a triangle's medians in a single point, but in Dynamic Geometry this is unremarkable: dynamically preserved proportions guarantee *any* three cevians of a triangle that *ever* intersect in a point will *always* intersect in a point, no matter how we deform the triangle.

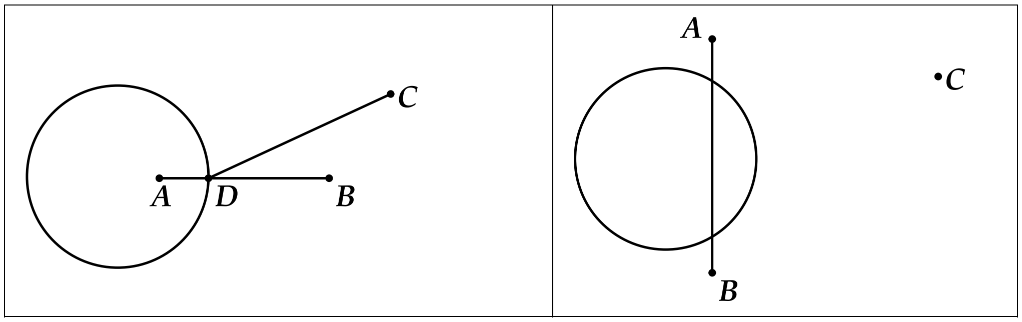


Figure 2. Quadratic indeterminacy: as *AB* (left) moves, where is *D* (right)?

In the static configuration of Figure 2 (left), a circle is intersected by a given segment *AB*; and a second segment, from *C*, connects to this intersection *D*. Now we drag points *A* and *B* into the configuration at right, leaving the circle and *C* unchanged. *AB* now intersects the circle twice rather than once. Where should *D* (and therefore segment *CD*) be located in this new configuration?

Mathematically a system of one solution has evolved into a system of two, with no clear mapping between the configurations in Euclidean geometry. To provide a definition of motion that allows *CD* to reappear in the righthand configuration, software developers must augment Euclidean geometry with a dynamic theory and behavior for quadratic intersections, and again balance aesthetic and functional considerations with concern for mathematical artifacts such designs will introduce in naïve exploration.

Here there is less agreement among software designers about what solution might be most fitting to the problem's constraints. Historically *Sketchpad* and *Cabri* both adopt solutions that begin by treating the segment *AB* as intrinsically directed, thus enabling a dynamic discrimination between the two potential intersections. However, they differ in their handling of many of the special cases that arise with finite segments, or with arcs that can effectively "invert" their orientations—and therefore the identity of their quadratic roots. Unhappily, in both programs, it is possible to cause a point to hop from one of two possible circle intersections to the opposite, under only a subtle motion of the configuration's independent points, a discontinuity of output despite a continuous transformation of input. The third major Dynamic Geometry software package, *Cinderella* [11], took issue with this design, and introduced a new solution based on a sophisticated internal mathematical architecture that guaranteed continuity in such transformations. Unfortunately, subsequent research [12] proved that this gain simultaneously eliminated determinism—the highly desirable property that the position of a construction's independent objects fully determines the position of its dependent ones. More recent Dynamic Geometry implementations (such as *GeoGebra* or *Desmos*) have returned to the designs first proposed for *Sketchpad* and *Cabri*, while sustaining numerous local exceptions and variations.

Numerous other situations arise in which the designer is forced to choose between two or more equally plausible mathematical outcomes of a certain dynamic interaction, including ones less narrowly divergent than the choice of "plus or minus a square root" arising between the roots of a quadratic. Consider two points *A* and *A'* related by geometric reflection through a mirror line *l*. In most if not all Dynamic Geometry software, dragging point *A* causes *A'* to move in mirror image. But an equally viable interpretation would move line *l* instead, maintaining a position bisecting the moving *A* and the fixed *A′*. Here a designer's choice (assuming a designer considers both options!) is more subtly pedagogical: should reflection dynamics emphasize the "equal but opposite" motions of mirroring—an emphasis on isometrics and chirality—or instead the "perpendicular bisector" role of the mirror, of potentially greater value to construction and proof?

