Development and application of the geometry constructions language to building computer geometric models

A A Boykov
MIREA – Russian Technological University, Moscow, Russia
E-mail: albophx@mail.ru

Abstract. Geometric models for industrial purposes are created in CAD systems making use of geometric constructions commands. At the same time, the so-called constructive or synthetic (as opposed to analytical) method to create a geometric model is implemented. The set of tools of any CAD system is limited, so accomplishing a specific task will require repeating a series of geometric constructions of the same type to form the desired shape. In this paper, a specific GC language (Geometry Constructions Language) is proposed to introduce an automated process of geometric modeling. To do this, theoretical issues of constructive geometry are analyzed, the main objects and syntax of the GC language are identified. The paper provides an example of building a model of a temple dome using GC language translator. It is noted that the GC language allows geometric modeling of regular curves (including trajectory of objects) and surfaces and applying new geometric correspondences and transformations. At the same time, geometric models in CAD systems are created faster since you don’t have to repeat manually the same type of construction or develop new equations and formulas for calculations.

1. Introduction
Geometric models can be created based on analytical and constructive (synthetic) methods. The analytical method involves calculating numerical values (of coordinates of points, coefficients of equations, etc.) describing the modeled object. The constructive method is used for building models of geometric shapes (objects) to satisfy given conditions making relations of newly built shapes with existing ones.

CAD systems enable users to create a geometric model of an object (constructive method). However, the set of tools of any CAD system is limited, so accomplishing a specific task will require repeating of a series of geometric constructions of the same type [1] or apply the analytical method [2, 3] converting geometric data to numerical data and vice versa. In both cases, the complexity of creating a geometric model of an object and margin of error increase.

Using CAD scripting languages is a viable alternative. In this case, geometric modeling requires specific knowledge beyond geometry, such as syntax of the language, architecture, internal data model of the CAD system, features of working with memory, data types, etc.

In this paper, a specific GC language is proposed. The GC language is intended to formalize and thereby simplify the automation of a synthetic method of computer geometric modeling.
2. Problem definition

Existing GC languages allow producing high-quality images of geometric models (usually two-dimensional) for printing or they allow formalizing geometric modeling for automated geometry theorem proving [4, 5]. In [6], GC language is proposed for the development of computer graphics. A more detailed review of methods to the symbolic description of geometric models based on constructions was previously discussed in [7]. Existing languages have some disadvantages which are given below:

- They don’t allow setting and using geometric correspondences.
- They don’t allow applying a geometric construction instantly to a set of objects.
- Compass and ruler, i.e. straight lines and circles, are considered as tools, while in practice some other tools, such as spline curves, are often used.
- They don’t allow creating geometric models for CAD systems.

Thus, there is a practical need to develop a GC language for CAD systems, devoid of these disadvantages. It will require the following solutions:

- Considering the fundamentals of constructive geometry in relation to CAD systems practice.
- Describing the elements and syntax of the GC language.
- Creating a translator prototype and considering the use of the GC language in the process of geometric modelling.

Let’s discuss the ways to meet these objectives.

3. Theory

Constructive geometry is usually related to planimetry and stereometry. Geometric models in CAD system are created in Euclidean flat or three-dimensional space, so it is quite reasonable to try to combine the constructive geometry with the use of CAD systems. Based on theoretical issues of constructive geometry [8, 9] considering the peculiarities related to CAD systems, the requirements for the GC language have been defined and the structure of its sentences has been proposed and will be described below in this section.

In constructive geometry, each geometrical construction solves a problem of construction and allows a modeler to draw a desired shape according to given conditions. In constructive geometry, a shape is defined as a set of points such as a single point, a line segment, a line, a circle, etc. Creating geometric models in CAD system is similar to drawing shapes solving a problem of construction. A geometric model can be considered as a set of geometric objects or a shape in its broad sense. We will refer to an “engineering geometry problem of construction” as the creation of a geometrical model in a CAD system according to given conditions.

In constructive geometry, the solution of the problem is reduced to the consistent application of axioms from a certain set. They are considered as primitives or elementary problems of construction.

