GCLC — A Tool for Constructive Euclidean Geometry and More than That

Predrag Janičić

e-mail: `janicic@matf.bg.ac.yu`
Faculty of Mathematics, University of Belgrade
Studentski trg 16, 11 000 Belgrade, Serbia

Abstract. We present GCLC/WINGCLC— a tool for visualizing geometrical (and not only geometrical) objects and notions, for teaching/studying mathematics, and for producing mathematical illustrations of high quality. GCLC uses a language GC for declarative representation of figures and for storing mathematical contents of visual nature in textual form. In GCLC, there is a build-in geometrical theorem prover which directly links visual and semantical geometrical information with deductive properties and machine–generated proofs.

1 Introduction

GCLC is a tool for visualizing objects and notions of geometry and other fields of mathematics (by generating figures and animations). It can be used for producing digital mathematical illustrations, for teaching and studying geometry (and not only geometry), and for storing visual mathematical contents in textual form — as figure descriptions in the GC language. GCLC provides easy-to-use support for many geometrical constructions, isometric transformations, and conics. The basic idea behind GCLC is that constructions are formal procedures, rather than drawings. Thus, in GCLC, producing mathematical illustrations is based on "describing figures" rather than of "drawing figures".[1] This approach stresses the fact that geometrical constructions are abstract, formal procedures and not figures. A figure can be generated on the basis of abstract description, in Cartesian model of a plane. A similar approach is used for illustrations for other supported fields. Figures can be displayed and exported as LaTeX files or bitmaps.

Although GCLC was initially built as a tool for converting formal descriptions of geometric constructions into LaTeX form (hence its name "Geometry Constructions → LaTeX Converter"), now it is much more than that. For instance, there is support for symbolic expressions, for drawing parametric curves, for program loops, etc; WINGCLC, a version with a Windows graphical interface, makes GCLC a dynamic geometry tool with a range of additional functionalities; a built-in geometry theorem prover can automatically prove a range of complex

[1] In a sense, this system is in spirit close to the TeX/LaTeX system [14, 16], or is parallel to it. Within the TeX/LaTeX system, the author (explicitly) describes the layout of his/her text.

theorems, etc. GCLC now links semantic information about a construction with its visual representation and with its deductive properties. Thus, it provides mathematical contents directly linked to visual information and supported by machine–generated proofs.

GCLC is under constant development from 1996. Some features of graphical interface of WINGCLC are presented in [13], some educational aspects of GCLC are presented in [7], the built-in theorem prover is described in [23, 11], and the mathematical contents management issues are discussed in [24]. This paper is the first general overview of the system.

Overview of the paper: The rest of the paper is organized as follows: in Section 2 we focus on formal geometrical constructions and illustrate the need for describing mathematical illustrations rather then drawing them; in Section 3 we give a brief overview of the language of the GCLC system; in Section 4 we describe basic features of the graphical interface; in Section 5 we describe the built-in geometry theorem prover; in Section 6 we present several examples, illustrating different features of GCLC; in Section 7 we briefly discuss applications of GCLC in producing mathematical illustrations, in storing mathematical contents of visual nature, and in teaching mathematics; in Section 8 we discuss some technical issues and give availability information; in Section 9 we give a short overview of the systems related to GCLC; in Section 10 we discuss potential directions for further work and in Section 11 we draw final conclusions. In Section A we give some additional examples.

2 Describing Formal Constructions

Geometrical constructions are the main area of GCLC. This type of mathematical problems is very relevant for the need to describe, and not draw images.

A geometrical construction is a sequence of specific, primitive construction steps. These primitive construction steps are also called *elementary constructions* and they are:

- construction (by *ruler*) of a line such that two given points belong to it;
- construction of a point such that it is the intersection of two lines (if such a point exist);
- construction (by *compass*) of a circle such that its center is one given point and such that the second given point belongs to it;
- construction (by *compass*) of a segment connecting two points;
- construction of intersections between a given line and a given circle (if such points exist).

By using the set of primitive constructions, one can define more involved, compound constructions (e.g., construction of right angle, construction of the segment midpoint, construction of the segment bisector etc.). In describing geometrical constructions, it is usual to use higher level constructions as well as the primitive ones.

GCLC follows the idea of formal constructions. It provides easy-to-use support for all primitive constructions, but also for a range of higher-level constructions. (Although motivated by the formal geometrical constructions, GCLC provides a support for some non-constructible objects too — for instance, in GCLC it is possible to determine/use a point obtained by rotation for 1°, although it is not possible to construct that point by ruler and compass).