Perhaps the broadest category of designed "inventions" of mathematical behavior in Dynamic Geometry involves situations where users attempt to drag some dependent object—an object that is constructed rather than independent or given. There is no precedent and even no allowance for such an operation in a geometry of strict deduction. To adhere narrowly to forward inferences reasoning only from givens to conclusions would outlaw such ideas categorically. The earliest versions of *Cabri* enforced such stricture. But in designing early *Sketchpad*, I remarked that in an environment in which different graphical objects of similar type (e.g. different segments) had similar visual identity and similar mouse-based accessibility, whether they were independent or dependent objects, users (including myself) would often *want* to drag such objects, and would often *try* to drag them. An important early design conception I held of *Sketchpad* was it should function as a tool abetting users' pursuit of their own goals rather than either as an authority such as a teacher establishing external goals or a censor didactically prohibiting them. An error message—"dragging such an object is outlawed by the very precepts of deduction itself!"—seemed ill-advised in this situation and unlike the behavior of any physical tool with which I was familiar. So from its earliest version *Sketchpad* has attempted to support users' desire to drag dependent objects, and to assert consistent and predictable mathematical behaviors to such actions.

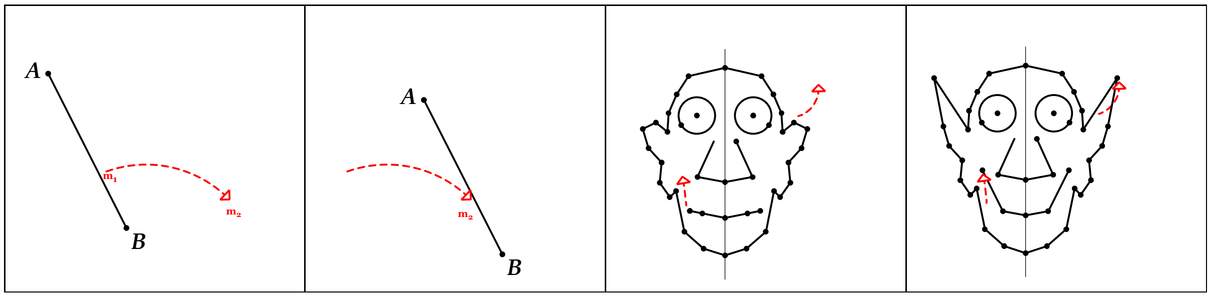
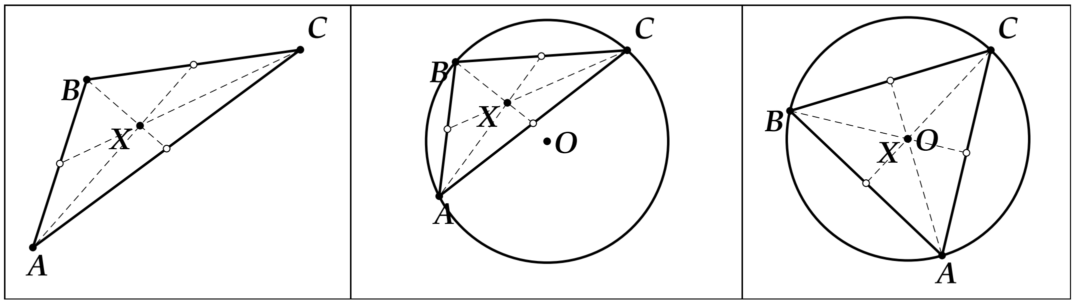


Figure 3. Base cases of "reverse dragging"

Figure 3 illustrates two extremely simple applications of reverse dragging. In the before/after sequence at left, a user drags a segment to the right. More precisely, a segment *AB*—an object that is itself dependent on the location of independent points *A* and *B*—is dragged by a mouse gesture from plane location *m1* to *m2*. (In this notation, *m1* and *m2* are not points defined in the construction; they represent only the extremes of a mouse motion.) In a deductively strict interpretation, the situation is under-constrained: an infinite number of potential segments pass through *m2*; the segment could reasonably evolve into any of these. Even introducing the same proportionate invariance as we introduced in Figure 1—with *m2* diving the new segment in the same ratio *m1* divided the old segment—fails to limit usefully the solution space. Instead, *Sketchpad* adopts the (I hope) obvious solution of attempting to preserve the length and orientation of the dragged segment, yielding the parallel translation of *AB* (shown in Figure 3 left, *after*). The important point here is that the justification for such a design draws from any of a number of sources—physical plausibility (in suggesting the segment maintains some "inertial" length or orientation, where possible), or from embracing a hypothetical "tool nature" (tools don't give error messages), or even from imagined user convenience (perhaps the user *wanted* to move the existing segment!)—rather than from any prescriptive mathematics preceding dynamic interactivity!

