Interactive 2D Constraint-Based Geometric Construction System

Benachir Medjdoub
The Martin Centre, University of Cambridge, UK

Key words: Sketching, Geometric Constraints, Interactivity, Geometric construction, Dynamic modifications.

Abstract: This paper presents a 2D Constraint-Based Geometric Construction System where positioning and manipulating geometry is very precise. An unusually simple interface makes this system particularly interactive and easy to use.
In our approach, the geometry types supported are: points, lines, circles, ellipses, circular arcs and b-spline curves. All the fundamental topologic constraints, i.e. tangent, parallel, perpendicular, coincident and concentric, are provided. Metric constraints, i.e. dividing the shapes into equal parts or fixing the geometric parameters, are also provided. These constraints are automatically applied by the application in response to the implied intentions of the end-user. Dynamic modifications of partially dimensioned models are supported, whereby the design is modified while enforcing the constraints. A graph-constructive approach is used to solve the model. As we are dealing with partial modifications, this resolution technique is quite sufficient, and makes our system stable and flexible. Our approach focuses highly on interactivity. Positioning a shape constrained to another is made directly through the graphic interface. Constraint relaxation is also done by direct manipulations. Modifications are made by dragging the geometry, or by typing into a numerical panel displaying the free shape parameters. Again, existing constraints are maintained as those numbers are applied. Well-constrained and under-constrained problems are discussed. This approach was developed in Java, JDK 3.0.1 of SGI's Java software.

1. INTRODUCTION

Geometry is a fundamental tool in the hand of any architect. Associative geometry encourages the plastic interaction with design form, provides geometric accuracy and allows for the exploration of variation in architectural design [Ais92]. Present day architectural CAD systems are not good at handling geometric constraints. How many would allow you to construct the ellipses tangent to two circles and one line, interactively choose one, then divide the elliptical perimeter into equal parts? Having done this, how many allow you to drag one of the circles, and see the ellipse and its subdivisions change, while maintaining all the constraints?

This problem has been considered by several communities using different approaches often in mechanical engineering and manufacturing [Ald88] [Kon90][Kra92] than architecture. Two major issues have arisen from previous research: the constraint structure and the method of resolution.

From consideration of architectural applications, our purposes and goals are as follows:

– We enlarge the geometric object structure to ellipses and splines with constraint model enrichment.

– We treat both well and under-constrained problems. These two classes of problem are driven by the user interface.

– We present a new user interface where constraining, modifying and relaxing constraints is done interactively and precisely.

We present related work in section 2. The representation of geometric constraints and the user interface are presented in sections 3 and 4. The constraint solver is described in section 5. In section 6 some examples are presented. Before concluding in section 8, the implementation is presented.

2. RELATED WORK

In previous work the geometric elements are generally restricted to four types, namely point, line circle and circular arc [Bou95][FH93][Fud93][GC98][LG82] [Ala93]. Systems which handle ellipses and splines are rare.

The repertoire of constraints differs from system to system. Although there is a common kernel of constraints that consists of distances and angles, the parallel, perpendicular and *on* constraints are usually treated separately. We could have a hierarchy of constraints (like the one presented in [BM89]), where the explicit constraints override the implicit constraints. Usually the geometric constraints are classified in two categories, the topologic or structural constraints (tangency, perpendicular, parallel, on and concentric) and the metric constraints [Ald88] (which fix coordinates, distances, lengths and angles).

Four main approaches have been developed to solve geometric constraints: the numerical approach, the symbolic approach, the constructive approach and the propagation approach.

In the numerical approach the constraints are translated into a system of equations and solved using iterative methods such as Newton iteration [SB93][LG82]. The exponential number of solutions and the large number of parameters make this method inappropriate when the initial sketch is just topologically defined. Sketchpad, described in [Sut63] was the first system to use the method of relaxation as an alternative. Others system such as ThingLab [Bor81] kept the relaxation as an alternative to other methods. The Newton-Raphson method has proved to be faster than relaxation [HN94].

The symbolic approach is quite similar to the numerical approach. The geometric constraints are also transformed into algebraic equations. However instead of using numeric methods to solve the algebraic equation, the system is solved with symbolic algebraic methods, such as Grobner's bases [Buc88], or the Wu-Ritt method [GC98]. Both methods can solve general non-linear systems of algebraic equations, but may require exponential running times.

The constructive approach is based on the fact that most configurations in an engineering drawing can be solved by ruler, compass and protractor. In these methods, the constraints are satisfied constructively. The first constructive approach was based on rules. It used rewrite rules to discover and execute the construction steps [Ald88][Bru86][Sun88]. It was shown that this method is correct and solves all problems that can be constructed using ruler and compass. The other constructive approach is based on graphs, and has two phases. First, a graph representing the constraints is analysed and a sequence of construction steps is derived. Second, the construction steps are carried out to derive the solution. This approach is fast and more methodical than the rule-constructive approach. However, as the repertoire of possible constraints increases so the graph-analysis algorithm has to be modified. Requicha [Rec77] uses dimensioned trees that allow only horizontal and vertical distance. Todd [Tod89] generalises the dimension trees and gives a characterisation of the expressive power of the solver. Kramer [Kra92] describes a 3D constraint solver that deals with constraints from kinematics. Owen [Owe91] presents an extension of this principle to include circularly dimensioned sketches, and DCM is a commercial constraint solver [DCM93] using this method.

