Interactive Variation of Geometric Shapes Inside a Constraint-Based Environment

Simon Kolmanič and Borut Žalik

University of Maribor

Faculty of Electrical Engineering and Computer Science

Smetanova 17, SI-2000 Maribor, SLOVENIA

tel: ++386 62 25-461, fax: ++386 62 225-013

e-mail: simon.kolmanic@uni-mb.si, http://www.uni-mb.si/~cgai/simon.html

Abstract: In this paper, a 2D constraint-based drawing system which distinguishes between visible and auxiliary geometry is presented. The auxiliary geometry is fixed by the geometrical constraints and represents a framework on which the entities of the visible geometry are mapped. In continuation of the paper, the used constraints and the parametrisation of the geometric objects are presented. The parametrisation of geometric objects expires automatically as the user adds new constraints to the object. With change of these parameters, the user can interactively create an instance of the object. So the instance generation is made very simple for the user.

Key words: Geometric modelling, constraint-based system, variation of geometry, interaction.

1. Introduction

A time between new series of new artefacts in today's competitive industry is shorter and shorter. A design time is without a doubt the critical part in a production process and therefore, computers have been intensively involved into this process. For this purposes, appropriate computer software - geometric modellers - have been developed. Although these programs offer speed, reliability and versatile construction possibilities, the classical solutions of geometric modellers have remarkable drawback (see for example [ARBA89]). The inability to express spatial relationships between presented geometrical objects or elements is without a doubt one of the most critical. Inside the classical geometric modeller it is not possible to explain, for example, that a screw is placed in the middle of a hole. If the hole is moved, the screw has to be moved, too. However, this task has to be performed by the user itself without any help of the computer program although it could be well defined and therefore, executed automatically. In this way, a reusability, a new desired feature in design process, is highly limited to the simple geometric transformations.

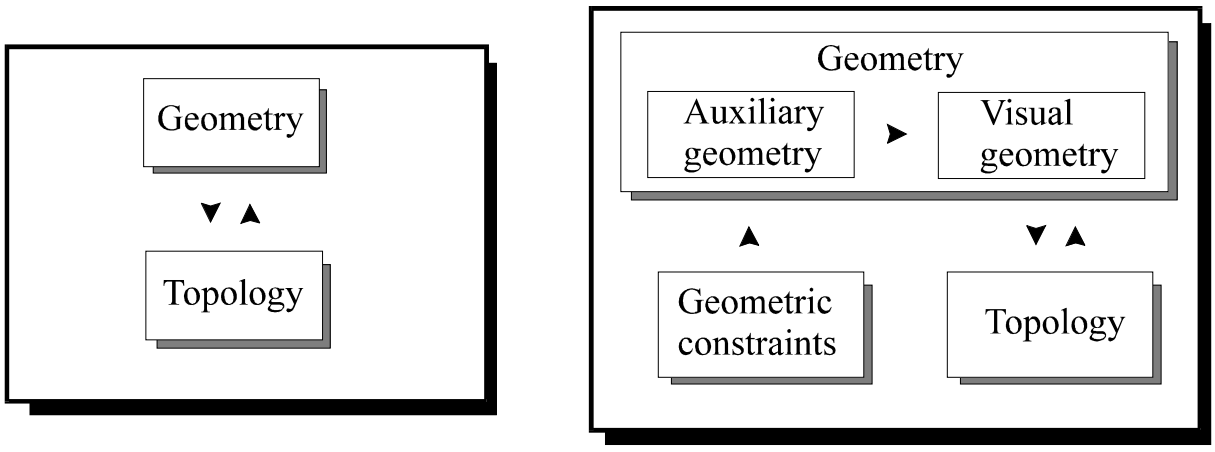
A new generation of geometric modellers tries to override mentioned drawbacks by introduction of feature-based [ROSS90] and constraint-based [ROLL89] paradigm. In our paper, an experimental 2D constraint-based geometric modeller is presented. Although being constraint-based, it is highly interactive and user-friendly what is achieved by splitting the geometry into visible and auxiliary part. The paper is organised into 6 chapters. In the second chapter, the division of geometry into visible and auxiliary geometry and difference between them are presented. In the third chapter, the geometrical constraints used in our system are described. In the fourth chapter, an interactive parametrisation and the instances generation is presented. The fifth chapter described our 2D constraint-based modelling system and how it

works. Here, the interactive constraint adding process and the interactive instances generation in our system are presented. The sixth chapter contains the conclusion of this paper. At the end of the paper, the references can be found.