R1 Requirement. The instructions of the GC language allow describing the creation of basic shapes, their modification and erasing.

To add an object to the model, we will use the following syntax:

$id_1 = \text{command}_1 (\text{parameters})$

To change (including erase) an object, we will use the following syntax:

$\text{command}_1 (id_1, \text{parameters})$

where $id_1$ is the internal name of the shape, and $\text{command}_1$ is the name of the geometric command.

R2 Requirement. The GC language allows specifying exactly the type of the basic shape when creating it, as well as any previously constructed object or its significant element (center, focus, etc.). In the language syntax, you can indicate exactly the type of the newly created object (defined by the command name), the previously constructed object (defined by the internal name), or the significant
elements of the object. To specify the elements of the object, use the “.” symbol: object1.center, object1.radius, etc.

When the sequence of instructions leading to the solution of the problem is found, it can be used as a new geometric command to solve some other problems.

**R3 Requirement.** The GC language has the means to describe the solution to the problem and introduce it as a new geometric command.

To represent the sequence of instructions in the form of a newly built geometric construction, the following syntax is used:

```
sub id1 (parameters)
...
ret (returned objects)
end sub
```

Here, id1 is the internal name of the newly built geometric construction, which can be used as the command name later. For simple cases, a short form can be used:

```
id1 (parameters) = command1 (... commandn (...) …)
```

The solution of the problem is almost always approximate. For example, calculations when building a shape in CAD systems are implemented on the basis of numerical methods [10], and when creating a shape there is a need of use a piece of a curve or surface which is not among the basic ones, such curves and surfaces are replaced with some accuracy by piecewise smooth (spline) ones. Therefore, dealing with practical problems in CAD systems where automation of geometrical construction is possible some graphic techniques that were not considered as sufficiently accurate still can be applied.

**R4 Requirement.** The GC language works with generalized shapes (spline curves and surfaces) as with basic ones.

Splines (generalized curves) are defined by lists of points:

```
curve1 = spline (parameters, list1)
```

If a command is executed with a curve as a parameter, where a point is expected, it is replaced by a group of points, and the command is applied to each of them, and a new spline is created.

On the other hand, constructive geometry allows a modeler to solve problems related to some particular shapes without the presence of the shapes themselves. To do this, the shape determinant, which substitutes it in the constructions, should be used.

**R5 Requirement.** The GC language allows describing geometric determinants of a shape to be used in constructive algorithms.

The syntax for a shape with a determinant is as follows:

```
id1 = new (element1 = value, element2 = value, …)
```

After executing this command, the elements of the determinant will be available using “.” symbol: id1.element1, id1.element2, …

Since the shape can be defined by different determinants, an “if” statement is used to ensure the choice of the solution of the problem:

```
if (condition1, construction then1,1 : condition2, construction then2,2 : ... : construction else)
```

In constructive geometry, the solution of a problem is not only the target shape, but a chain of related geometric shapes from the original to the target. If one of the original shapes is moved, then the target shape and all the intermediate shapes are changed by means of constructive relations.

The solution of the problem is an immediate geometry machine [9], the performance of which is similar to a parameterized model of CAD systems. Many of the features that are common for geometric machines are still not used in CAD systems. For example, it is impossible to define geometric correspondences and transformations and to apply some geometric operation defined for an element (for example, a point) to a set of such elements (to a curve) at once.

**R6 Requirement.** The GC language allows geometric operators to be applied to a set of shapes.

A list is used to input a finite collection of objects:

```
list1 = [object1, object2, ...]
```

If the command returns several shapes (for example, the set of intersection points), then a list can be used to separate them:
\[ id_1, id_2, \ldots ] = \text{command}_1 \text{ (parameters)} \]

If a list is used as a command parameter, the command is applied to all its elements, in this case a list is also returned.

To perform some actions with the objects in the list, a loop construction can be used:

\[
\text{for (object)} \text{ in [list]} \\
\text{...} \\
\text{end for}
\]

The list item counter is accessible using “$” symbol: object, and list, $ have the same meaning.