There is a need of distinguishing abstract (i.e., formal, axiomatic) nature of geometrical objects and their semantics and usual models. A geometrical construction is a mere procedure of abstract steps and not a picture. However, for each (Euclidean) construction, there is its counterpart in the standard Cartesian model. While a construction is an abstract procedure, in order to make its usual representation in Cartesian plane (or, more precisely, in Cartesian model of Euclidean plane), one still has to make a link between these two. For instance, given three vertices of a triangle, one can construct a center of its circumcircle, but in order to visualize and represent this construction in Cartesian plane, he/she has to take three particular Cartesian points as vertices of the triangle (see the example given in Fig. 1). Thus, figure descriptions in GCLC are usually made by a list of definitions of several (usually very few) fixed points (defined in terms of Cartesian plane, i.e., by pairs of coordinates) and a list of construction steps based on these points.

3 GC Language

In GCLC, figures are described in the GC language. GC syntax is very simple, but, at the same time, it enables describing very complex figures in very few lines. Describing images in the GC language does not require programming skills. Descriptions via GC commands directly reflect mathematical objects to be visualized and are easily understandable to mathematicians. Therefore, GC is a higher-level language (with support for a number of advanced geometrical concepts) designed for mathematicians, and not a machine-oriented script language.

GC language consists of the following groups of commands (examples for different groups of commands are given in Section 6):

Basic definitions: these commands include commands for defining fixed points, for defining a line on the basis of two selected points, defining a circle, a numerical constant etc.

Basic constructions: these constructions include constructions of intersection points for two lines, and for a line and a circle, construction of the midpoint of a given segment, the bisector of an angle, the segment bisectors, perpendicular lines, parallel lines, etc.

Transformations: these commands include commands for translation, rotation, line-symmetry, half-turn, but also some non-isometric transformations like scaling, circle inversion etc.

Commands for calculations, expressions, and loops: there are commands for calculating angles determined by triples of points, distances between points, for generating random numbers, for calculating symbolic expressions and support for *while*-loops.

Drawing commands: there are commands for drawing lines, segments, circles, arcs, and ellipses in several modes.

Labelling and printing commands: points can be labelled, marked in a number of ways. In addition, a text can be attached to a particular point.

Cartesian commands: this group of commands provides support for direct access to a user–defined Cartesian system. A user can define a system, its unit, and, within it, he/she can define points, lines, conics, tangents etc. and can also draw curves given in parametric form.

Low level commands: there is support for changing line thickness, color, clipping area, figure dimensions etc.

Commands for describing animations: this group of commands provides support for making animations within WINGCLC. Some points can be defined to move from one position to another; points can also be traced.

Commands for the geometry theorem prover: using support for the built-in geometry theorem prover, the user can provide the conjecture, can control a proof level and can limit a maximal number of proof steps.

4 Graphical Interface

WINGCLC provides a range of interactive functionalities. In addition to tools for processing picture descriptions and locating errors, tools (*watch window*) for monitoring values of selected objects in a construction (so WINGCLC can work as a *geometrical calculator*), there are also tools for easy and interactive moving of fixed points, updating pictures and making animations. (Animations and traced points can be defined both interactively and via GCLC commands.) These interactive features can be very useful in teaching geometry, but can also help studying geometry or even help some research (with WINGCLC serving as a machine assistant). Figure 5 illustrates some of the mentioned tools and devices (traces, animations, *watch windows*, etc.)

5 Theorem Prover

Automated theorem proving in geometry has two major lines of research: synthetic proof style and algebraic proof style (see, for instance, [18] for a survey). Algebraic proof style methods are based on reducing geometric properties to algebraic properties expressed in terms of Cartesian coordinates. These methods are usually very efficient, but the proofs they produce do not reflect the geometric nature of the problem and they give only a *yes* or *no* conclusion. Synthetic methods attempt to automate traditional geometry proof methods.

The geometry theorem prover built into GCLC is based on the area method [3, 4, 23].[2] This method belongs to the group of synthetic methods. It produces

[2] The theorem prover is developed in collaboration with prof. Pedro Quaresma from University of Coimbra.

traditional, human-readable proofs, with a clear justification for each proof step. The main idea of the method is to express hypotheses of a theorem using a set of constructive statements, each of them introducing a new point, and to express a conclusion by an equality of expressions in *geometric quantities* (e.g., signed area of a triangle), without referring to Cartesian coordinates. The proof is then based on eliminating (in reverse order) the points introduced before, using for that purpose a set of appropriate lemmas. After eliminating all introduced points, the current goal becomes a trivial equality that can be simply tested for validity. In all stages, different expression simplifications are applied to the current goal. The method does not have any branching, which makes it very efficient. A wide range of geometric conjectures can be simply stated within GCLC and proved by the prover.

