Abstract

Geometry

M

constructions can be used

This paper presents an object-oriented approach to interactive modelling of

to numerical methods. The prototype GEObject is implemented under Actor
in Windows 3.0 environment.

graphics

I 1ntroducti

when created and displayed on a screen, and only their pixel components are still available to users. What is therefore needed is a "geometry expert environment" that

Indeed, in traditional computer graphics libraries or graphics systems, which are not usually structured on their knowledge domain, a geometric entity is mainly modelled onits vi

considering its structure and geometrical p
visualised as a series of pixels of variable length on the screen, while a graphics library can offer a routine to trace a line given its end-points co-ordinates, which represents the

having a mid-point. To construct a segment with such properties one has to explicitly write a computer program, in a given language. This is also true in many object-oriented design a geometric modelling system that can increase its capabilities adding new

[illegible]

the choice of an object-oriented environment allows to encapsulate geometric knowledge
is the prototype GEObject, an object-oriented system, that is described in the following:
expected geometrical properties.

A "geometry expert environment" could be the basis of any interactive graphics system

needs increase they can extend system capabilities, easing system tailoring or

i n

Given the highly constructive character of Euclidean Geometry, it seems natural to

customisation.

Constructive geometry

postul

lines and a compass to draw circles. In Euclid's postulates we read:

1. it is possible to draw a straight line from one point to another;
2. given a segment it is possible to produce a straight line;

Complex geometric figures can be modelled using the known procedures of elementary plane geometry, e.g. we can impose a relationship of congruence, parallelism or orthogonality among lines, or we can impose intersection among lines or circles in order to obtain points that are meaningful in a given geometric construction. These procedures can all be implemented by using a compass and a ruler.

N e

compass: the duplication of the cube, the trisection of the angle and the quadrature

circle. These problems cannot be solved with these tools, as it was proved later on, after

angle, for instance, can be trisected with a marked ruler, that is a ruler on whose edge one can mark lengths.

must be satisfied when a figure is transformed.

A case study for constraint satisfaction

constructions. Euclid in his Elements stated five postulates that regulate the geometrical

a quadrilateral is a parallelogram. Alan Borning [Borning 1986] uses this theorem to

systems, like our.

[illegible] vertices of

pseudo

of the theorem in different configurations by moving any vertex of the quadrilateral.

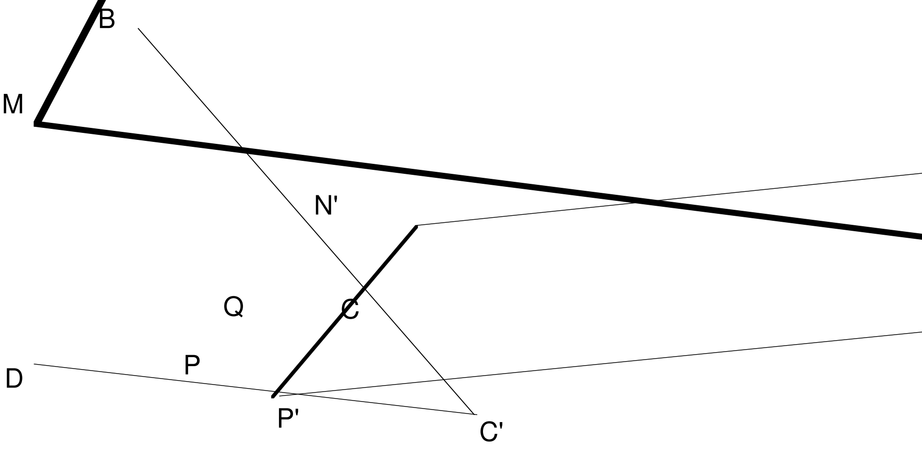
ThingLab provides this kind of flexibility and interactivity, because the construction of

SuperClass

```
                aLine
         midpoint: aPoint

                                                       /* rule
         midpoint <- (linepoint1 + linepoint2) / 2      /* method 1
         linepoint1 <- linepoint2 + ((midpoint - linepoint2 )*2)
                                                       /* method 2
         linepoint2 <- linepoint1 + ((midpoint - linepoint1)*2)
                                                       /* method 3
```

and its interpretative model. A geometric problem is thereafter solved applying algebraic methods.



This declarative approach requires to explicitly state the properties and relationships that hold among the geometric elements of the figure. In fact the mid-points of the four sides

ordinates of the end-points of a segment.

The alternative procedural approach is based only on geometric constructions, grounded on geometric knowledge. A geometric construction problem is a problem of drawing a figure satisfying given conditions, by using some prescribed tools, only a

procedure to build a figure satisfying the problem, even though the solution exists; or inconsistent, if it does not exist a figure satisfying the given constraints.

finite number of times. The problem can be: possible; impossible, if it can not be found a

The procedural message. Construction of mid-point by its axis done by recalling geometrical methods

Only the final depiction on paper or screen requires an algebraic interpretation, in order to compute its co-ordinates in a given Cartesian system. The inscribed parallelogram of the quadrilateral theorem can then be constructed applying repeatedly the above mid-point

The procedural approach allows us to construct a geometric figure in a natural and

characterisation allows to mimic the usual compass-and-ruler technique of elementary Euclidean geometry, and it allows to satisfy constraints during geometric manipulation without recurring to a numerical general constraint solver.

primitives:

¥ drawing a segment given two points;

only: any point generated by intersecting circles and segments is a new geometric entity.