2. Visible and Auxiliary Geometry

From the early days of geometric design, the geometry has been divided into a visible and an auxiliary part. With the introduction of computers into the design process, this division was lost. The classical representation of a geometrical object in a computer consists of topological and geometric data as shown in Figure 1a. The topology determines the neighbourhood relations between topological elements and the geometry, which is mapped onto the topological data, determines the position of corresponding geometric elements. However, this representation does not enable the user to define even simple spatial relationships, which are needed at construction. It is not possible to say, for example, that two line segments marked as ls_1 and ls_2 in Figure 2a lies on the same line l. If the slope of line l is changed, the slopes of both line segments have to be changed, too.

Because of this, in our system the geometry is divided into two parts - the visible and the auxiliary part [ŽALI96a] (Figure 1b). The topology again determines how the presented geometric elements of the visible geometry are connected. However, the actual position of the visible geometry is determined by the auxiliary geometry which is determined by the geometric constraints.



a) b)

Figure 1 a) A classical approach of representation of the shape of geometric object
b) A reformed representation of geometric object

Only the most elementary geometric entities are needed to express the auxiliary geometry. On the other hand, the visible geometry can take a form of any known well-defined form (e.g., line segment, B-spline, NURBS). In our system, points, lines, and circles define the auxiliary geometry, while line segments, circular arcs, and Bézier cubic are members of the visible geometry. Figure 2b explains how easy the distinguishing between the auxiliary and visible geometry can be done in the case of the Bézier cubic. The control points p_1, p_2, p_3, p_4 belong to the auxiliary geometry. The inner control points p_2 and p_3 are determined as the intersections between auxiliary lines. If the position (or slope) of any auxiliary line is changed the inner control points p_2 and p_3 get the new position and in this way, the whole shape of the Bézier cubic is changed.

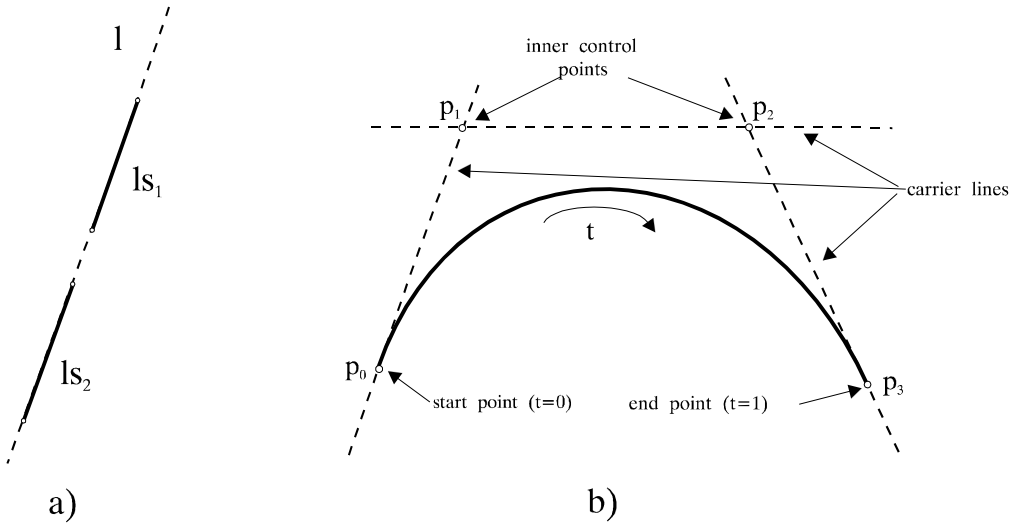


Figure 2 a) Line segments ls_1 and ls_2 lies on line l
b) Cubic Bézier curve with corresponding auxiliary geometry

All changes, that user wants to make, have to happen interactively. He has to be able to see the sketch of the designed geometrical object even if the auxiliary geometry is underconstrained, with other words, if there is not enough data to solve all spatial relations. System have to update auxiliary geometry each time the new constraint is obtained. In the next section the tool is represented that enables these operations.