The prover is tightly integrated in GCLC. This means that one can use the prover to reason about a GCLC construction (i.e., about objects introduced in it), without changing and adapting it for the deduction process — the user only needs to add the conclusion he/she wants to prove. The proofs are generated in LaTeX form. For more details about the prover, see [23, 11].

6 Examples

Geometrical constructions. The example given in Fig. 1 illustrates one simple geometrical construction. Groups of commands are explained by comments (marked by the symbol `%`) within the description itself. As many other similar descriptions, this one has basically three parts (not necessarily separated): one with defining fixed points (with coordinates in Cartesian plane), one with construction steps, and one with labelling and drawing commands. By changing one of the three fixed points, the whole of the illustration is updated. In this example, three side bisectors of the triangle ABC are constructed. It is a simple fact that these three lines intersect at one point (at the center of the circumcircle). This can be also stated in the following form: pairwise intersections of the side bisectors, the points `O_1` and `O_2`, are identical. This property (as well as much more complex properties or hypotheses) can be, in a sense, explored within GCLC. Namely, `d` is defined to be the distance between `O_1` and `O_2`, and one can monitor the value of `d` to ensure that it is equal to zero (for these and for any other three particular vertices).

Cartesian commands. Example given in Fig. 2 illustrates the support for a direct access to a user-defined Cartesian system. In this example, there is a description of one conic (parabola), via its canonical parameters, and one its tangent. This example also illustrates how a rather complex figure can be described in only a few lines.

Parametric curves. Example given in Fig. 3 illustrates the support for parametric curves. The first curve, is drawn for parameter x ranging from -3 to 4, increased by the step 0.05.

```
% fixed points
point A 10 10
point B 50 10
point C 40 50

% side bisectors
med a B C
med b A C
med c B A

% intersections of bisectors
intersec O_1 a b
intersec O_2 a c
distance d O_1 O_2

% marking points
cmark_b A
cmark_b B
cmark_t C
cmark_lt O_1
cmark_rt O_2

% drawing the sides of the triangle ABC
drawsegment A B
drawsegment A C
drawsegment B C

% drawing the circumcircle of a triangle
drawcircle O_1 A
```

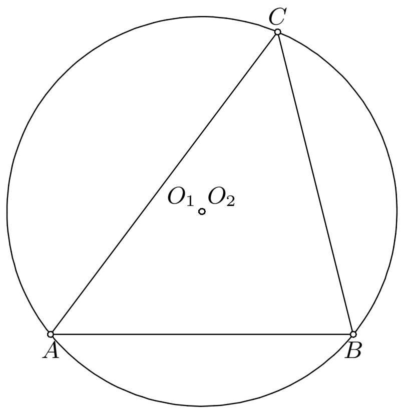


Fig. 1. Example of a GC description of a geometrical construction (left) and the corresponding (LaTeX) output (right).

While-loops. Example given in Fig. 4 illustrates while-loops. The construction described within this example shows that, for any line segment AB, the locus of all points L such that the angle ALB is right angle, is the circle with the perimeter AB. The point B is rotated (giving the point B') around the point A for the angle *phi* ranging from $0°$ to $70°$, and the point L is determined as a foot of the perpendicular from B to AB'. Points L for different values of *phi* are connected by line segments.

Animations. An animation in WINGCLC is defined as a formal construction with a set of fixed points that linearly move from an initial to a destination position. All positions of one selected point make *trace* (similar to *locus*), drawn in a selected color. The *watch window* is used for monitoring values of objects used in the construction. The screenshot shown in Fig. 5 illustrates some of the features and devices of WINGCLC (traces, animations, *watch windows*, etc.)

```
% define and draw Cartesian axis
ang_picture 5 5 55 55
ang_origin 20 20
ang_drawsystem

% define a conic
ang_conic h 0 0 1 -1 0 -3

% construct a point P on the conic
% and the tangent in P
ang_point A1 2 2
ang_point A2 3 2
line l A1 A2
ang_intersec2 P P2 h l
ang_tangent p P h

% draw the conic and the tangent
cmark_t P
ang_drawline p
ang_drawconic h
```