¥ o

¥ getting points intersecting couples of segments, couples of circles or a

intersection of two given lines; using the compass, on the other hand, we can

procedure.

two given circles.

intuitive way, and to define implicitly geometric constraints by specifying the

The object-oriented approach to geometric knowledge representation allows to define

relationships among geometric entities during the construction. Procedural geometric

closed polygon (genus-species hierarchy) and a point is part of a segment, which, in turn

hierarchical relationships.

A geometric entity can be considered as an instance of its geometric class; classes encapsulate, as methods, their behaviour in response to messages sent by other entities or

to instantiate an object a class method must be invoked, while the specific geometric behaviour of an object is ruled by instance methods. To apply a geometric transformation

[illegible]

proper creation messages, the class methods. Instance methods are then invoked, in the constraints.

tokens, and the effect of " " places.

(Jensen, 139 l-ruler constructions. Places

can be app geometric objects. Places are represe

geometric en in the corresponding

type of geo er ones, and about the co-

the instantiation of points P and Q gives an initial marking for the net and allows to fire

the segment PQ. Points P and Q are still available for further manipulation, because the construction has not cancelled them; therefore their places are still marked. A net can be considered as a configuration, the initial marking, can be interpreted as the actual drawing of a segment given a couple of points.

1394

P

Fig. 4. The mid-point construction net

With Petri net notation we now represent the construction of the quadrilateral problem. We need first to show, Fig.4 [2], the net of a sub problem - the classical mid-point construction - where we define by example the construction method, *mid_pt*.

---

[2] From now on, in order to simplify the graph depiction, we shall not display arcs returning tokens to input places of transitions; moreover, in the following graphs, we use other construction methods: *line_2p*, to return a line given two points; *circle_c_rad*, to build a circle given a centre and a radius; *intersect*, to return intersection points between entities.

We can use the new *mid_pt* method in any further construction, as in the net of Fig.5,