To divide the curves into sets of points, the following type of loop should be used:

\[
\text{for (number of points)} \text{ point}_0 \text{ in [curve]} \\
\text{...} \\
\text{end for}
\]

R7 Requirement. The GC language allows describing geometric correspondences and new geometric transformations.

The correspondence between two point rows on the carriers curve_1 and curve_2 is established by the following loop form syntax:

\[
\text{for (number of points)} \text{ point}_1 \text{ in [curve]}, \text{ point}_2 \text{ in [curve]} \\
\text{...} \\
\text{end for}
\]

If the point match is given by a syntax:

\[
\text{sub correspondence}_1 \text{ (point)} \\
\text{ point}_1 = \ldots \\
\text{ret (point)} \\
\text{end sub}
\]

then you can create a new curve by mapping the original one (as a spline):

\[
\text{curve}_2 = \text{prow (correspondence}_1 \text{, curve)}
\]

– where curve_1 is the original basic or generalized curve.

Geometric transformations are a common part of GC language. The transformation is created as an object:

\[
\text{transformation}_1 = \text{command}_TR \text{ (options)}
\]

It can be used to transform other shapes:

\[
\text{list} = \text{trobj (transformation}_1 \text{, object}, \text{ object}, \ldots ) \text{ // use direct transformation} \\
\text{list} = \text{trobi (transformation}_1 \text{, object}, \text{ object}, \ldots ) \text{ // inverse transformation}
\]

Transformations preserve the correspondence of the basic types. If the correspondence cannot be maintained, a generalized curve will be created.

The elements of the GC language discussed above are based on constructive geometry in general and are the most common ones. They define the structure of language syntax and are not limited by the dimension of the space (two- or three-dimensional space) and the set of commands. Choosing the dimension of space in which a construction is made and the set of commands, we will get the subject GC language for solving practical problems.

4. Results of experiments

Let's consider the application of the GC language in practice and we will limit ourselves to a shape on the plane. The software implementation of geometric commands is simpler in this case; however, such language can be used for three-and above dimensional modeling [11]. We have created a simple translator and implemented about 50 commands, some of which are shown in Table 1. The translator works in interpreter mode (JavaScript is used) in a Web browser.

There is particular interest in the creation of computer model domes of cathedrals and temples [1].

Figure 1(a) shows one of the onion-shaped domes of the Cathedral of the Intercession of the Most Holy Theotokos (also known as St. Basil's Cathedral). In [1], it is proposed to create a model of such a dome based on Dupin cyclides. Having a single element consisting of fragments of two cyclides (Figure 1(b)), the dome model can be obtained using standard CAD-system tools. However, the Dupin cyclide is not among the basic objects of CAD systems. Creating a geometric model of a cyclide
requires construction of its circular generatrices, essentially, manually. The cyclide model will be approximate in this case.

Table 1. Geometry constructions language commands.

| Command | Description | Using |
|---------|-------------|-------|
| pxy     | Point with given coordinates | pxy (x, y [, w]) |
| pab     | Point at the intersection of two given straight lines | pab (linea, lineb) |
| psp     | Point on a given curve specified by given parameter value | psp (curve, parameter) |
| pac     | Point at the intersection of a given straight line and a given circle | pac (line, circle) |
| pac2dir / pac2nd | Second closest / most distant point at the intersection of a straight line and a circle | pac2dir / pac2nd (line, circle, pointto/from) |
| sab     | Straight line through two given points | sab (pointa, pointb) |
| soa / spa | Straight line perpendicular/parallel to the given one | soa / spa (line, point) |
| ccx     | Circle with given centre and radius / point / tangent line | ccx (center, radius / point / tangentline) |
| c2p     | Circle with given ends of the diameter | c2p (pointa, pointb) |
| a3p     | Arc of a circle defined by three given points | a3p (start, anypoint, end) |
| vr / hr | Vertical / Horizontal straight line | vr (point, ofs [, sign]) |
| prow    | Spline built by the given mapping of the given curve | prow (SUB, curve) |
| z       | Setting the shape’s z-coordinate | z (obj, zcoord) |
| !cout3d | Extracting the shape reference points for export to the CAD system | !cout3d (obj1, …) |
| !erase  | Erasing shapes | !erase (ob1, …) |

The shape of the element will be defined by parameters $h_0$, $R$, and $r$, as shown in Figure 1(b). The considered algorithm for constructing the model element is usually performed in a three-stage process:

1. Constructing new built shapes that control shaping.
2. Defining of construction of a single generatrix for one control point.
3. Building a structured point cloud when the control point is moved.