3. The Use of Constraints in Order to Describe the Spatial Relations

To describe spatial relations between geometrical entities, geometrical constraints are used in our system. Constraints are expressed usually in a declarative way, that is done commonly by predicates [ALDE91, SUZU90]. The predicates in our system are divided in three groups:

- Dimensional constraints determine positions, coordinates, distances and angles. Their constituent part are variables which can be considered as the parameters of geometric object. Here is the list of dimensional constraints used in our system.

Point(p_i, x, y)	Point p_i gets absolute coordinates (x, y).
AngleValue(l_i, α)	Line l_i gets absolute value of its slope.
Distance(p_i/l_i, p_j/l_j, d)	Distance between points p_i and p_j (lines l_i and l_j) is d.
Angle(l_i, l_j, α)	Angle between lines l_i and l_j is α.

- Structural constraints determine the spatial relationships between geometrical entities. Those spatial relations are not changeable. If we require that two lines are parallel, the lines have to stay parallel forever.

HLine(l_i)	Line l_i is horizontal.
VLine(l_i)	Line l_i is vertical.
Through(l_i, p_j)	Line l_i passes through point p_j.
On(p_i, l_j)	Point p_i lies on line l_j.

Perpendicular(l_i, l_j)	Lines l_i and l_j are perpendicular.
Parallel(l_i, l_j)	Lines l_i and l_j are parallel.
Middle(p_i, p_j, p_k)	Point p_j is in the middle of points p_i and p_k.
Coincidence(p_i/l_i, p_j/l_j)	Points p_i and p_j (lines l_i and l_j) coincide.
Symmetric(p_i /l_i, p_j /l_j, p_k /l_k)	Points p_i and p_k (lines l_i and l_k) are symmetric regarding to point p_j (line l_j).

By using the dimensional and structural constraints, spatial relations are successfully described in a declarative way by the user. Such description seems to be close to the human way of thinking. Predicates of dimensional and structural constraints are sufficient for designing a shape of a geometric object, but in time we establish that we need another constraints for efficient parametrisation of the geometric objects. With existing predicates the parametrisation of geometric objects is not the best. We add therefore more constraints, namely pure numerical predicates.

- By the predicates of pure numerical constraints, simple numerical relations between parameters of dimensional constraints are expressed.

Product(a, b, c)	$c = a * b$;	$a = c * b^{-1}$;	$b = c * a^{-1}$;
Sum(a, b, c)	$c = a + b$;	$a = c - b$;	$b = c - a$;

With the pure numerical constraints we can, for example, define that two distances (let us call them d_1 and d_2) are related with each other. If distance d_2 is changed, then the distance d_1 should be also changed automatically according to the established constraint. In this way, the connection between parameters of two or more dimensional constraints is made.

Each inserted constraint has to be, of course, solved somehow. In our system we use a local propagation of known states [LELE88, FREE90] and its implementation is done by a Biconnected Constraint Description Graph - BCDG [ŽALI96b]. BCDG is very flexible structure which enables an interactive inserting of constraints and supports the work with completely and partially constrained geometric objects.

4. An Interactive Parametrisation of Geometric Objects and Generation of Instances

In real situations, a huge number of constraints have to be inserted by the user [KURL93] to constrain a geometric object completely. This task is extremely demanding because the user has to insert manually an exact number of non-conflict constraints. If the constraining process is not interactive, this task is even harder. Quite opposite, our system is completely interactive and it solves each inserted constraint as soon as possible and shows the results to the user. To reduce the number of manually inserted constraints, the system observes the actions of the user and tries to extract all self-understandable constraints automatically [ŽALI96a].

The constraining of a geometric object can be done in different ways giving in the final stage different descriptions with different parameters which control the shape of the geometric object. To generate an instance, in different situations, different parametric schemes are useful. To override this problem, our system supports the work with partially constrained geometric objects. This means that during the design stage, only some key parameters have to be defined. For example, in figure 3, a geometric object - smoking pipe - is shown where parameters d_1, d_2, d_3, and α are introduced. A closer look at figure 3 shows that point p_8 is not constrained at all. Despite this, in our system we can work normally with such geometric

object and even change interactively defined parameters. The user can add the missing constraints later, when he needs them.