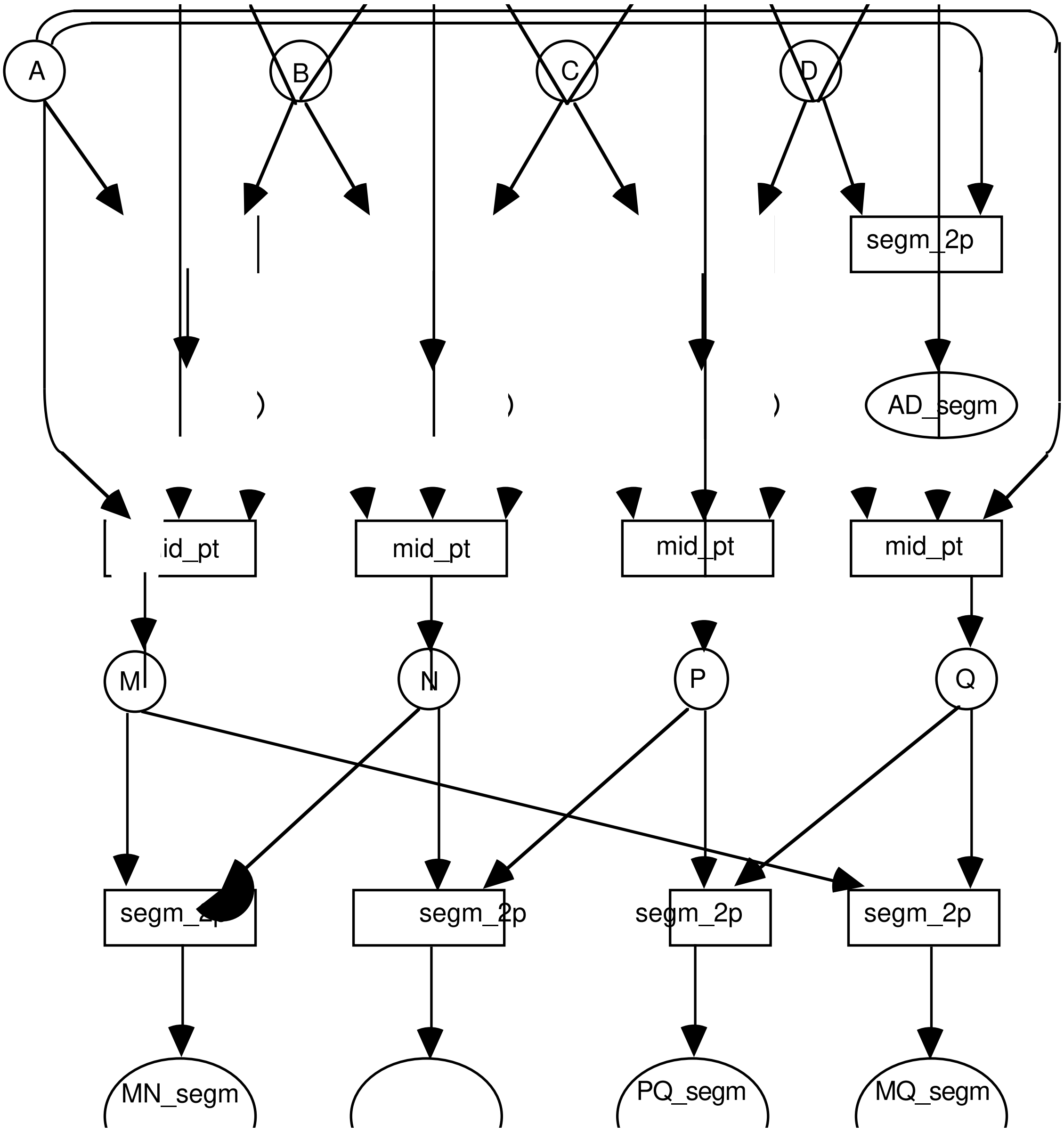


Fig. 5. The geometric construction of the quadrilateral theorem

From these examples we see how the Petri net represents the process of building a geometric construction showing the relationships among the involved geometric entities. Such relationships are indeed geometric constraints, that are not expressed in term of algebraic equations, but only in terms of pure geometric properties that entities must satisfy. Characterising geometric constraints in this way avoids recurring to non geometric tools. Petri nets ease the task of recognising constrained geometric entities as those whose corresponding places are produced by some constructive method. On the opposite primitive points of the construction are represented by places with no input arcs, and they are the only unconstrained points.

2.4 Transformations of geometric constructions

After we have built a geometric figure, it is important to be able to transform it, maintaining its constraints; and the problem is to know which transformations are allowed, because not all of them are compatible with the constraints. The transformations that we consider are rigid roto-translations applied to some of the points involved in the construction. A transformation applied to all the figure points trivially satisfy the constraints, while a transformation of sub-parts may not be allowed by the given constraints. Generally the constraints that can be solved in a given geometric figure depend on the chosen construction.

A Petri net of a geometric construction represents the class of all the figures that satisfy the constraints imposed during the construction; and any figure, which is a legitimate transformation of a given one, is still represented by the same network. What discriminates two such figures is only the colour of the initial markings of their unique net. The colour of a token holds the type of entity and the plane geometric co-ordinates of its constituent points. So changing the marking of places of primitive points corresponds to transforming their geometric co-ordinates; this new marking fires again the net generating the transformed figure. Therefore changing initial markings in a Petri net describe a geometric transformation applied to primitive points. For instance, back to the parallelogram theorem, if we change the token of place C in Fig.5 to implement the transformation of Fig.1, the net evolves to generate a new figure: quadrilateral ABC'D, and its inscribed parallelogram MN'P'Q.

3 GEObject

In GEObject we designed primitive classes corresponding to primitive entities and methods to instantiate primitive objects: *GeoPoint*, *GeoLine*, *GeoCircle* and *GeoAngle*. We structured the class *GeoLine* into sub-classes, to implement a genus-species hierarchy of subclasses, namely *GeoHalfLine* and *GeoSegment*, which will inherit the properties of their super-class *GeoLine*. The same can be said for class *GeoCircle*. *GeoArc*.

In a class there might be more than one class method: to instantiate a line we have defined one class method 'given two points', which is inherited by half-lines and segments, whereas to instantiate an object of class *GeoCircle* we have defined two class methods 'given a centre and a point' or 'given a centre and the radius'.

The instance method to compute intersection between lines and/or circles is the most important method in GEObject, because any new non primitive point can be generated only by intersecting objects. Intersection is implemented as a specific polymorphic instance method for classes *GeoLine* and *GeoCircle*. Instance methods are inherited by sub-classes, which have also their own specific methods. For example, the method *intersection* is inherited by subclass *GeoHalfLine* and its subclass *GeoSegment*. Subclass *GeoSegment*, in turn, has also its own specific methods, such as: *length*. Usually during a geometric construction, we impose more complex relationships, such as congruence, parallelism or orthogonality among lines and tangency with circles; these relationships are themselves the result of a construction.

The primitive object *GeoPoint* plays a special role, because any other object, even a circle, which is primitive, requires the instantiation of at least a point, i.e. its centre. This fact emphasises also the existence in GEObject of a part-of hierarchy: the two end-points of a segment are part of it; a segment in turn can be a part of a circle, because to define it we can give a point, its centre, and a segment, its radius. A special class in the genus-species hierarchy is *GeoLink*, that allows to define new classes constructively, e.g. to define a hierarchy of polygons, e.g. quadrilaterals, squares. Moreover new instance methods can be defined by constructions. Any class, primitive or newly defined ones,

---

[3] GEObject is implemented in Actor under the window manager Windows 3.0. Actor is a Trademark of The Whitewater Group and Windows is a Trademark of Microsoft Corporation.

can be enriched by appropriate new class and instance methods. These two aspects of the

and flexibility of object-oriented programming.

