Computer-Aided Design 42 (2010) 151–161

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

A constraint-based dynamic geometry system

Marc Freixas, Robert Joan-Arinyo *, Antoni Soto-Riera

Grup d'Informàtica a l'Enginyeria, Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain

ARTICLE INFO

Article history:
Received 25 June 2008
Accepted 20 February 2009

Keywords:
Dynamic geometry
Geometric constraint solving
Linkages
Mechanisms

ABSTRACT

Dynamic geometry systems are tools for geometric visualization. They allow the user to define geometric elements, establish relationships between them and explore the dynamic behavior of the remaining geometric elements when one of them is moved. The main problem in dynamic geometry systems is the ambiguity that arises from operations that lead to more than one possible solution. Most dynamic geometry systems deal with this problem in such a way that the solution selection method leads to a fixed dynamic behavior of the system. This is specially annoying when the behavior observed is not the one the user intended.

In this work we propose a modular architecture for dynamic geometry systems built upon a set of functional units which will allow us to apply some well-known results from the Geometric Constraint Solving field. A functional unit called *filter* will provide the user with tools to unambiguously capture the expected dynamic behavior of a given geometric problem.

1. Introduction

Dynamic geometry is a discipline that appeared during the 80s as a new tool in geometry. A number of software systems were designed for teaching geometry in secondary schools where the ruler and compass were replaced by computers featuring high resolution color screens for user–computer interaction. The key concept in dynamic geometry is *interaction*, that is, select a geometric object in the screen, move it and see immediately how the geometric construction changes.

Compared to ruler-and-compass drawings on the paper, dynamic geometry systems offer two clear advantages. First they provide tools to create accurate drawings (intersection points, tangencies, etc). Second the computer can record the way the user constructed the geometric elements that allows it to quickly rebuild the construction every time the user changes the values assigned to some parameters. As a consequence, exploration and experimentation are encouraged by showing to the user which parts of the construction change and which remain unchanged.

As far as we know, software applications related to dynamic geometry trace back to 1985 when the first version of Juno, a constraint-based graphic editor for user interfaces was reported [1]. More recently a specific version devoted to dynamic geometry, called Juno-2, has been released [2]. Among the dynamic geometry systems reported as research or commercial packages we find Cabri Géomètre [3,4], Geometers Sketchpad [5], GéoSpécif [6–8], and UniGéom [9,10]. Specific details on each system have been collected by Winroth in [11].

When using most of these dynamic geometry systems, the user can find that strange things happen from time to time. An object might suddenly jump into a different position or disappear completely or a group of objects may converge to the same position. It turns out that these effects are caused by a number of nontrivial mathematical issues that must be addressed.

Basically, the problems are caused by ambiguities. One source of ambiguity is the fact that, in general, geometric operations have more than one solution, for example, intersecting a line and a circle. Another ambiguity appears when a problem with a well-defined solution whenever geometric elements are in general position, say computing the point where two straight lines intersect, reaches a degenerate configuration, for example, the straight lines became parallel. While the user is defining the construction, he or she is responsible to resolve these ambiguities. Usually, the user selects interactively the intended solution among those solution instances offered by the system. However, when the user is dragging a geometric element, the system is responsible to choose the intended solution, the same solution that the user would select if we could ask him or her again. Therefore the core problem of dynamic geometry is to find a well-defined method for handling ambiguities while some parameters of a construction are changed continuously.

In order to work out a satisfactory solution for these problems, an ideal dynamic geometry system should exhibit three properties: *determinism*, *conservatism*, and *continuity* [12]. Determinism is

* Corresponding author. Tel.: +34 93 401 66 69; fax: +34 93 401 60 50.
E-mail address: robert@lsi.upc.edu (R. Joan-Arinyo).

0010-4485/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2009.02.016

the property by which, for a given problem and parameters assignment, there is at most one instance of the construction. Clearly, systems considering point and lines with construction steps where operations with circles are not involved have determinism. If circles and, in general, conics are involved the system is, in principle, no longer deterministic because, for example, a straight line and a circle intersect in up to two different points.

A dynamic geometry system is conservative if when you move geometric elements and then undo all your moves by reversing them, then you get the same geometric construction. Notice that if a dynamic system is deterministic, you always have the same solution instance for the same problem and parameters assignment, therefore it is also conservative. If the system is nondeterministic we might not expect to be conservative.

In our context, defining continuity is rather elusive and gives rise to challenging issues. We say that a system is continuous if small changes in parameters assignment result in small changes in the placements of the geometric elements in the problem. That is, we do not want to have large jumps of geometric elements for small changes in the parameters assignment values. Unfortunately, as shown by Kortenkamp [12], as soon as a dynamic geometry system includes constructions with more than one possible solution, it is not possible to have determinism and continuity or conservatism and continuity.