The propagation approach based on the constraint programming technique does not guarantee to derive a solution or to have a reasonable worst case running time [Doh95].

3. REPRESENTATION OF GEOMETRIC CONSTRAINTS

We have defined our constraint model and the 2D geometric elements according to the most common constraints used in architectural drawing. In [Bou72] a parallel was made between the architectural design process and the geometry classification proposed by the mathematician Felix Klein. Five levels of geometry are defined, among them topologic and Euclidian geometry. Topologic geometry defines continuity and neighbourhood in a building; it corresponds to the sketch step. Euclidian geometry defines distances and angles, and corresponds to the last step of design. From these two levels we can deduce the two categories of constraints which are known as the topologic and metric constraints.

3.1 Geometric entities

The description of 2D geometry in this paper is based on six types of geometric entities, namely point, line, circle, circular arc, ellipse and b-spline curve. Each geometric entity is characterised by a set of attributes which can be described in three categories:

– Geometric attributes: a circle defined by its radius and its centre. An ellipse is defined by its two radii, its centre and its orientation which generate the coefficients of the corresponding conic equation. A spline is defined by its knots.

– Topologic attributes: list of points which define the type of the constraints and the other geometric entities with which the current geometry is constrained.

– Degrees of freedom: boolean variables which correspond to the geometric parameters. In the case of an under-constrained problem these variables allow us to fix some geometric parameter. A priori some parameters are fixed (i.e. degrees of freedom corresponding to line's slope are fixed).

Points can be generated in different ways; explicitly by the user from the menu bar, by intersection points between different geometry, or implicitly by applying a constraint between two elements. In all the cases the point has a label referring to the kind of the constraint, i.e. if the point is a tangent point the label *tangent* is activated. This label is represented by a boolean variable. The points and their label play a central role in the constraint solving method.

3.2 Geometric constraints

We have considered the two constraint categories: topologic constraints and metric constraints (some examples are shown in Figure 1).

Topologic constraints comprise:

– *Tangent* constraint between lines, circles, arcs and ellipses, in all combinations.

– *Perpendicular* constraint between line and line, line and circle, line and arc and line and ellipse.

– *On* constraint: we can constrain a point to be on any other shape, by the way the shape can be constrained to pass over a set of points (i.e. circle by one, two or three points…). These points can be intersection points.

– *Parallel* constraint between lines.

– *Concentric* constraint between circles and arcs.

Metric constraints: are needed more in case of the under-constrained problem. The user can fix according to his need some parameter such as the radius of a circle, or the slope of a line. An example is shown in section 5.2. Among the metric constraints are dividing shapes into equal parts, which is very useful in architectural drawing.

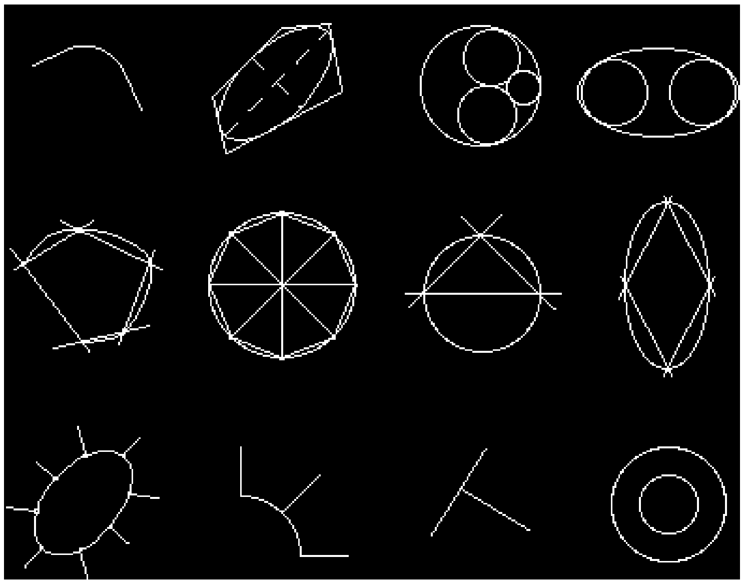


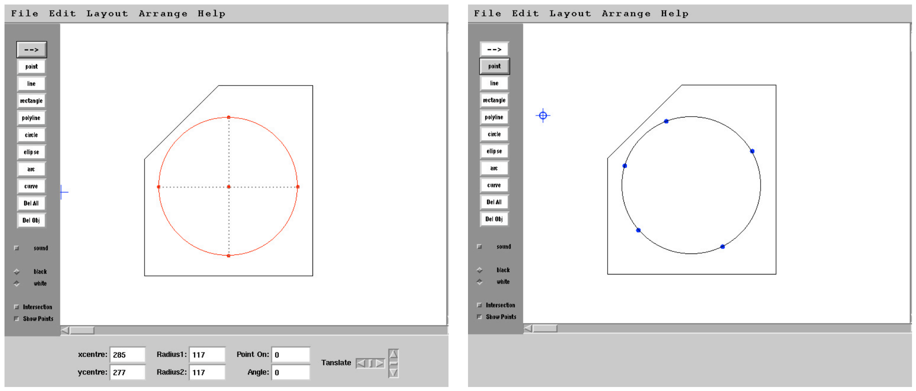
Figure 1. Some geometric constraints supported by our system

The mathematics of a constraint depends on its type. Some constraints are reduced to a quadratic equation. Other constraints such as tangents between ellipses are more complex and need iterative methods to be solved.