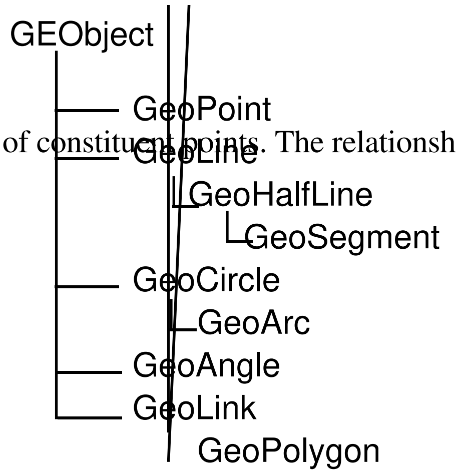


Fig. 6. Genus-Species Hierarchy of GEObject classes

A construction process creates a network of relationships among objects, that corresponds to the Petri net representation of the construction, where tokens hold the co-ordinates of the instances of constituent points. The relationship network induces a partial

ordering among instantiated objects; we call ancestor/descendent two objects for which holds the ordering relation. Ancestor/descendent relationship corresponds to input/output relationship among places linked by a transition in the Petri net of the construction. Therefore primitive points in the figure have no ancestors in the network. Objects have special attributes to hold reference to their ancestors/descendants, if any.

some points in the figure. We distinguish two cases: either the points are primitive, that is they are unconstrained and can satisfy the required transformations, or the points are derived, that is they are produced by an intersection method. This last case requires to solve an inverse problem, which is under-determined. Therefore, without an explicit procedure to deal with inverse constructions, the only reasonable solution would be to rigidly roto-translate the figure. A transformation can also be applied to some entities, different from points, such as a segment or a circle. In this case, the effect of a transformation is still a rigid roto-translation applied to the points that originate the entities.

Nevertheless this strategy is not fully satisfying: we often would like to modify a complex figure by moving some non primitive points as well, and to maintain the relationships and constraints among the objects. The extension to deal with this more complex situation is discussed after presenting the algorithm for transformation without inverse constrcutions.

3.1 The algorithm for transformations

The following algorithm deals with transformations on primitive and derived objects by propagating messages along the relationship network, when only direct constructions are

```
        if is_primitive
```

```
then


else
        transform_msg ( ancestors (anObject) )
```

The transformation of a construction is essentially the reconstruction of the figure, based on new values of the co-ordinates of some primitive point, as explained in/2.4. The propagation method broadcasts the transformation message to all descendants of an object, invoking the class or instance methods that have been activated during the construction of the figure. A non primitive object receiving the transformation message *transform_msg* broadcasts it to all its ancestors recursively. The algorithm is distributed among objects, because each object knows its ancestors and descendants, respecting the fundamental encapsulation principle of object-oriented programming.

3.2

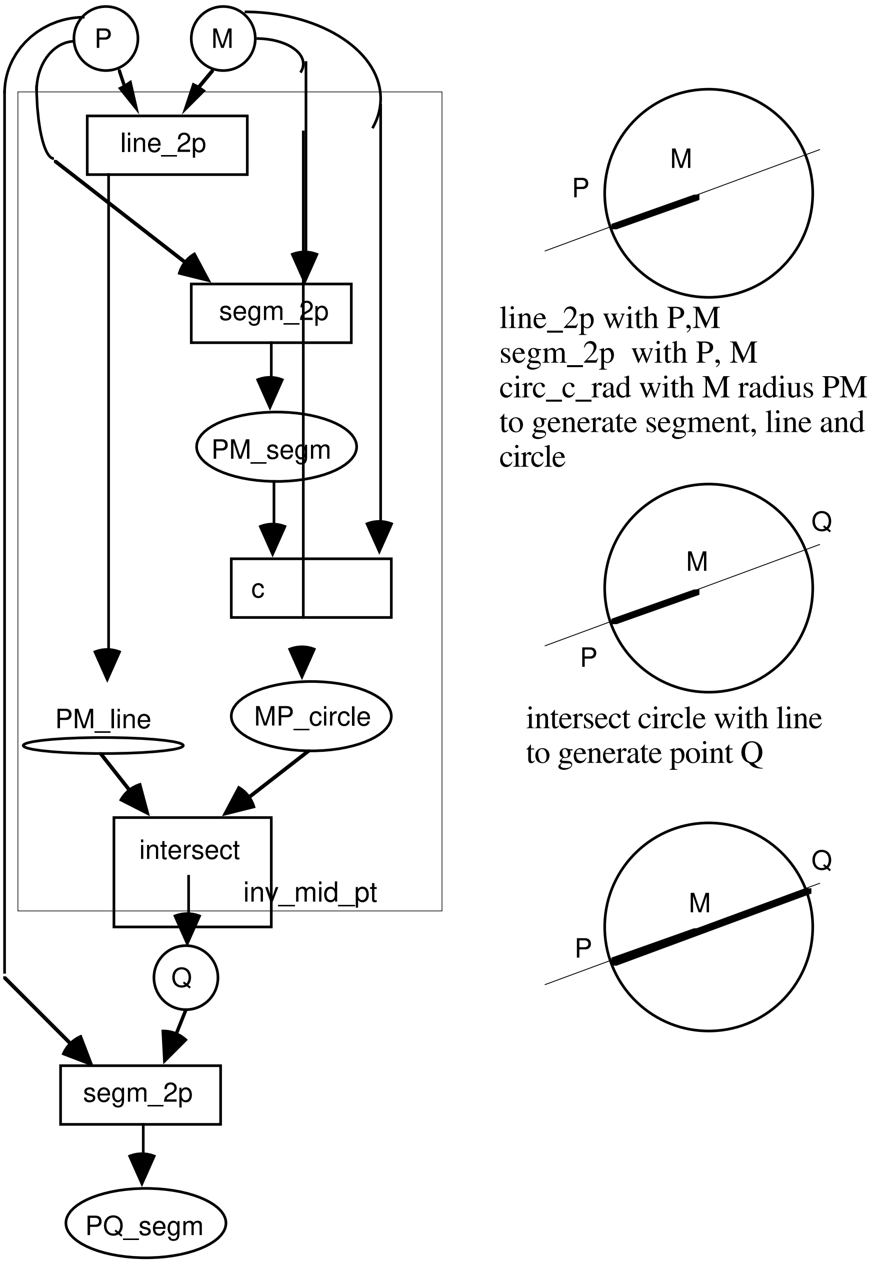
Let us now discuss a problem, which is critical and often encountered in geometric modelling environments: the problem of inverse constructions. An inverse problem arises when we would like to start from the solution of a previous problem (the direct problem) to find out the entities that have generated it. An inverse problem can be solvable or not, often it is under-determined.