Systems based on a more deeper understanding of the mathematics behind dynamic geometry which solve some ambiguities are Projective Drawing Board, developed by Winroth [11], and Cinderella developed by Richter-Gebert and Kortemkamp [12,13]. Cinderella 2, an enhanced version of Cinderella has been lately launched [14]. A remarkable accomplishment in the open source software field is GeoGebra [15]. The problem now is that the solution selection method implemented in these systems leads to a fixed dynamic behavior of the system. This is specially annoying when this behavior is not the one the user intended.

According to Allen et al. [16] geometric constraint solving plays a central role in dynamic geometry systems. As we will see, this is the case for our system. Among the geometric constraint solving techniques reported in the literature, we are interested in the one known as *constructive*.

Constructive geometric constraint solving provides tools specifically well suited to deal with dynamic geometry systems requirements: they record the way the geometric construction was defined, called *construction plan*, and select one solution instance among those possible through the definition of an index that uniquely identifies it. Moreover, by using solvers that handle equational constraints, see for example [17–20], relationships between two geometric objects which play a paramount role in dynamic geometry, for example *equal distance*, can be naturally captured.

In this paper we report on a modular dynamic geometric system built on top of a constructive variational geometric constraint solver which provides tools for efficiently capturing the dynamic behavior of a geometric construction. We assume that the equations underlying the set of geometric constraints have at most degree two. Given a construction plan for the problem at hand, the system first extracts a graph of candidate dynamic transitions. Then by pruning this graph according to a set of transition rules, the system yields a new graph which captures the specific intended dynamic behavior. Continuity is achieved by requiring at each transition both continuity in the driving parameter values and continuity in the placement of geometric elements.

The rest of the paper is organized as follows. Section 2 recalls the basics of constructive geometric constraint solving used in our work. Section 3 is devoted to present our dynamic geometry system. A preliminary implementation and the results yielded are reported in Section 4. Finally we offer a brief summary and some directions for future work in Section 5.

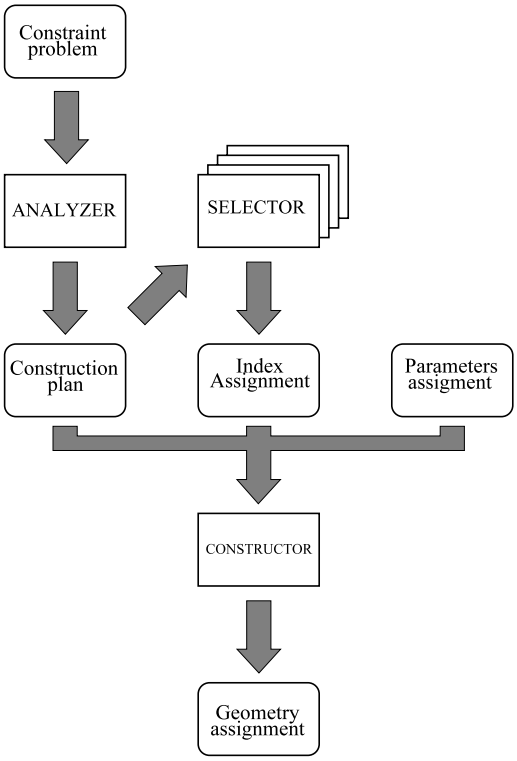

Fig. 1. An architecture for constructive solvers.

2. Constructive geometric constraint solving

Geometric constraint solving is an arguable core technology of computer aided design and, by extension, geometric constraint solving is also applicable in virtual reality and is closely related in a technical sense to geometric theorem proving. For solution techniques, geometric constraint solving also borrows heavily from symbolic algebraic computation and matroid theory.

Many techniques have been reported in the literature that provide powerful and efficient methods for solving systems of geometric constraints. For a review, see Hoffmann et al. [21]. Among all the geometric constraint solving techniques, our interest here focuses on the one known as *constructive*. See [22–25,18,19,26–29] for an in-depth discussion on this topic.

Constructive geometric constraint solving is a technique widely used in the geometric constraint solving field. Since we build our dynamic geometry system on top of a constructive solver, we briefly recall here the main features underlying this technique. An architecture for constructive solvers is illustrated in Fig. 1 where square boxes are *functional units* and rounded boxes are *data entities*. The functional units are the analyzer, the index selector and the constructor. The data entities are the geometric constraint problem, the construction plan, the parameters assignment, the index assignment and the geometry assignment. The geometric constraint problem and the parameters assignment, are supplied by the user. The other entities are computed by the system.

In this technology, the user defines a *geometric constraint problem* by sketching some *geometric elements* taken from a given repertoire (points, lines, circles, etc) and annotating the sketch with a set of geometric relationships, called *constraints*, (point–point distance, point–line distance, angle between two lines and so on), that must be fulfilled.

Fig. 2(a) illustrates a piston-connecting rod-crankshaft example and Fig. 2(b) is an abstraction represented as a geometric constraint problem. The set of geometric elements includes four points $\{p_0, p_1, p_2, p_3\}$ and a straight line $\{l\}$. The set of constraints includes