In the sequence in Figure 3 right, the image is symmetric with respect to a vertical mirror of reflection. Moving elements on either side of the mirror causes the opposite image to move opposite, regardless of which of the pair is technically an "independent" object. Even though one side necessarily preceded the other in the construction sequence behind the original configuration, the designed dynamics emphasize the symmetry of the present transformation over the dependency of its historical construction.

These cases of "reverse dragging" are no more arbitrary—or rather, no more dependent on new innovation and productive design—than the earlier discussed figurations, but they appear to be more controversial among practitioners. *Cinderella* joined *Cabri* in adopting a rigid, forward-only propagation of dependency. Then subsequent versions of *Cabri* relented, acknowledging the tremendous convenience users found in being able to reposition a segment or a circle by dragging that object directly, rather than by dragging its defining points. Then *GeoGebra* copied *Cabri*'s model. But both suffer from a somewhat capricious limitation: reverse dragging works only on objects that themselves depend immediately from independent points, yet other similar objects remain irreversible. Attempting to drag them has no effect. This gives rise to an unfortunate situation in which the very draggability of an object no longer depends on that object's fundamental definition (a segment defined through two points, say), but instead on non-obvious conditions imposed on the its ancestry. (To drag a segment through two points it is not sufficient, in *GeoGebra* for example, for its parental points to be draggable; they must instead be free.)



By contrast, *Sketchpad*—and more recently, *Desmos*—deploy reverse draggability as a generalized premise of Dynamic Geometry, as something consistently available rather than limited to special expedient circumstances. Dragging any constructed element attempts to relocate its determining geometry in such a way that the element remains under the mouse over the course of the motion. This is of course a heuristically-defined search, since—just as in the forward-dragging cases listed above—multiple and diverse solutions exist. From an analytic perspective we are now asking neither "what is the solution to this set of equations?" nor "which of the multiple solutions to this system of equations is preferrable?" but rather "which system of equations might produce this desired solution?" It is an entirely different approach to problem posing, and occasionally the system yields startling insight. In the example of Figure 4, left, a user begins with construction of medians of a triangle, that concurs in a point (the centroid *X*). Knowing the circumcenter of a triangle is a similar point of concurrence (of side bisectors), she might wonder "is it possible for the centroid and circumcenter to be the same point?" Drawing a circle and reinscribing her triangle inside it allows her to compare the triangle's centroid to its circumcenter *O* (Figure 4, middle). She then drags the centroid—a highly dependent object, a point of concurrence of three medians, which are themselves segments connecting points on the circle (the reinscribed triangle's vertices) to midpoints of the sides opposite those vertices—and moves it toward the center of the circle. The triangle wriggles and stretches into a position (Figure 4, right) where *X* and *O* coincide at just the moment that—ah ha!—the triangle becomes equilateral. Such provocations to insight cannot occur for the strict axiomatician, for whom reverse dragging is geometric heresy, but are very in keeping with the inductive and exploratory approaches that were encouraged as complements to strictly deductive enterprise by the mathematics reform movements of the 1980s.

Reflection

Collectively, this case history of design situations demonstrates that when the surface of traditional mathematical practice is wrapped taut across a new representational medium, fissures or tears or holes open up in which prior mathematics is insufficient to describe the new shape. These can be viewed as inadequacies of the novel medium or—equally—as opportunities to ask and answer new mathematical questions. Thus every new representation both opens and closes doors to potential forms of activity. When we survey how new questions are answered and how such answers are ratified, we see a variety of mechanics at play. Sometimes even where a question has never been previously asked, a solution appears so obviously sufficient there is little interest in questioning it further (the point-on-segment example). In other situations, an issue appears ultimately resolved by reasoned consensus emerging from a community's grappling over time with various alternatives (the quadratic intersection example). Elsewhere, a variety of equivalent solutions seem to vary largely on sociocultural lines, reflecting pedagogic norms for instance that may vary place to place, and in these moments we may sustain multiple viable solutions although perhaps only one in one place at one time. (For example: should a constructed symmetry better appear as a construction or as a symmetry, in situations where both cannot be emphasized equally?). Even where a solution arises as the result of one individual's singular or quixotic vision, often such vision appears consonant with developments in surrounding socio-material and sociocultural milieux (as in the origin of Dynamic Geometry in the dawn of the modern graphical user interface; or in the relation of reverse dragging to the "inductive approaches" encouraged by the 1989 *NCTM Standards* [13]). These varied mechanics demonstrate not just how a novel tool arises within a social context but how a community of practice adopts and empowers a new representation.