Let’s place the origin as shown in Figure 1(d); we assume that $O$, $v$, and $h$ are given. The construction of the outlines of the lower cyclide (circles $k_1$ and $k_2$) is written as follows:

$$[K1, K2] = pac (h, ccx (O, r))$$

$$[k1, k2] = ccx ([K1, K2], R)$$

To construct the outlines of the second cyclide, first construct the point $H_0$:

$$h1 = hr (h0, ‘.’)$$

$$H0 = pab (v, h1)$$

Using $H_0$, we construct the circles $o_1$ and $o_2$ that touch $k_1$ and $v$ or $k_2$ and $v$, respectively. Let’s introduce the following construction:

```
sub cscp (s, c, P)
  norm = soa (s, P)
  [S1, S2] = pac (spa (norm, c.center), c)
  [K1, K2] = pac2nd (sab (P, [S1, S2]), c, [S1, S2])
  [W1, W2] = pab (sab (c.center, [K1, K2]), norm)
```
\[ [o1, o2] = ccx ([W1, W2], P) \]
\[ ret (o1, o2, K1, K2) \]
\[ !erase (S1, S2, norm, W1, W2) \]
\[ end sub \]

**Figure 1.** Constructing the model of the dome element.

Construction ‘cscp’ (see Figure 2(c)) returns circles \((o1, o2)\) and tangent points \((K1, K2)\).

\[ [.., o1, .., R1] = cscp (v, k1, H0) \]
\[ [.., o2, .., R2] = cscp (v, k2, H0) \]

We create carriers for control points which represent two conjugate arcs \(a1\) and \(a2\):

\[ [a1, ..] = a3p (pac(v; k1) \{1\}, H0, R1) \]
\[ [a2, ..] = a3p (R1, H0, pac2nd (\ vr (R1), o1, R1) \) \]

Let’s consider the formation of a circular generatrix in three-dimensional space. The control point passes first through the arc \(a1\), then half of the arc \(a2\) (Figure 1(f)). For each position, we draw a projection of the section – the segment \(d0\) (Figure 1(d)). On the segment \(d0\), as on the diameter, we construct the circle \(s0\) which represents the generatrix in its true dimensions (Figure 1(e)). For every point \(P0\) on \(s0\), the spatial coordinates should be identified. To do this, we construct the perpendicular from \(P0\) to \(d0\). Now \(Pd\) contains the \(X\) and \(Y\) coordinates, and the signed distance from \(P0\) to \(d0\) is the \(Z\) coordinate. Let's introduce the instruction:

\[ sub circz (P0) \]
\[ Pd = pab (soa (d0, P0), d0) \]
z(Pd, md (d0, P0))
ret(Pd)
end sub

Let's go through the $a_1$ arc (in the example, the arc is divided into 22 pieces):

```plaintext
[oldP, _] = pac (v, k1)
for (22) P in (a1)
  d0 = sab (O, P)
  oldP = pac2dir (d0, k2, oldP)
  [s0, _] = c2p (P, oldP)
  !cout3d (prow (circz, s0))
end for
```

In a similar way, we will go through half of the $a_2$ arc (the example contains 12 pieces):

```plaintext
for (12) i in (1:0.5)
  P = psp (a2, i)
  ...
end for
```

The structured point cloud is exporting to the CAD system as a text file [11], where a spline surface is stretched over it (Figure 2(a)). Then the dome element is duplicated by standard CAD system tools (Figure 2(b)). The considered example shows that a one-view drawing is sufficient for the synthesis of a three-dimensional model based on a constructive method. By changing the initial values of the parameters $h_0$, $R$, and $r$, you can get other versions of the dome element and the whole dome.