In GEObject to solve inverse problems we adopt once more geometrical constructive methods, introducing in the system the concept of coincident points. If the inverse problem, besides the direct one, has a constructive solution, then applying both constructions we generate new multiple instances of points. For example, given two circles, with known radii and centres, a simple construction gives their intersection points. Its inverse problem, to find the centres given the intersection points and the radii, can easily be solved constructively as well. The centers, originated by the inverse construction, are the same geometrical points, that were primitive in the direct construction, even though they have been obtained in different ways.

The notion of coincident points is strictly a geometric concept and its implementation in GEObject is the key to allow uniform point manipulation during a figure transformation, without recurring to logical or numerical constraint solving methods.

3.2.1 An inverse construction example: the segment mid-point

Let us consider the problem of modifying the segment and its mid-point in any possible way: translating an end-point and re-computing the segment's mid-point or translating the mid-point, after fixing one end-point (anchor) and re-computing the other one. The Petri net in Fig. 4 gives information to solve only the direct problem, in order to solve the inverse one, we give the following construction:



line_2p with P,M
segm_2p with P, M
circ_c_rad with M radius PM
to generate segment, line and
circle

intersect circle with line
to generate point Q

Fig. 7. The construction of a segment given its mid-point and one end-point

We have now two methods to manage the segment and its mid-point: the direct one *mid_pt* and an inverse method *inv_mid_pt*. Applying method *mid_pt*, given PQ, we instantiate mid-point M and applying methods *inv_mid_pt* to PM and to QM we generate points Q’ and P’ respectively, as in Fig. 8. Apparently we have now three different segments and mid-points: PQ, PQ’ and P’Q and their mid-points M, M’ and M". Therefore in order to transform the segment and its points, including the mid-point, we need to impose a coincidence relationship to the segments and hence to their end-points.

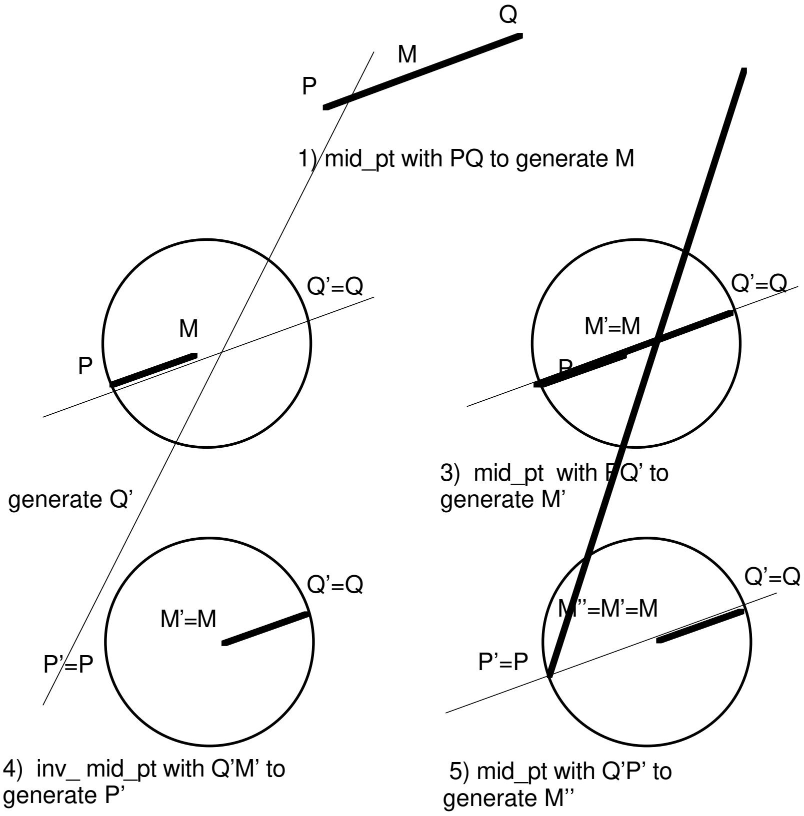


Fig. 8. Applying direct and inverse methods to construct segment mid-point, producing multiple coincident points

These constructions allow now to transform any point. Above constructions are described by the following Petri net, where we represent multiple occurrences of coincident points with places linked together by a special transition, which guaranties that their tokens are always of the same colour. We call this transition *coincide*. The object-oriented implementation of coincident points is realised by means of a class variable, *has_alias*, that establishes the relationship with all coincident points.