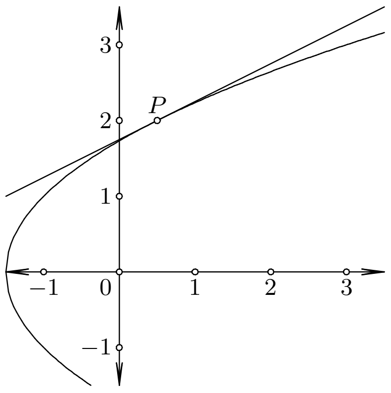


Fig. 2. Illustration for Cartesian commands

```
ang_picture 2 2 58 58
ang_origin 25 25
ang_unit 7
ang_drawsystem_a

ang_draw_parametric_curve x
       {-3; x<4; x+0.05}
       { x; sin(pow(x,2))*cos(x) }

% polar coordinates
number rho 2
ang_draw_parametric_curve phi
       { 0 ; phi<6; phi+0.1}
       { phi*rho*sin(phi)/5 ;
rho*cos(phi) }
```

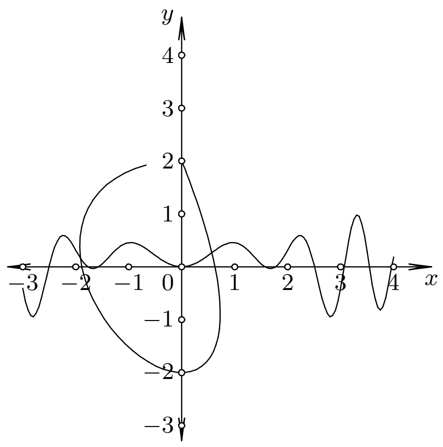


Fig. 3. Illustration for parametric curves

```
point A 5 5
point B 50 5

cmark_b A
cmark_b B

drawsegment A B
translate L_old B B B

number phi 0
while { phi<=70 }
{
       rotate B' A phi B
       line a B' A
       foot L B a

       drawsegment L L_old
       translate L_old L L L

       expression phi { phi+1 }
}

cmark_lt B'
drawdashsegment A B'
drawdashsegment B L
```

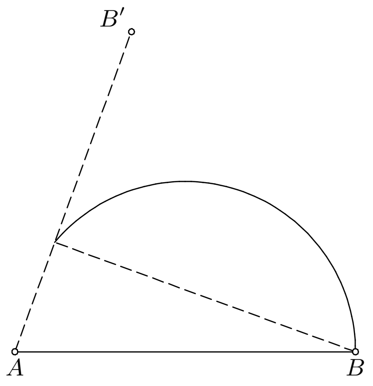
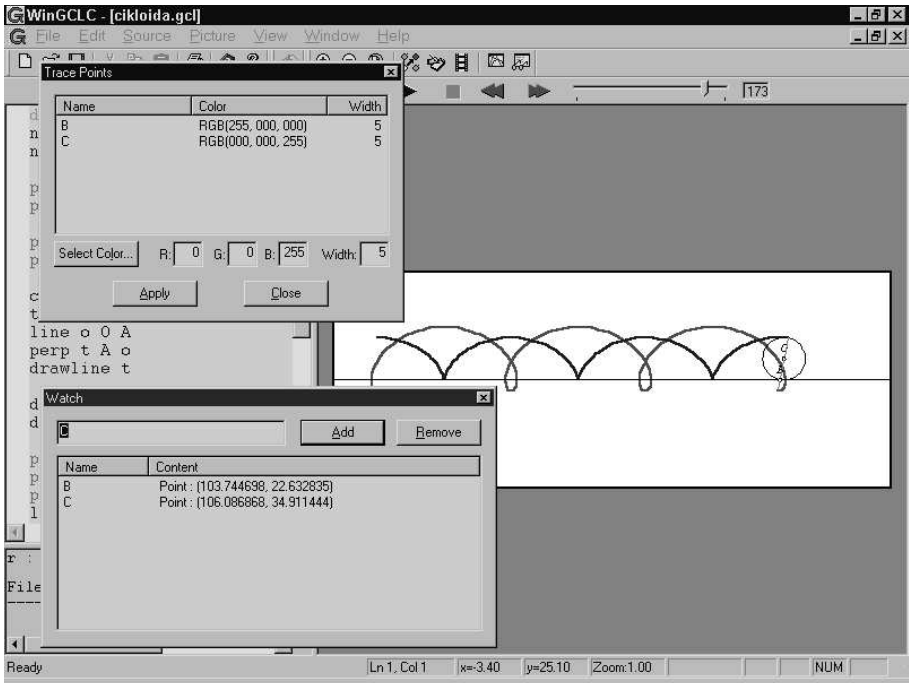