A minor modification of the considered algorithm such as rotation of each circle by a fixed angle during the formation of its circle-skeleton (see Figure 2(c) the circles of the cyclide are shown in grey, the circles after the rotation are shown in black), allows getting a geometric model of the element for a different type of dome (Figure 2(d)). In this case, the so-called stratified geometric transformation is implemented.

![Figure 2. Construction of a computer model of the dome in a CAD system.](image)

5. Consideration of the results

Interpreters execute program code rather slowly. It takes about 10 seconds to create the structured point cloud given in the example. When the number of circles is increased up to 100, the processing time increases up to 30 seconds. However, manual construction of the same number of circular generatrix manually requires much more time. The program text represents a parameterized model and allows a modeler to change the initial values to get new solutions of the problem.

In addition, the described constructions can be used to solve any other problems. In particular, the construction “circz” can be used to form a structured point cloud of any cyclic surface according to a one-view drawing: it is sufficient to provide the construction of $d_0$ and $s_0$ for each circular generatrix.
6. Conclusions
To sum up, the aim is achieved. Based on the analysis of constructive geometry, a GC language has been proposed allowing automated construction of geometric models in CAD systems. Experiments show that the GC language, even with a comparatively small number of commands and translator, is an effective tool for obtaining geometric models and it also expands the capabilities of the CAD system. It becomes possible to create computer models of curves (including the trajectories of objects) and surfaces [12], and to use new geometric correspondences [2]. At the same time, repeated manual construction of a geometric model or using the analytical method is not required anymore.

There are no other previous studies on the developing a GC language for geometric modeling in CAD systems and on the application of constructive geometry to working in CAD systems. Prospects of the proposed method allow, on the one hand, expanding the set of CAD system tools (new built commands), and on the other hand, achieving closer integration of the GC language and CAD systems, for example, executing programs directly in a CAD system.

References
[1] Salkov N A 2020 Application of the Dupin cyclide in temple architecture J. of Phys.: Conf. Ser. 1546 012042
[2] Beglov I A 2020 Generation of the surfaces via quasi-rotation of higher order J. of Phys.: Conf. Ser. 1546 012032
[3] Karpov D A and Struchenkov V I 2019 Dynamic programming as a method of spline approximation in the CAD Systems of Linear Constructions Russian Technological J. 7 3 pp 77-88
[4] Janicic P 2010 Geometry Constructions Language J. of Automated Reasoning 44 pp 3-24
[5] Liang T and Wang D 2007 On the Design and Implementation of a Geometric-Object-Oriented Language Frontiers of Computer Science in China 1 2 pp 180-190
[6] Haiyan Y U, Hongming C A I and Yuanjun H E 2016 Geometric basis: a geometric solving cell for geometric computing Proc. 17th Int. conf. on geometry and graphics (China: Beijing) pp. 47-48
[7] Boykov A A 2019 The language of geometrical constructions and its applications in development of CAD-systems Proc. Int. conf. SCVRT-2019 pp 42-50
[8] Chetveruhin N F 2018 Geometric construction methods (Moscow: Lenand Publ.) p 152
[9] Val’kov K I 1974 Introduction to modeling theory (Lenigrad) p 152
[10] Golovanov N N 2002 Geometric modeling (Moscow: Phys.-math. lit. Publ.) p 472
[11] Boykov A A 2018 About building of models for objects in space of four and more dimensions in educational process Geometry and Graphics 5 4 pp 54-71
[12] Boykov A A 2018 Some issues of automation of creating curve lines and surfaces by constructive geometrical methods in CAD Proc. Int. conf. SCVRT-2018 pp 281-290

Source of Funding. Acknowledgements
The work was carried out within the framework of scientific topic No. 140-IRTS RTU MIREA.
The author thanks Professor N. A. Salkov for his advice on the use of Dupin cyclides.