In the network of Fig. 9 we introduced an 'anchor' to fix point P, to allow one *inv_mid_pt* method only to fire, when we transform the mid-point M. Without anchoring one end-point a random transformation *inv_mid_pt* would fire, and the segment would either stretches on P or Q randomly. In the Petri net 'anchoring' a place means to modify a colour attribute of its token.

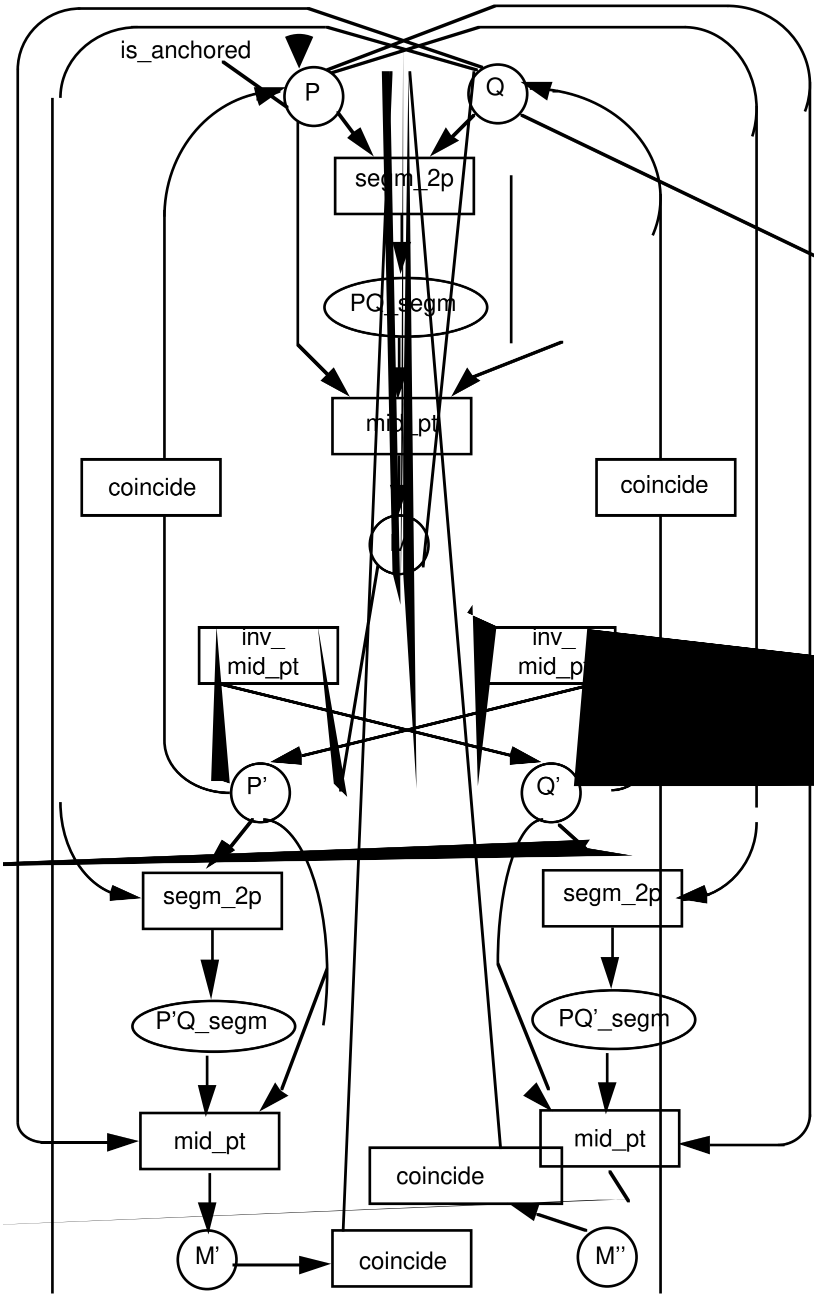


Fig. 9. The complete graph for a segment mid-point

We can now generalise the message propagation algorithm on the relationship network to implement transformations when some inverse constructions are involved, by refining the method invoked by *transform_msg*:

```
transform_msg (anObject):
        if  not  is_anchored
                then
                        if is_primitive or has_aliases
                                then
```

```
            transform_msg (descendent (anObject) )
        else
```

The generalised algorithm allows to transform a figure in a more general way, increasing the flexibility of geometric transformations preserving constraints. In a figure coincident points are the result of some inverse construction, that requires to impose the coincidence relationship on some derived points. It is important to underline that it is the user’s responsibility to specify an inverse construction for a given figure, if it exists.

In our system the coincidence relationship is the only constraint that is imposed a posteriori. Any other constraint must be satisfied by finding a suitable geometric construction.