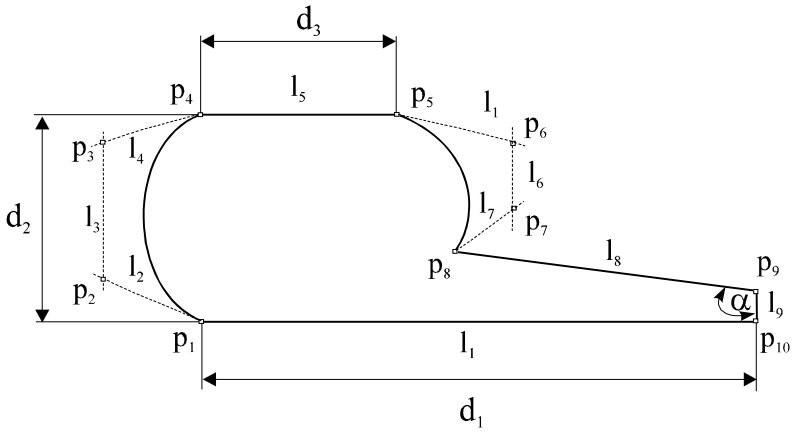


Figure 3. Smoking pipe, with some key parameters

How does our system behave when it works with a partially constrained geometric object? In general, inserted constraints tried to be solved and corresponding geometric entities are modified appropriately. Those geometric entities which are not yet related with geometric constraints take initial approximate values. For example, if in figure 3, the distance d_1 is changed and a constraint propagation starts at the point p_1, only the points p_9, p_{10} are moved. Other points stay at the same position including the point p_8. The point p_8 can be constrained, for example, with defining the distance between points p_8 and p_9 (let us name it d_4). To prevent the required shape of the pipe, an numeric constraint binding distances d_1 and d_4 has to be established.

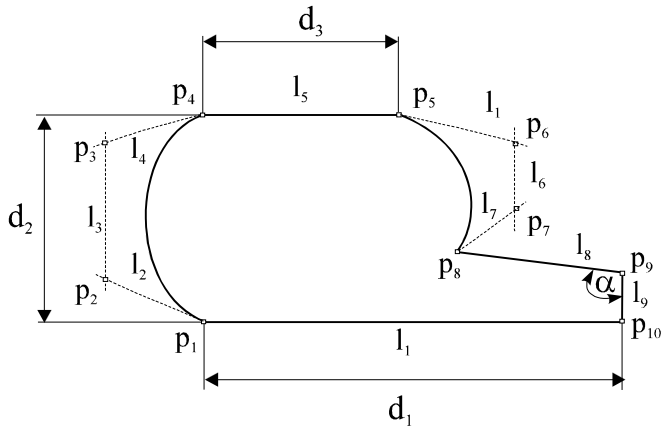


Figure 4 An instance of the smoking pipe after we reduce the distance d_1

Now, if the distance d_1 is changed, the distance d_4 has to be changed, too, according to established numeric constraint. In our system, the same result is achieved without introducing these two constraints by using the initial geometric data.

5. An Interactive Generation of Instances in our System

In this section, the use of our system is presented. The main window in our application is the drawing window where the user can design the geometric object. In this window, the tools for drawing geometrical entities of visual and auxiliary geometry such as points, lines, line

segments, Bézier cubic and arcs are available. The user can design geometric objects exactly in the same way as in the classical hand-made blueprints, or he can start with an sketch of the geometric object that is changed then to the desired form. In both cases, the geometrical constraints are needed to describe spatial relationships between geometrical entities.