Fig. 4. Illustration for while-loops

Theorem prover. For the example shown in Figure 1, it can be checked that for any particular three points `A`, `B`, and `C`, the points `O_1` and `O_2` (pairwise intersections of the side bisectors) are identical. Using the prover, one can ensure that this is valid statement, i.e., the distance between points `O_1` and `O_2` is always equal to zero. This statement can be given to the prover by simply adding the following line:

```
prove { equal { pythagoras_difference3 O_1 O_2 O_1 } 0 }
```

to the code given in Figure 1. The conjecture is stated within the command `prove` and via the geometric quantity *Pythagoras difference* (P_3). By definition, $P_3(A, B, C) = AB^2 + CB^2 - AC^2$, hence, the value P_3(`O_1`,`O_2`,`O_1`) is equal to 0 if and only if the points `O_1` and `O_2` are identical (for more details, see [23]). The proof is exported to a LaTeX file, with explanations for each proof step. Figure 6 shows last steps of the proof made by the prover (the proof consists of 119 steps and it took 0.035 seconds of CPU time). This example illustrates how GCLC provides geometrical contents directly linked to visual information and supported by machine–generated proofs.



Name	Color	Width
B	RGB(255, 000, 000)	5
C	RGB(000, 000, 255)	5

R: 0 G: 0 B: 255 Width: 5

```
line o O A
perp t A o
drawline t
```

Name	Content
B	Point : (103.744698, 22.632835)
C	Point : (106.086868, 34.911444)

Ready Ln 1, Col 1 x=-3.40 y=25.10 Zoom:1.00 NUM

Fig. 5. *Trace* and *watch* windows with cycloid described in WinGCLC

(113) $(0.062500 \cdot (P_{CBC} \cdot S_{BAC})) = \left(\frac{1}{4} \cdot \left(P_{CBM_a^0} \cdot S_{BAM_a^0}\right)\right)$, by algebraic simplifications

(114) $(0.062500 \cdot (P_{CBC} \cdot S_{BAC})) = \left(\frac{1}{4} \cdot \left(\left(P_{CBB} + \left(\frac{1}{2} \cdot (P_{CBC} + (-1 \cdot P_{CBB}))\right)\right) \cdot S_{BAM_a^0}\right)\right)$, by Lemma 29 (point M_a^0 eliminated)

(115) $(0.062500 \cdot (P_{CBC} \cdot S_{BAC})) = \left(\frac{1}{4} \cdot \left(\left(0 + \left(\frac{1}{2} \cdot (P_{CBC} + (-1 \cdot 0))\right)\right) \cdot S_{BAM_a^0}\right)\right)$, by geometric simplifications

(116) $(0.062500 \cdot S_{BAC}) = \left(\frac{1}{8} \cdot S_{BAM_a^0}\right)$, by algebraic simplifications

28

(117) $(0.062500 \cdot S_{BAC}) = \left(\frac{1}{8} \cdot \left(S_{BAB} + \left(\frac{1}{2} \cdot (S_{BAC} + (-1 \cdot S_{BAB}))\right)\right)\right)$, by Lemma 29 (point M_a^0 eliminated)

(118) $(0.062500 \cdot S_{BAC}) = \left(\frac{1}{8} \cdot \left(0 + \left(\frac{1}{2} \cdot (S_{BAC} + (-1 \cdot 0))\right)\right)\right)$, by geometric simplifications

(119) $0 = 0$, by algebraic simplifications

Fig. 6. Last steps of the proof of the Circumcircle theorem

7 Applications

In this section we briefly discuss three main fields of application for GCLC: in producing mathematical illustrations, for storing mathematical contents, and in teaching mathematics.

7.1 Producing Digital Illustrations

GCLC can serve as a tool for making digital illustrations of high quality. Descriptions made in GC language can be (internally) visualized or can be converted into some other format — LaTeX or bitmap format. Figures in LaTeX format produced by GCLC can be included directly in LaTeX documents, hence they use LaTeX fonts and formulae which is essential for good looking figures in LaTeX documents (while this is a problem for many other formats and tools). Pictures in bitmap format are suitable for different conversions and processing. Although these two picture formats (LaTeX and bitmap) have their advantages, GCLC figures are normally stored in their original, source form. This form is not only precise and sufficient for producing pictures, but also very concise: for instance, all figures from a university book with 120 illustrated geometrical problems [12] have together (in uncompressed, GCLC form) less than 130Kb. GCLC has been used for producing illustrations for a number of other books and articles.

7.2 Storing Mathematical Contents

We advocate for *describing* (rather than *drawing*) mathematical illustrations. Descriptions of images should be given in formal, but human-readable, easily understandable language, close to the intended mathematical meaning. Mathematical illustrations should be stored in such form (rather than in the form of images).