4. USER INTERFACE AND INTERACTIVITY

Architectural drawing is a process of adding and modifying. Drawings are built up gradually by adding to what has already been drawn [And92]. Van Sommers refers to this as *accretion* [VSo84]. This build-up tends to take place by locating new elements in some relation to what is already there – *anchoring*. Modifying is done in relation to what is already drawn. Two kinds of modifications can be made: topological and geometrical. The user interface of our approach is based on these two principles of adding and modifying. As an example we are going to construct and modify a geometric model, and demonstrate how easy it is to use this system.

All the constraints defined are done so explicitly by the user. Constraints are located using the right mouse button. Drawing without constraint is done using the left mouse button. In the first step of our example (Figure 2a), we draw an irregular polygon and an ellipse. After that we position five points constrained to be on the ellipse (Figure 2b). In order to do this we simply have to select the button *"point"* on the left menu bar and, with the right mouse button, click five times on the ellipse. The system recognises the approximate position on the ellipse. The *on* constraint is then applied and solved exactly. These actions release a flow of information. The points take the label *on.* The topologic parameter of the ellipse includes the fives points, and the fives points include the ellipse in their topologic parameters. The points may not be on more than two shapes, which avoids conflicts in solving the constraint.



xcentre: 285 Radius1: 117 Point On: 0
ycentre: 277 Radius2: 117 Angle: 0

Figure 2(a,b). Drawing an irregular polygon and five points constrained to be on an ellipse

As it is shown in Figure 3, positioning the ellipse tangent to the line is done directly by dragging the point on the line. When an approximate tangency is recognised by the dragging routine, a tangent constraint is applied, and solved exactly. Dragging a second point to touch a second line, a new tangent constraint is added, while maintaining the first. As soon as the tangent constraint is applied the point becomes a tangent point, in this case its label which was *on* becomes *tangent.* The point is added to the line's topologic parameter, and a line into the point's.

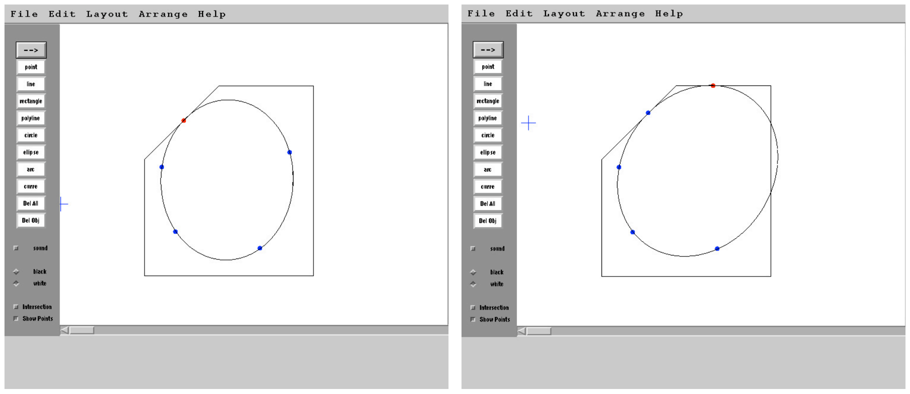
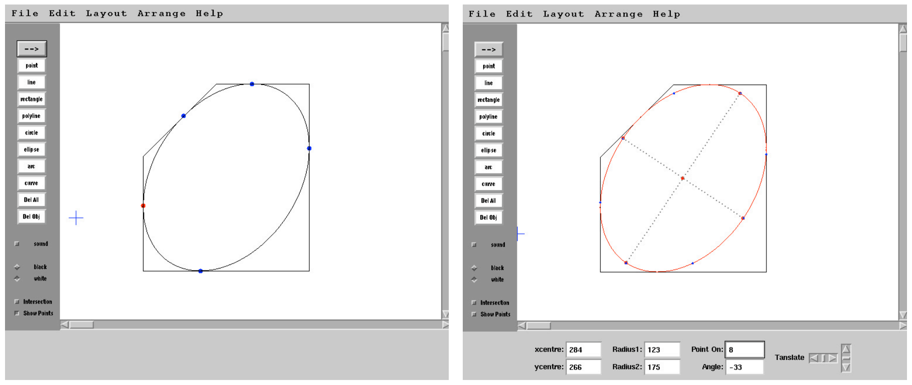


Figure 3(a,b). Drag a point on the ellipse to the line; a tangent constraint is applied