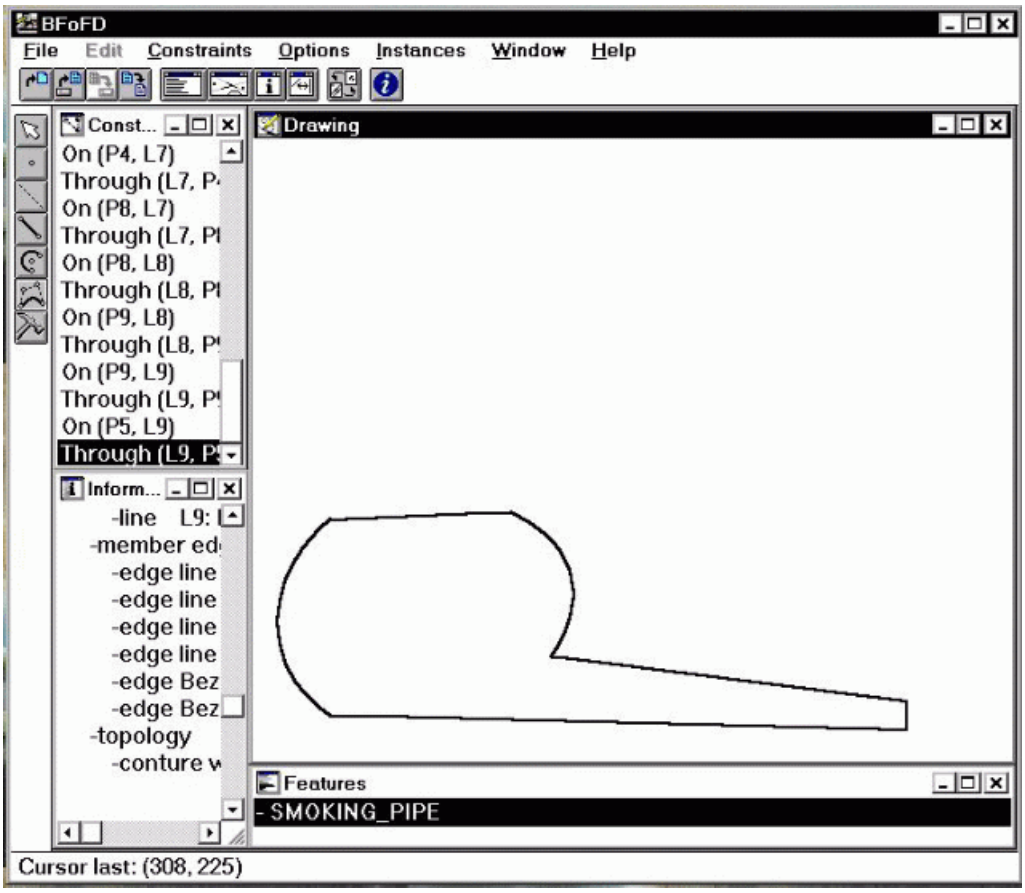


Figure 6 The sketch of geometric object smoking pipe

The other windows shown in figure 6 contain textual data about topology, used constraints and the name of edited geometric object, and are not necessary for designing of the geometrical objects. In drawing window, in figure 6, a sketch of a geometric object smoking pipe can be seen. The pipe is partially constrained with self-extracted structural constraints. All the dimensional constraints needed for the parametrisation of the geometric object still have to be added by the user manually. This is done by picking of the command for adding the constraints in systems menu. A dialog box with a list of available constraints appears on the screen after that (see figure 7). The user chooses the needed constraint in this list, and select the geometric entities that will be connected by selected constraint. Result of this operation is visible on the screen immediately.

The geometric object smoking pipe have not the desired shape in the drawing window in the figure 6. Lines l_1 and l_5 (see figure 3) have wrong slopes. Therefore, we add the constraint *HLine*, to repair the slope of line l_1. The slope of line l_5 was also set to 0° automatically, as we define that lines l_1 and l_5 are parallel. As the slopes of lines l_1 and l_5 were corrected, the dimensional constraints were added. Result of described constraining process can be seen in figure 7.