3.3 A virtual geometric laboratory

To experiment our approach and algorithms we implemented a virtual geometric laboratory. The user interface is based on object-oriented paradigm, as well. Users interact with geometric objects, primitive or structured, sending messages by means of a basic action: the picking event sends back a message to the interface which pops up a menu, to select a method among the available ones for the selected object.

The system allows users to perform interactively different activities:

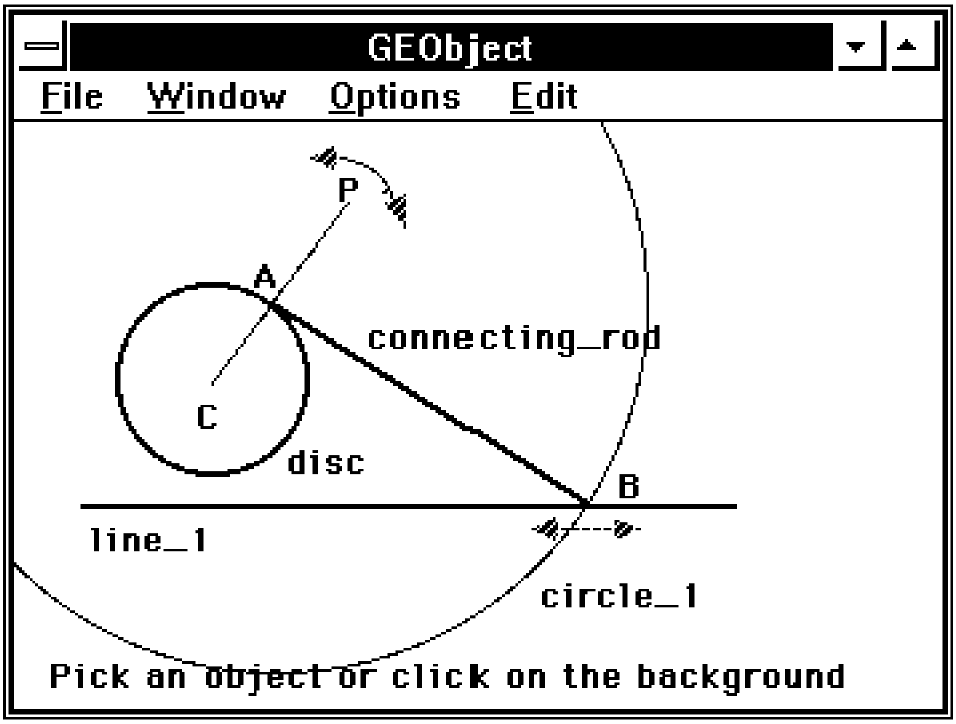
¥
point, length of a segment;
¥
¥ defining a new method via a geometric construction;
¥ defining a new class via a geometric construction.

In order to define a new method or a new class an editing mode is activated, during which the system records the sequence of objects selected and the messages sent by users, generating the relationship network. It is the user s responsibility to define parameters and returned objects of a new method or class. A new method or a new class is included in the list of available instance or class methods. This procedure to define new

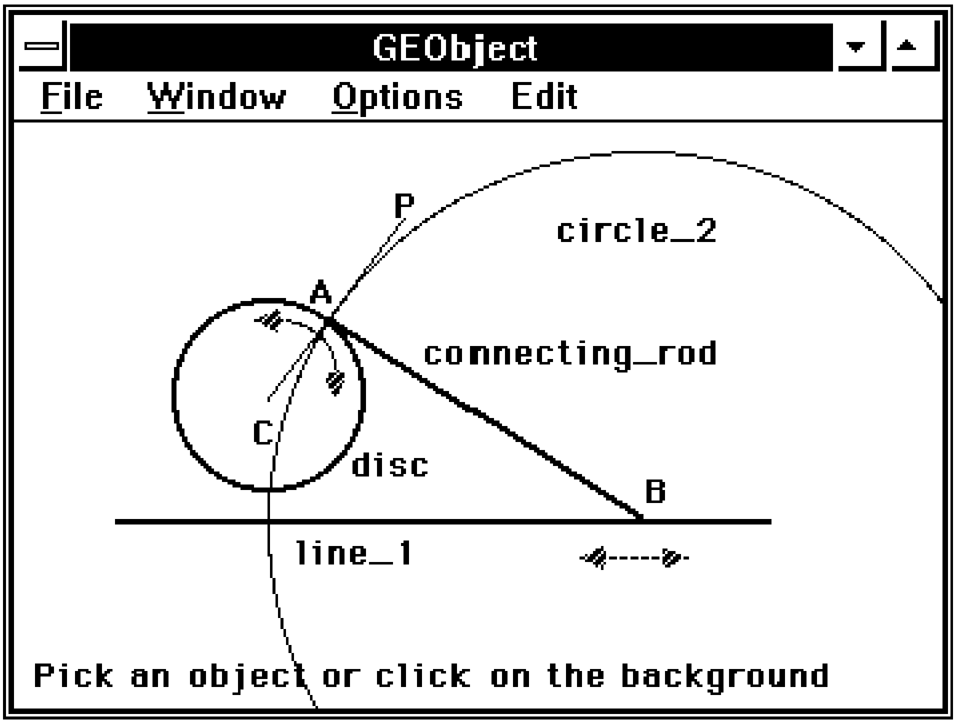
programming.