It is hubris to suggest that a handful of software programs, or even an entire genre of them, form some sort of epochal advance in the long evolution of mathematical representation or practice. Software goes obsolete within the lifetime of its authors and compared to the durability of journal publication (to say nothing of the durability of *The Elements*!), is as fragile as Archimedes' drawings in the sand. And yet, the design and social ratification of Dynamic Geometry representations stands as a small example of an ongoing and much larger revolution in representation. The rise of propositional and symbolic mathematics, in the era of Plato and Euclid, corresponds historically to the rise of pervasive writing and the logocentric technologies of print. 2500 years later, print—with its relatively fixed symbol catalog and its physical archive in the bookshelves and libraries of the world—is being rapidly and systemically replaced by computational representations. While computers excel at symbol manipulation—and therefore will extend and further symbolic mathematics—it is perhaps a skeuomorphic error to think of them only as symbol processors; they are electricity processors. While we may code their currents and voltages into symbolic zeroes and ones, we then deploy those symbols to depict graphical diagrams and temporal motions, and to capture gesturing hands, probing fingers, and rolling mice. These non-symbolic infrastructures point the way to a post-symbolic mathematics, of which Dynamic Geometry is perhaps an early suggestion. As in biological evolution there is no goal or even clear direction, only a mechanism of change. Ultimately, what we mean by mathematics is a function of what we do, when we do mathematics, and of what we think, when we think mathematically. Both these in turn are constituted by the representations, technologies, infrastructure and affordances of the moment. When we look closely at any one of them, such representations or technologies reveal themselves to be caught, in that moment—as in every moment—between the nostalgia for a vanished past, and the uncertain vision of many potential futures.

References

[1]. Jackiw, N. (1991, first version) *The Geometer's Sketchpad* (computer Software). Key Curriculum Press.

[2]. Balacheff, Nicolas (1988) Une étude des processus de preuve en mathématique chez des élèves de collège. Modélisation et simulation. Institut National Polytechnique de Grenoble - INPG; Université Joseph-Fourier - Grenoble I.

[3]. Seidenberg, A. (1961) The ritual origin of geometry. *Arch. Hist. Exact Sci.* 1, 488–527 (1961). https://doi.org/10.1007/BF00327767.

[4]. Seidenberg, A. (1962) The ritual origin of counting. *Arch. Hist. Exact. Sci*. 2, 1–40 (1962). https://doi.org/10.1007/BF00325159.

[5]. Lakoff, G., & Núñez, R. E. (2000). *Where mathematics comes from: How the embodied mind brings mathematics into being.* Basic Books.

[6]. Laborde, J.-M. et al. (1990, first version) *Cabri Géomètre* (computer software), LSD-IMAG, Université Joseph-Fourier - Grenoble.

[7]. Finzer, W. and Bennett, Dan. (1995) From Drawing to Construction with The Geometer's Sketchpad. *The Mathematics Teacher*, 88.5 (May).

[8]. Hoyles, C. and Noss, R. (1994) Dynamic Geometry Environments: What's the Point? *The Mathematics Teacher,* 87.9 (December). https://doi.org/10.5951/MT.87.9.0716.

[9]. Hofstadter, Douglas R. Discovery and Dissection of a Geometric Gem. In James R. King and Doris Schattschneider (eds.), *Geometry Turned On!: Dynamic Software in Learning, Teaching, and Research*. Washington, D.C.: The Mathematical Association of America (1997): 3–14.

[10]. Norman, Don (2013). *The Design of Everyday Things: Revised & Expanded Edition*. Basic Books. ISBN 978-0-465-05065-9.

[11]. Kortenkamp, U. and Richter-Gebert, J. (2000, first version) *Cinderella* (computer software). Springer.

[12]. Gawlick, T. (2001). Dynamic Notions for Dynamic Geometry. In Borovcnik and Kautschitsch (eds.), *Fifth International Conference on Technology in Mathematics Teaching,* Klagenfurt.

[13]. National Council of Teachers of Mathematics (1989). *Curriculum and Evaluation Standards for School Mathematics*. Reston, Virginia.