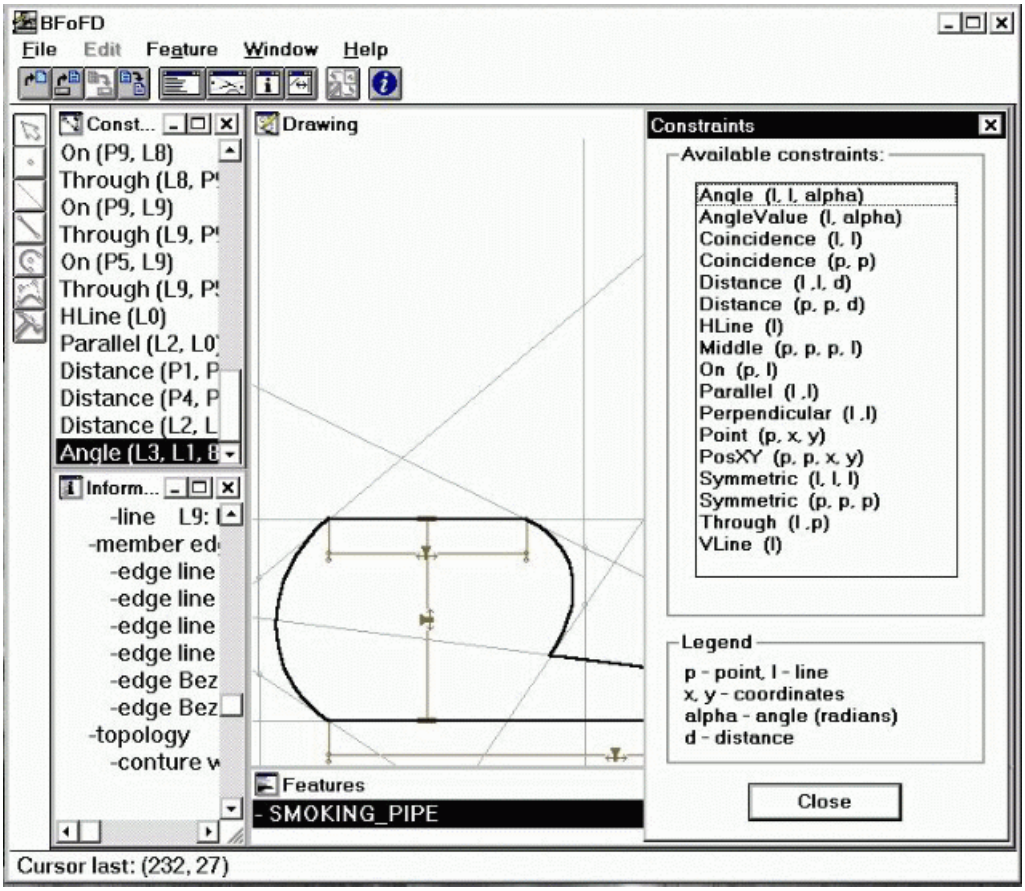


Figure 7 The geometric object Smoking pipe after we repair the lines

There are some symbols visible in the drawing window in figure 7. Let us explain their meaning. By designing our system we tried to enable the user so easy building of instances, as adding constraints to the object. For that purpose, we establish the special marks for all constraints important for the user. These marks and the constraints that are represented with them are presented in table 1. The marks have two meanings. First, they can be used to show all the constraints important for the user on the screen in the design process. This can be helpful to the user for the better parametrisation of the geometric object, because in the window of used constraints, there are many constraints like *On* and *Through* that are not important for the parametrisation. In the figure 7, such use of marks of geometric constraints is presented.

Geometric constraint	Mark
Angle(l_i, l_j, α)	
AngleValue(l_i, α)	
Distance(p_i/l_i, p_j/l_j, d)	
Parallel(l_i, l_j)	
Perpendicular(l_i, l_j)	
Symmetric(p_i /l_i, p_j /l_j, p_k /l_k)	S
Point(p_i, x, y)	

Table1 - Marks for some geometric constraints in our system

The second and more important use of symbols in our system is meant for increasing of interactivity by the instances generation process. As the user choose command for instances generation, he has to select one geometric entity that will be changed. Marks for all constraints, associated with selected geometric entity appear on the screen. When the user pick the symbol of dimensional constraints, the dialog box for attribute changing appears on the screen (see figure 8). Attributes can be easily changed by the scroll bar. After the user click on the button Apply, the changes are visible on the screen. With button Ok, the changes are confirmed.