A lot of mathematical contents, both in education and in research, is of visual nature. In many (or most) lecture notes, books, and research articles there are mathematical illustrations. They carry mathematical information, some mathematical message that is represented visually rather than in textual or numerical form. Usually, such message is better understandable to a reader when represented visually. On the other hand, this visual information is usually not mathematically rigorous; it is usually approximation and/or interpretation of some mathematical objects, notions, concepts, numerical data, proofs, ideas etc. A reader interprets the visual information and in his/her mind creates a formal mathematical information. It is often assumed that the reader (with a support from the given textual explanations, earlier experience with illustrations, standard mathematical background, intuition, etc) can understand, "read" the right mathematical message from the illustration. Although a mathematical message is carried by an illustration, that message cannot always be reproduced from the illustration itself. In addition, a mathematician, the author or a reader of a mathematical text, may need to alter an image, to modify some of its characteristics (not only characteristics such as dimensions, but rather some characteristics implying the mathematical contents), to make it more general or more specific, and also to store it in a way that enables these sorts of transformations.

Let us consider geometrical illustrations: a complex geometrical construction may be illustrated by an image and can indeed make the understanding of the text easier. However, without a given context, without provided textual explanations of the problem, it is unlikely that one could guess the right nature and description of the construction. In addition, the Cartesian interpretation of Euclidean geometry is just one of possible interpretations and, hence, potentially misses some of the abstract geometrical meaning. The main point is: an image itself does not provide precise geometrical message. It is better to have a figure description that is formal and can be used for producing the required image, required mathematical illustration. Such information should be stored instead of images. Mathematical content stored in this way (via formal descriptions) is easy to understand, maintain, modify and process in different ways.

The program GCLC is developed along the lines of the given motivation. It is based on using the GC language, in which one can describe a number of geometrical constructions, but also other mathematical objects. Figure descriptions are declarative, precise and brief descriptions of mathematical content and from them corresponding illustrations can be generated. This way, GCLC can be seen as a tool for storing mathematical contents of visual natura in textual form.

7.3 Using GCLC in Teaching Mathematics

In teaching and studying geometry, students can interactively use GCLC to make different attempts in making constructions and/or exploring some mathematical (especially geometrical) objects, notions, ideas, problems, proofs, properties etc [7]. Formally describing mathematical objects is similar to programming, so this helps computer science students to better understand geometry notions and mathematics students to get familiar with programming. Interactive work makes this sort of studying more interesting and more fruitful. The built-in theorem prover can help students link semantic and deductive aspects of geometry.

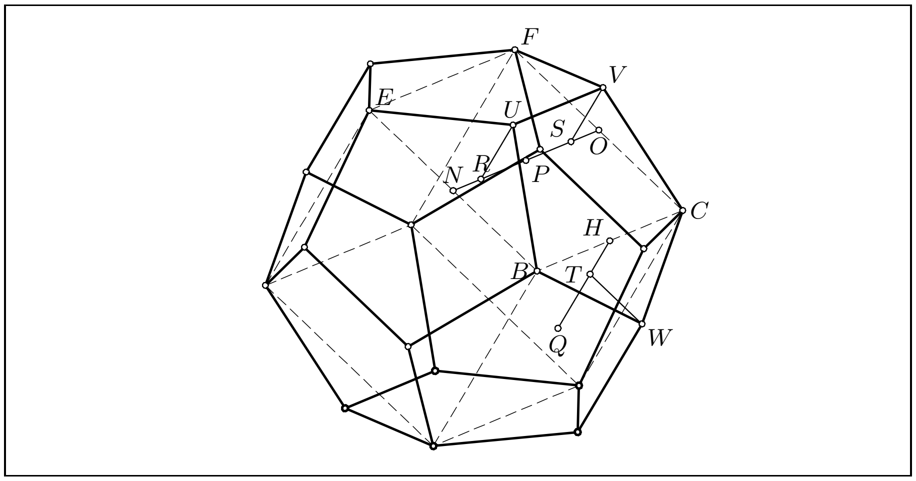


Fig. 7. Illustration for Euclid's construction of dodecahedron

Producing mathematical images and teaching/studying mathematics are not far from each other. For instance, within the geometry courses at the Faculty of Mathematics, University of Belgrade, prof. Zoran Lučić with his students made an electronic version of Euclid's masterpiece of the classical mathematics — *The Elements* [17]. All figures in the book are described in the GC language and directly reflect the accompanying geometrical text. This is probably the first edition of *The Elements* that includes formal, rigorous description of all images, descriptions that directly reflect the accompanying mathematical text. Figure 7 shows Lučić's detailed illustration for Euclid's construction of dodecahedron. This figure is made by using the means of descriptive geometry and shows that three-dimensional objects can be also formally describe in GC language, despite the fact that it is basically designed for plane geometry.