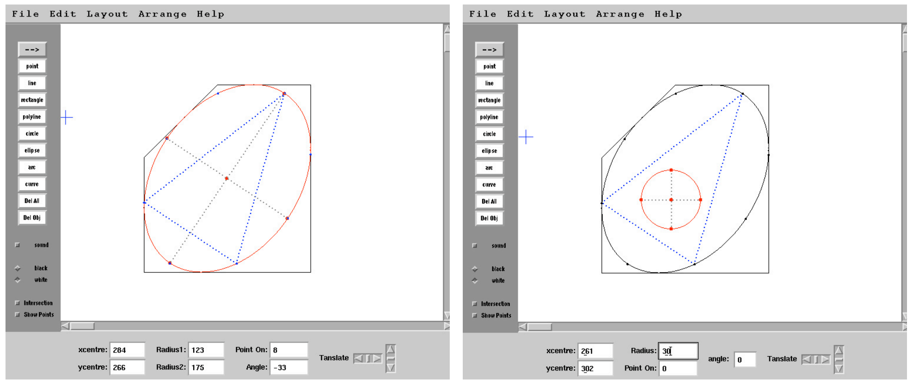
We continue until the ellipse becomes tangent to the fifth line (Figure 4a). In Figure 4b we show eight points constrained *on* the ellipse and dividing the elliptical perimeter into equal parts. This constraint is applied after selecting the ellipse and validating the number of the points on the appropriate numerical panel.



xcentre: 284 Radius1: 123 Point On: 8
ycentre: 266 Radius2: 175 Angle: -33

Figure 4(a,b). Continue until ellipse is tangent to fives lines (On the left), constraint 8 points to be on the ellipse dividing the elliptical perimeter into equal parts

For the next step we select three construction lines defined by three points on the ellipse and draw a circle (Figure 5).

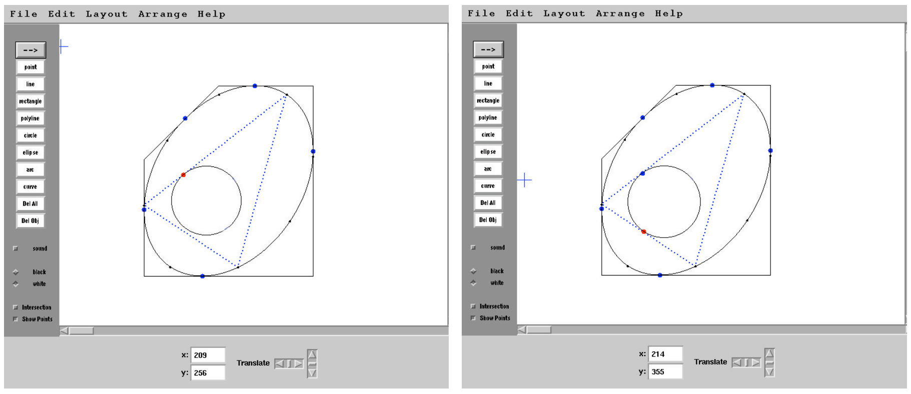


xcentre: 284 Radius1: 123 Point On: 8
ycentre: 266 Radius2: 175 Angle: -33

xcentre: 261 Radius: 30 angle: 0
ycentre: 302 Point On: 0

Figure 5(a,b). Three construction lines defining a triangle between three points on the ellipse are selected

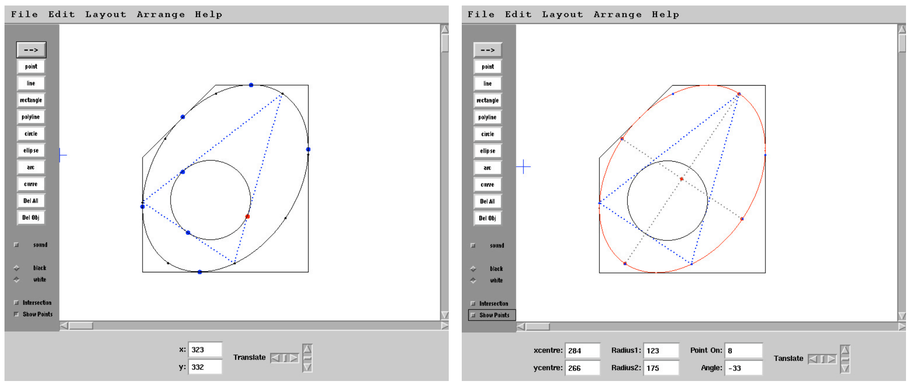
We drag the circle from any point on the circle to the construction lines. When the approximate tangent is recognised by the dragging routine, the tangent constraint is solved exactly. We drag a second point to the second construction line. A new tangent constraint is added, while maintaining the first. We continue until the circle becomes tangent to the three construction lines (Figure 6 and Figure 7). The three tangent points take the label *tangent* and their topologic parameters are updated. In the case of the circle, the points are generated implicitly by the system as soon as the tangent constraint is recognised.



x: 209
y: 256

x: 214
y: 355

Figure 6(a,b). Drag a point on the circle to the construction line; a tangent constraint is applied



x: 323
y: 332

xcentre: 284
ycentre: 266
Radius1: 123
Radius2: 175
Point On: 8
Angle: -33

Figure 7(a,b). Continue until the circle becomes tangent to the three construction lines

Modifying the geometric model can be done in two ways; by the numerical panel or by direct manipulation. In the Figure 8 we move the line by fixing its slope parameter (angles between the lines). The geometric model change as the constraints are maintained.