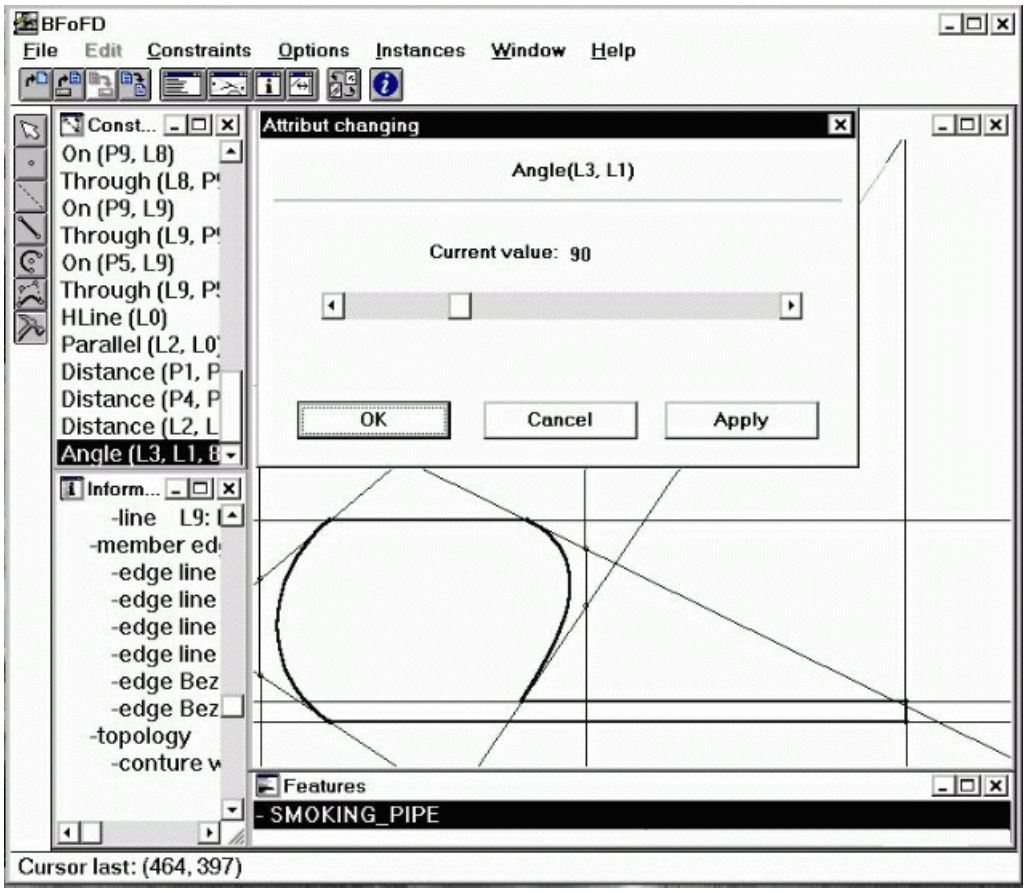
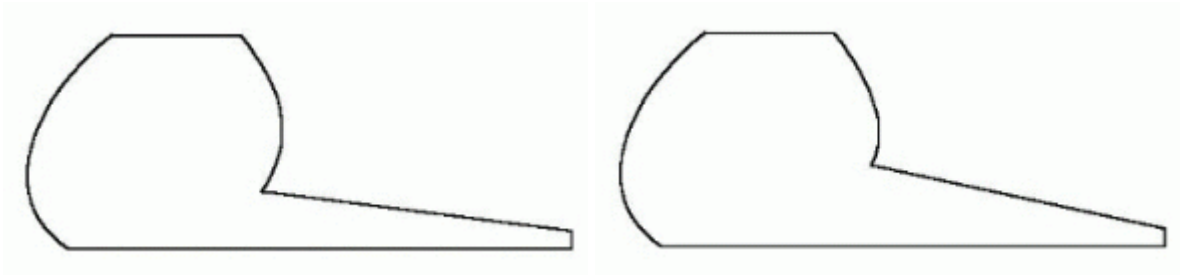


Figure 8 Interactive building of instances with use of scroll bar

All dialog boxes for attribute changing look similar. They have one scroll bar, except the dialog box for changing attributes for constraint *Point*. There are two scroll bars in that dialog box, for the each coordinate one. A generation of instances is made very simple with use of such dialog boxes.

In order to change the shape of geometric object smoking pipe, we change the angle between line l_8 an l_9. The proceedings of this change was next: First, the command for instance generation is selected. Then the line l_8 and mark for angle between two lines are selected. After that, the dialog box for attribute changing appears. In the dialog box, we change the attribute for the angle between lines. We set the value of the angle on 90° and press the Apply button. Situation after that can be seen in figure 8. The parameters of the selected constraint can be changed so many times, that the user is satisfied with the shape of the designed geometric object. After the changes are accepted by the user, he will work in the rest of the design process with changed shape of geometric object. For example, we reduce distance d_3 on our geometric object smoking pipe, and accept this change. The result of this

change can be seen in figure 9a. But we are still not satisfied with look of our object, therefore we increase angle α and accept its change. Result of this operation is presented in figure 9b. Both the pipes in figures 9a and 9b look similar to that in figure 6, but in classical drawing systems, it is quicker to design their shapes from the beginning than to modify the pipe in figure 6. In our system, these two new shapes are easily generated from the one in figure 6 by changing the values of two parameters only.