8 Technical Issues, Versions and Availability

GCLC/WINGCLC programs are implemented in C++ programming language. The basic, command line version has around 18000 lines of code. GCLC programs are very small in size: the command line version of GCLC has around 350Kb, WINGCLC has around 700Kb.

There are command-line versions of GCLC for Windows and for Linux. WINGCLC is the version of GCLC for Windows, with a graphical user-friendly interface. As yet, there is no version with a graphical user interface for Linux.

GCLC package (with a manual file and sample files) is freely available from `www.matf.bg.ac.yu/~janicic/gclc/` and from EMIS (The European Mathematical Information Service) servers (`www.emis.de/misc/index.html`).

Figures in LaTeX format generated by GCLC are included in a LaTeX documents by the `\input` command. They use a simple package `gclc.sty`, with definitions that can be changed (so, for instance, can use some particular LaTeX drawing package).

9 Related Work

GCLC/WINGCLC is related to a family of similar, dynamic geometry tools such as *Cinderella*, [25, 6], *Geometer's Sketchpad*, [10, 8] *Eukleides* [20], *Cabri*, [2, 15], *JavaView* [22, 21]. GCLC share a number of features with these tools, but also have some specific features. The main features in which GCLC/WINGCLC differs from similar tools are:

- the deduction module (that directly links visual and semantical geometrical information with deductive properties and machine–generated proofs);
- features that go beyond Euclidean geometry; for instance, GCLC can be used for easily producing figures in Cartesian plane, including graphs of functions (see Figure 8); therefore, GCLC can substitute a wide range of tools for producing mathematical (not only geometrical) illustrations.

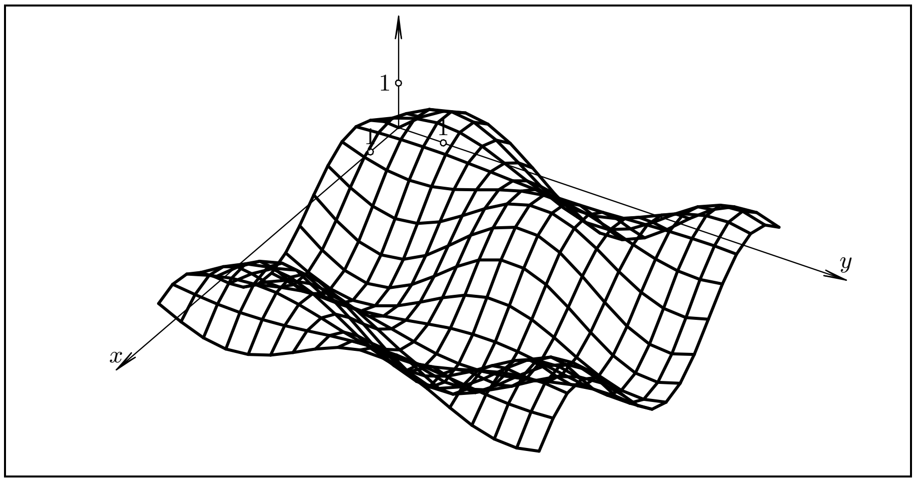


Fig. 8. Function of arity two drawn by GCLC

Some of the advantages of GCLC/WINGCLC (comparing to other tools) are also its free availability, its simplicity, small size of the program, its output files natively supported by LaTeX, interactive features such as animations and traces, etc. In contrast to some dynamic geometry tools, GCLC/WINGCLC focuses on explicitly and formally describing figures (instead of drawing figures) and thus, focusing on *meaning* rather than only on *layout* of figures. These explicit descriptions are easy to write and understand, and they directly reflect the mathematical meaning illustrated by figures.

There are links between GCLC and other tools: there are converters from *JavaView* `.jvx` code and from *Eukleides* code to GCLC code. This brings additional power to GCLC (by making available models created in other tools).

Concerning geometrical theorem proving, there are also some systems related to GCLC/WINGCLC. *Geometry Expert* [9] is a dynamic geometry system with algebraic based theorem prover. There are geometrical theorem provers based on the area method also within the systems Coq [19] and Theorema [1].

10 Further Work

The next version of GCLC/WINGCLC will have support for 3D Cartesian system, for plotting graphs, for exporting figures to EPS and SVG format, and for exporting figures descriptions and proofs to XML format. For future work, we also are planning to:

– implement additional geometrical theorem provers and build them into GCLC; one of the candidates is a prover based on Wu's algorithm [5];
– develop new additional modules to GCLC (for hyperbolical geometry, descriptive geometry, projective geometry etc.), so GCLC could work as a native platform for a range of geometries;

– develop new tools for linking GCLC with other mathematical software tools;
– make XML version of GC format, and link GCLC with some popular markup languages for mathematical contents.