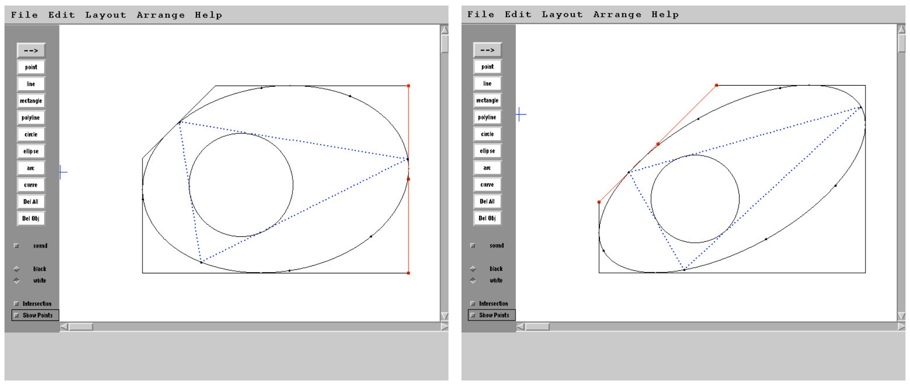


Figure 8(a,b). Move the lines without modifying the slopes, the geometric model changes as all the constraints are maintained

Constraint relaxation is also done by direct manipulation. You only have to drag a tangent point outside for the constraint to be relaxed. If the point is dragged with the left button of the mouse, the constraint is completely eliminated. When the geometric model is over-constrained during modification, the system gives you a message asking you to eliminate some constraints. The system allows you a free choice of the type of constraint to eliminate.

5. CONSTRAINT SOLVER

We have chosen in our approach the graph-constructive solving method. This method presents many advantages: in particular it is fast, more methodical and proven to be sound [Fud93]. Two steps characterise this method: *graph construction-analysis* and *constraint solving*.

– *Graph construction-analysis*: the user sketch, annotated with constraints, is translated into a graph whose vertices correspond to geometric elements: points, lines, circles, ellipses, splines, arcs, and whose edges represent the constraints between them. After the construction step, the graph is analysed and a constraint solving strategy is defined. The graph construction is performed only in case of drawing or modifying with constraint. In the example shown in Figure 3a, when we drag a point to the first line, the constraint *ellipse through five points* is maintained. In this case as soon as we select the point on the ellipse with the right mouse button, the graph as indicated in Figure 9a is generated. Graph analysis deduces the type of constraint to solve. In this case the constraint "*ellipse through five points"* is detected and solved at each modification of the first point. But when the first point is almost on the line, it is recognised, and the graph is reconstructed. A node representing the line is added (as indicated in Figure 9b). When this point is near the tangent point between the ellipse and the line, the point acquires the label *tangent,* which again modifies the graph, and a new constraint is detected corresponding to the new situation. This constraint becomes *"ellipse through four points and tangent to one line"*. The fifth point became the tangent point with the first line.

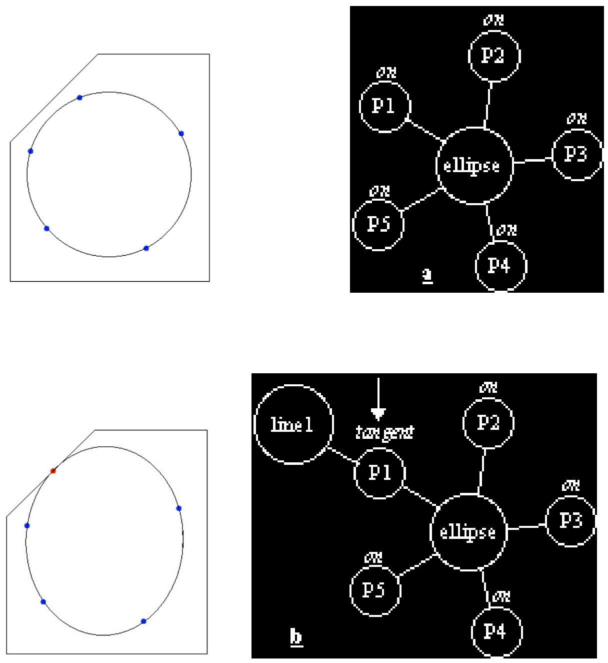


Figure 9(a,b). Geometric model and corresponding constraint graph.

– *Constraint solving*: when the construction steps involve ruler-and-compass constructions, only quadratic equations need to be solved. However we are more ambitious and some construction steps are permitted that are not ruler-and-compass; and in those situations the solutions are found numerically. In the example shown in Figure 8, the constraint which maintains the eight points dividing the elliptic perimeter into equal parts is solved using the elliptic integral of the second kind. Its resolution is done using Simpson's rule, with an error of less than 1/10000. The resolution of the tangent constraints between ellipse and other geometric entities use iterative methods, with the same error margin.

5.1 Well-constrained problems

A well-constrained problem can have many solutions. Thanks to certain heuristics we conduct the system to find a particular solution. The example shown in Figure 10 is well constrained (as the circle is defined by three points), yet can have up to 8 solutions. The solution which best matches the mouse position is used.