a) b)

Figure 9 a) Smoking pipe after we change attribute of constraint *Distance*
b) Smoking pipe after we change attribute of constraint *Angle*

The object smoking pipe was under-constrained in presented example. This means that the exact values of position of some geometric entities could not be exactly determinate. Despite this, the instance generation is possible. This is very important feature, because it enables to use the under-constrained objects for composing them to the complex geometrical objects. In this way, the reusability of geometric objects is increased. Beside this feature, our system automatically adds the self-understandable constraints, such as *On* and *Through* to the objects and the user can see all the changes made by constraints on the screen at any time.

6. Conclusion

In this paper, an interactive 2D constraint-based modelling system was described. It distinguishes between two types of geometry. The auxiliary geometry is controlled by geometric constraints, and the visible geometry is built on the framework defined by the auxiliary geometry. The auxiliary geometry has been used widely in the past at hand-made blueprints, but it is not handled at all in the present drawing packages. In our system, the auxiliary geometry is very important for its working. The constraint solver used here tries to calculate only the positions of the entities of the auxiliary geometry. Positions of the visible geometry are fixed then with position of the auxiliary geometry. If some geometric entities are not yet related with geometric constraints, so that their positions cannot be calculated, the system use the initial approximate values. In this way the terms well-constrained and under-constrained loose their meaning. For the system, it is not important if the geometric object is well-constrained or not, the user can in both cases interactively change the shape of the object. Therefore it is possible to make instances of even partially-constrained objects. Beside that, our system automatically extracts all self-understandable constraints by following users actions. The number of constraints required from the user is importantly reduced in this way.

Presented system has been written in C++ and runs on personal computers under MS Windows. Our future work will be concentrated on interactive tools for linking individual form features into final geometric object.

References

ALDE91 Aldefeld, B., Malberg, H., Richter, H., & Voss, K., (1991) Rule-Based Variational Geometry in Computer-Aided Design, in: Artificial intelligence in Design, D. T. Pham, (eds.), Springer Verlag, pp. 27-46.

ARBA89 Arbab, F., (1989) Examples of Geometric Reasoning in OAR, in: Intelligent CAD Systems II, V. Akman, P. J. W. ten Hagen, P. J., Veerkamp (eds.), Springer Verlag, pp. 32-57.

FREE90 Freeman-Benson, B. N., Maloney, J., & Borning, A., (1990) An incremental Constraint Solver, Communications of the ACM, Vol. 33, No. 1, pp. 54-63.

KURL93 Kurlander, D., & Feiner, S., (1993) Inferring Constraints from Multiple Snapshots, ACM Transactions on graphics, Vol. 12, No. 4, pp. 227-304.

LELE88 Leler, W., (1988) Constraint Programming Language, Addison-Wesley.

ROSS90 Rossignac, J.R., (1990) Issues on Feature-Based Editing and Interrogation of Solid Models, Computer & Graphics, Vol. 14, No. 2, pp. 149-172.

ROLL89 Roller, D., Schonek, F., & Verroust, A., (1989) Dimension-Driven Geometry in CAD: A Survey, in: Strasser, W. & Seidel, H.-P. (Eds.), Theory and Practice of Geometric Modeling, Springer Verlag.

SUZU90 Suzuki, H., Ando, H., & Kimura, F., (1990) Geometric Constraints and Reasoning for Geometrical CAD Systems, Computer & Graphics, Vol. 14, No. 2, pp. 211 - 224.

ŽALI96a Žalik, B, Kolmanič, S., & Podgorelec, D., (1996) A Drawing System Based on Separated Visible and Auxiliary Geometry, Advances in Computer-Aided Design, Proceedings of CADEX'96, Austrija, pp. 151 - 160.

ŽALI96b Žalik, B., Guid, N., & Clapworthy, G., (1996) Constraint-based Object Modelling, Journal of Engineering Design, Vol. 7, No. 2, pp. 209 - 232.