3.3.1

An important application of constraint maintenance is found in computer animation: to simulate the cinematic of an articulated structure requires to maintain its internal links. Let us show an example that can easily be solved by our approach: the transformation of a circular motion into a rectilinear one, implemented by the disc-connecting rod-piston system. To fully characterise this structure we need to describe a direct and an inverse construction, so that the propagation algorithm, when activated by a transformation message, is able to transform the relevant points of the construction and to simulate the motion.



Pick an object or click on the background



Pick an object or click on the background

Fig. 10 Cinematic system disc-connecting rod-piston.
Top: the direct construction to drive the piston by rotating the disc.
Bottom: the inverse construction to drive the disc by sliding the piston.

The disc, the rod length and the line_1 are given. In the left figure the direct construction allows to find the point B given a point A on the disc and finding the intersection of line_1 with a circle (circle_1) whose radius is equal to the rod length: AB is the resulting connecting rod. In the right figure the inverse construction allows to find point A on the disc given point B, by intersecting line_1 with a circle (circle_2) centered in B whose radius is equal to the given rod length. Coincidence relationship must be imposed on the two occurrences of point A: A is a point constrained on the disc and the intersection of the disc and circle_2. The direct construction makes possible to rotate point A on the disc and causing point B to slide on the line; the inverse construction makes possible the opposite movement: to translate point B along the line and causing point A to rotate on the disc. It is important to note that we solve this cinematic system in a straightforward manner without solving any inverse equation system.

The point constrained to a given entity is a sub-problem that involves a direct and an inverse problem, as well. A point on a circle can be seen as the result of an intersection between the circle and a generic segment. To be able to move the point on the circle it is necessary to solve the inverse problem of intersection between the circle and the segment, as well. For instance in Fig. 10, the point A on the disc is obtained by intersecting the disc with segment CP; rotating A requires to re-build segment CP.

Many similar cinematic system can be constructed and their motion simulated, e.g. an articulated chain of rods with rotating links, like a robot arm, or a quadrilateral, typically used in mechanical systems.

4 Conclusions and related works

GEObject is an object-oriented system, that allows to model geometric figures and that incrementally extend the class of geometric entities by constructions, using compass and ruler, with no explicit programming involved. The relationships established during a construction express the constraints among entities; the system maintains an internal representation of such relationship net which is used by an algorithm that propagates geometric constraints, without recurring to numerical constraint solving methods. The system allows to manipulate complex figures, like a cinematism, dealing also with direct and inverse constructions. Object orientation in GEObject is characterised by the representation of geometric entities on the basis of fundamental properties of euclidean geometry.

Many sophisticated interactive graphics systems Sutherland [1963], Borning [1981], Nelson [1985] are based on numerical constraint satisfaction methods, which are computationally complex. Moreover their accuracy is a critical problem and it is difficult to specify exactly or modify the constraints, avoiding incompleteness, redundancy or unsatisfiability.

The importance of constructive approach to picture building has been pointed out also by Noma et al. [1989]. Their approach is mainly a library based system that requires explicit programming and does not support construction hierarchies, while our object oriented approach allows us to build more complex constructions from simpler ones, and extend easily the set of available construction methods, without recurring to explicit programming.

L.E.G.O. by Fuller et al. [1989], is a LISP based system which adopts the constructive approach, in contrast to constraint solving techniques. It is a highly interactive environment for teaching and exploring constructive geometry. L.E.G.O. also adopts a visual programming paradigm to extend the set of geometric constructions, so that they become available for future use.

Gleicher [1992] identifies the limits of constraint based approaches separating the problem into two main steps: establishing relationships, adopting 'augmented snap dragging' technique, and maintaining them during figure manipulation. The relationship maintenance stage is implemented by solving systems of differential equations that express modification of the original figure.

Object oriented paradigm with constraint based system has been proposed by various authors as Teixeira [1993], Hill [1991], Rankin & Burns [1991]. In particular Texeira adopts a network propagation method, similar to Borning approach, based on a declarative representation of geometry.

Maulsby et al. [1989] also suggest a solution to constraints specification, which is based on the recording of the history of drawing operations and on creating a state graph, that a geometric knowledge base uses to recover the figure when a transformation is applied.

Pineda [1992] suggests an approach based on a symbolic characterisation of constraints rather then a numerical one; so constraint solution is reduced to a logical problem of satisfying predicates that express relationships. We consider a limit of such approach the necessity of explicitly declaring predicates among objects to establish constraints of a figure.

The value of object-oriented approach to graphics programming has been pointed out by Wisskirchen [1990] who has implemented GEO++, a graphics kernel based on an object-oriented approach, and developed using a full object-oriented environment. This system is more an object version of traditional standard graphics packages, like GKS, rather then a new design of geometric knowledge representation.

Our system shares similarities also with two commercial products: Cabri, by Baulac et alt. [1991] and Geometer's Sketchpad, Jackview [1991]. Both systems support compass and ruler geometric constructions. Their approach to geometric transformation is rather