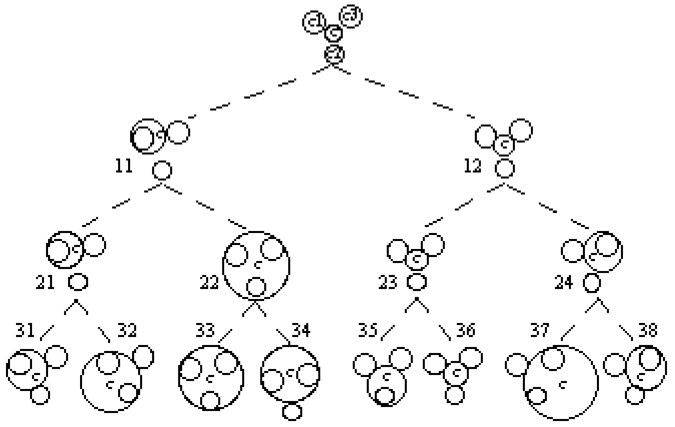


Figure 10. The eight possible solutions driven by the user decision thank to the mouse position

5.2 Under-constrained problems

Our approach supports the under-constrained problems. In this case the system is driven by the degree of freedom of the geometric elements. For example, by default, the circle has both of its parameters, the radius and the centre, free. Here, the user can display the degrees of freedom of the circle and can fix or release them. The degrees of freedom don't intervene in the well-constrained problems. In the example shown in Figure 11 we have a tangent between two circles *c1* and *c2*. In case of any modification we have several possibilities. If we would like to modify the radius of *c2* without moving the tangent point (case 1, Figure 11) we leave free both parameters of *c2*. If we would like to modify the centre of *c2* without modifying its radius, we fix its radius parameter (case 2, Figure 11). This modification can be done by dragging the circle *c2*, or numerically by using the numerical panel. In the first case we could put the exact radius wanted for *c2*. In the second case, we could put the exact angle between *c1* and *c2*.

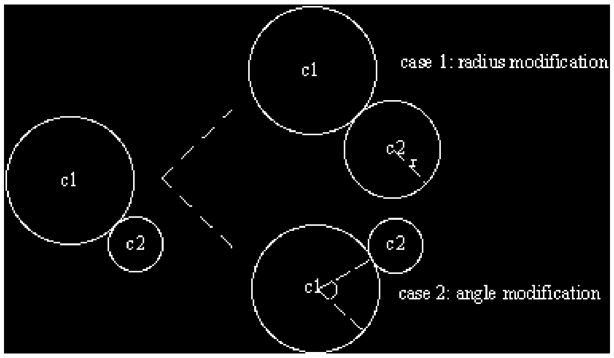
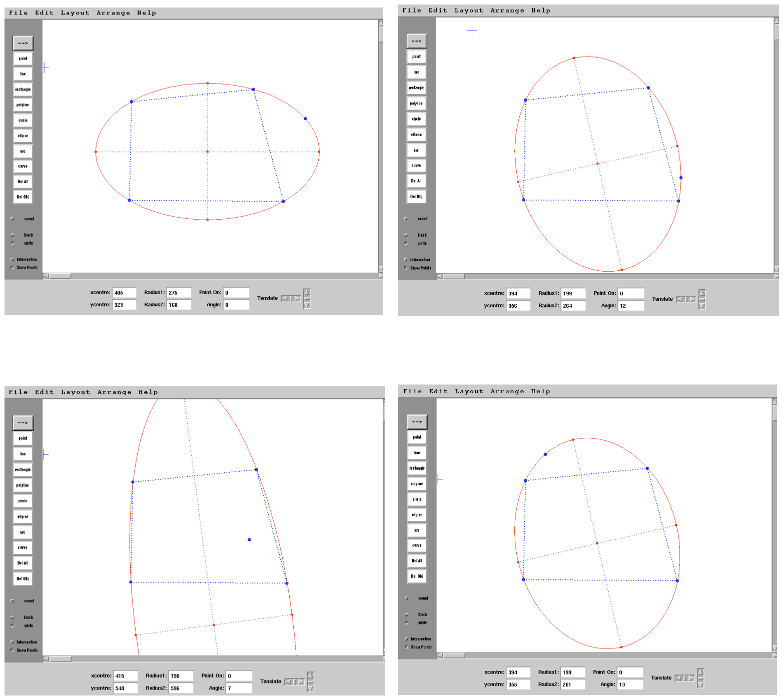


Figure 11. Different solutions driven by degrees of freedom

5.3 Conflict Management

Some conflicts can appear in the case of modifications. In Figure 12 we show an ellipse constrained by five points. If we drag one point, the new ellipse parameters are computed (Figure 12b). If the conditions $A\cdot C-B\cdot B>0$ and the eccentricity $e<1$ are not satisfied (A, B, C and e correspond to the conic parameters), there is no solution. This situation corresponds geometrically to one point being inscribed in the polygon formed by the other four. In this case the system keeps displayed the last solution found (Figure 12c) until the conditions are satisfied again (Figure 12d).



xcentre: 405 Radius1: 275 Point On: 0
ycentre: 323 Radius2: 168 Angle: 0

xcentre: 394 Radius1: 199 Point On: 0
ycentre: 356 Radius2: 264 Angle: 12

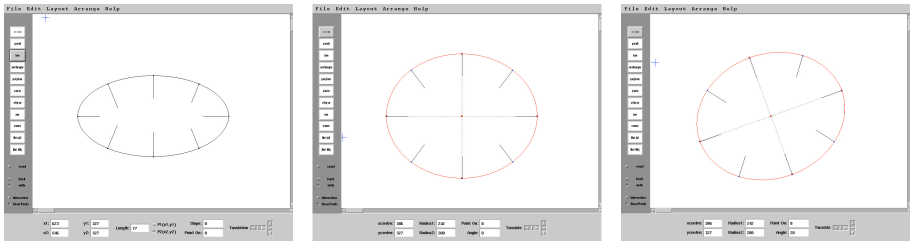
xcentre: 415 Radius1: 190 Point On: 0
ycentre: 548 Radius2: 596 Angle: 7

xcentre: 394 Radius1: 199 Point On: 0
ycentre: 355 Radius2: 261 Angle: 13

Figure 12(a,b,c,d). Conflict management

6. EXAMPLES

We have tested several examples which allow us to evaluate the performance of this approach. The example of Figure 13 shows how it is easy to work with elliptic forms. This geometric model is defined by an elliptic perimeter divided into eight equal parts. From each point a line perpendicular to the ellipse is drawn, these lines have the same length. It is possible to make a lot of variations of this model maintaining all the constraint: modify the ellipse orientation, its radii. Examples of buildings that have an elliptic perimeter include the *Lipstich Building* in New York designed by Philip Johnson and the *conference room of the Chamber of commerce* in Berlin designed by Nicholas Grimshaw & Partners.



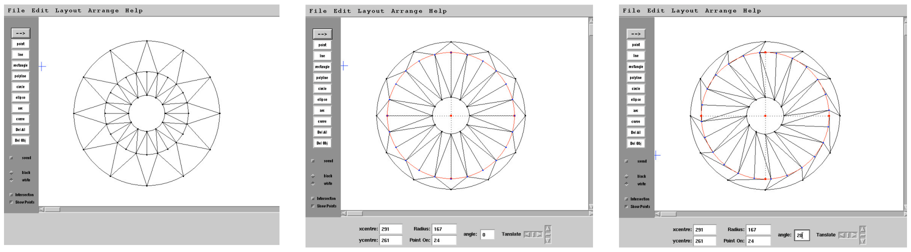
x1: 623 y1: 327 Length: 77 P1(x1,y1) Slope: 0
x2: 546 y2: 327 P2(x2,y2) Point On: 0

xcentre: 386 Radius1: 242 Point On: 8
ycentre: 327 Radius2: 200 Angle: 0

xcentre: 386 Radius1: 242 Point On: 8
ycentre: 327 Radius2: 200 Angle: 20

Figure 13. Example with an elliptic form

In Figure 14, the geometric model is defined by three circles and lines. Many variations of this model are presented: radius (Figure 14b) and circle orientation (Figure 14c).

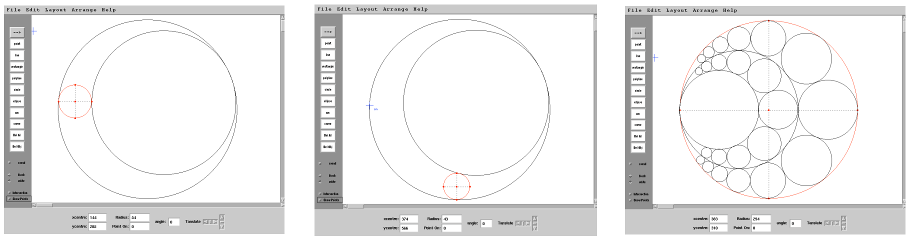


xcentre: 291 Radius: 167 angle: 0
ycentre: 261 Point On: 24

xcentre: 291 Radius: 167 angle: 20
ycentre: 261 Point On: 24

Figure 14(a,b,c). Example with circles and lines

In Figure 15, playing with circles is shown. Figure 15b shows the modification of the small circle just by a simple dragging. The Figure 15c shows an arbelos.



xcentre: 144 Radius: 54 angle: 0
ycentre: 285 Point On: 0

xcentre: 374 Radius: 43 angle: 0
ycentre: 566 Point On: 0

xcentre: 383 Radius: 294 angle: 0
ycentre: 310 Point On: 0

Figure 15(a,b,c). Example with circles

The last example (Figure 16) is a model with circles and an ellipse. Some variations are also shown.

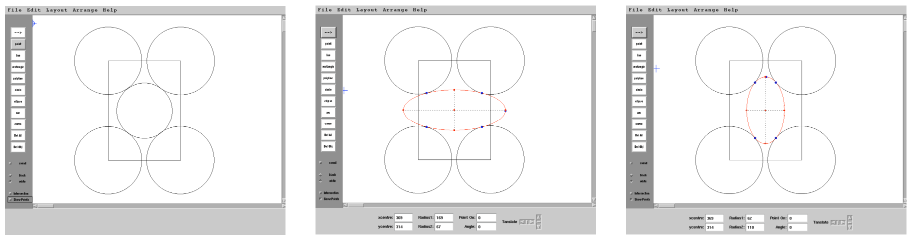


Figure 16. Example with circles and ellipses

7. IMPLEMENTATION

This application is developed in Java, JDK 3.0.1 of SGI's Java software (IRIX 5.3). This release is conformant with the behaviour of Sun's 1.1.3 JDK. Java is an Oriented Object Language that is based loosely upon C++. Java Applets can be transmitted over the Internet and executed, in theory by any Java-compatible Web Browser. We find in practice that this applet executes well under Linux and Windows/NT, but poorly on the Macintosh.

8. CONCLUSION

The approach presented in this paper show an interactive, simple and high-level modification of part geometry via numerical panel or by simple geometry dragging. Several aspects were discussed: the constraint and geometric object structure, the method of constraint solving, and interaction via the user interface.

Compared to previous work, we have enlarged the geometric object structure to ellipses and splines with constraint model enrichment. We have discussed well and under-constrained problems. The system manages constraint conflicts, without any instability. Another advantage of this approach is the particular user interface which makes this system interactive, simple and useful. This interface supports the resolution of both well and under-constrained problems.

Many extensions are presently under study:

– Constraint enrichment, particularly with respect to splines.

– A diagnostic tool for the over-constrained problem in order to guide the user to relax a specific constraint.

– Incorporating this system as a module in a CAAD package.

ACKNOWLEDGEMENTS

This work is sponsored by Informatix Inc. The author would like to thank Paul Richens, the Martin Centre director, for valuable help and comments about various aspects of this research project.

REFERENCES

Aish R, 1992, "Computer-Aided Design Software to Augment the Creation of Form", in *Computer in Architecture*, 97-104, Longman, London.

Alasdair S, et al., 1993, "Constraint Definition System: a Computer-Algebra Based Approach to Solving Geometric-Constraint Problem", *Computer Aided Design,* 25 (12) 741-750.

Aldefeld B, 1988, "Variation of Geometries Based on a Geometric-Reasoning Method", *Computer Aided Design*, 20 (3) 117-126.

Andre G P, et al., 1992, "Computer Aids For Design Development", in *Computer in Architecture*, 15-24, Longman, London.

Bruderlin B, 1986, "Constructing Three Dimensional Geometric Objects Defined by Constraints", *In workshop on Interactive 3D Graphics ACM*, 111-129.

Borning A H, 1981, "The Programming Language Aspects of ThingLab, a Constraint Oriented Simulation Laboratory", *ACM TOPLAS*, 3 (4) 353-387.

Borning A, Maher M, et al., 1989, "Constraints Hierarchies and Logic Programming", *In Proc. Of the 6th International Logic Programming Conference*, 149-164.

Boudon Ph, et al., 1972, *La geometrie chez l'architecte : Etude exploratoire* (UP-architecture-Nancy).

Bouma W, et al., 1995, "A Geometric Constraint Solver", *Computer Aided Design*, 27 487-501.

Buchberger B, 1988, "Algebraic Methods for Geometric Reasoning*", Annual Reviews in Computer Science*, 3 85-120.

Dimensional Constraint Manager, 1993, D-Cubed Ltd, 68 Castel Street, Cambridge, CB3 0AJ, UK.

Dohmen M, 1995, "A Survey of Constraint Satisfaction Techniques for Geometric Modeling", *Computer & Graphics,* 19 (6) 831-845.

Fudos I, 1993, "Editable Representations for 2D Geometric Design", *Master of Science Thesis*, Perdue University.

Fudos I, Hoffmann C M, 1993, "Correctness Proof of a Geometric Constraint Solver", *Technical Report* CSD-TR-93-076, Department of Computer Sciences, Purdue University.

Gao X S, Chou S C, 1998, "Solving Geometric Constraint Problems. I. A global propagation approach", *Computer Aided Design*, 30 (1) 47-54.

Gao X S, Chou S C, 1998, "Solving Geometric Constraint System. II. A Symbolic Approach and Decision of Rc-constructibility", *Computer Aided Design*, 30 (2) 115-122.

Heydon A, Nelson G, 1994, "The Juno-2 Constraint-based Drawing Editor", SRC Research Report 131a.

Kondo K, 1990, "PIGMOD: Parametric and Interactive Geometric Modeller for Mechanical Design", *Computer Aided Design*, 22 (10) 633-644.

Kramer G, 1992, *Solving Geometric Constraint Systems* (MIT Press).

Light R, Gossard D, 1982, "Modification of Geometric Models Through Variational Geometry", *Computer Aided Design*, 14 (4) 209-214.

Owen J C, 1991, "Algebraic Solution for Geometry from Dimensional Constraints", *In ACM Symp. Found. of Solid Modeling,* Austin TX*,* 397-407.

Requicha A, 1977, "Dimensionining and tolerancing", Technical report, Production Automation Project", University of Rochester, PADL TM-19.

Solano L, Brunet P, 1993, "A System for Constructive Constraint-based Modeling", *Modeling in Computer Graphics*, B. Falcidieno and T. Kunii editors, Springer Verlag.

Sunde G, 1988, "Specification of Shape by Dimensions and Other Geometric Constraints*", In Geometric Modeling for CAD Applications*, North Holland, IFIP, 199-213.

Sutherland I, 1963, "Sketchpad, a Man-machine Graphical Communication System", *In Proc. of the Spring Joint Comp. Conference*, 329-345, IFIPS.

Todd Ph, 1989, "A k-tree Generalization that Characterizes Consistency of Dimensioned Engineering Drawings", *SIAM J. DISC. MATH.*, 2(2) 255-261.

Van Sommers, 1984, *Drawing and Cognition* (Cambridge University Press, Cambridge).