11 Conclusions

In this paper we presented the software package GCLC/WINGCLC for visualizing mathematical objects and notions, for teaching/studying mathematics, and for producing mathematical illustrations of high quality. GCLC uses the GC language for declarative representation of figures, suitable for storing mathematical contents of visual nature. With such representation of information, the intended mathematical message and meaning of mathematical illustrations is possible to preserve and reconstruct.

After first ten years of development, GCLC is much more than a geometrical tool. There is support for symbolic expressions, for drawing parametric curves, for program loops, and WINGCLC makes GCLC an interactive, dynamic mathematical tool with a range of functionalities. The built-in geometry theorem prover can automatically prove a range of complex theorems. It links semantic information about a construction with its deductive properties. It provides mathematical contents directly linked to visual information and supported by machine–generated proofs.

The system is publicly available and is already being used by a number of mathematicians. We are planning to further improve it and extend it.

References

1. Bruno et.al. Buchberger. Theorema: Towards computer-aided mathematical theory exploration. *Journal of Applied Logic*, 2006.
2. Cabri site. http://www.cabri.com.
3. C. C. Chou, OU X. S. Gao, and J. Z. Zhang. Automated production of traditional proofs for constructive geometry theorems. In *Eighth Annual IEEE Symposium on Logic in Computer Science*, 1993.
4. S.C. Chou, X.S. Gao, and J.Z. Zhang. *Machine Proofs in Geometry.* World Scientific, Singapore, 1994.
5. Shang-Ching Chou. *Mechanical Geometry Theorem Proving.* D.Reidel Publishing Company, Dordrecht, 1988.
6. Cinderella site. http://www.cinderella.de.
7. Mirjana Djorić and Predrag Janičić. Constructions, instructions, interactions . *Teaching Mathematics and its Applications*, 23(2):69–88, 2004.
8. Geometer's Sketchpad site. http://www.keypress.com/sketchpad/.
9. GEX site. http://woody.cs.wichita.edu/gex/7-10/gex.html.
10. Nicholas Jackiw. *The Geometer's Sketchpad v4.0.* Emeryville: Key Curriculum Press, 2001.
11. Predrag Janičić and Pedro Quaresma. System description: Gclcprover + geothms. *International Joint Conference on Automated Reasoning (IJCAR-2006).* LNAI 4130. Springer, 2006.

12. Predrag Janičić. *Zbirka zadataka iz geometrije*. Skripta Internacional, Beograd, 1st edition 1997, 6th edition 2005. Collection of problems in geometry (in Serbian).
13. Predrag Janičić and Ivan Trajković. WinGCLC — a Workbench for Formally Describing Figures. SCCG 2003, ACM Press.
14. Donald Knuth. *TeXBook*. Addison Wesley Professional, 1986.
15. J.-M. Laborde and R. Strasser. Cabri-géométre: A Microworld of Geometry for Guided Discovery Learning. *Zentrablatt Für Didactic der Matematik*, 22(5), 1990.
16. Leslie Lamport. *LaTeX: A Document Preparation System*. Addison Wesley Professional, 1994.
17. Zoran Lučić et. al. Euclid's *Elements*. http://www.matf.bg.ac.yu/nastavno/zlucic/.
18. Noboru Matsuda and Kurt Vanlehn. Gramy: A geometry theorem prover capable of construction. *Journal of Automated Reasoning*, 32:3–33, 2004.
19. Julien Narboux. A decision procedure for geometry in coq. In *TPHOLS 2004*, volume 3223 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.
20. Christian Obrecht. Eukleides. http://www.eukleides.org/.
21. Konrad Polthier et. al. JavaView. on-line at: http://http://javaview.de/.
22. K. Polthier, S. Khadem, E. Preuss, and U. Reitebuch. Publication of interactive visualizations with JavaView. *Multimedia Tools for Communicating Mathematics*. Springer, 2002.
23. P. Quaresma and P. Janičić. Framework for the Constructive Geometry. TR2006/001, Center for Informatics and Systems, University of Coimbra, 2006.
24. Pedro Quaresma and Predrag Janičić. Integrating dynamic geometry software, deduction systems, and theorem repositories. *Mathematical Knowledge Management (MKM-2006)*, LNAI 4108. Springer-Verlag, 2006.
25. Jürgen Richter-Gebert and Ulrich Kortenkamp. *Cinderella - The interactive geometry software*. Springer, 1999.

A Additional Examples

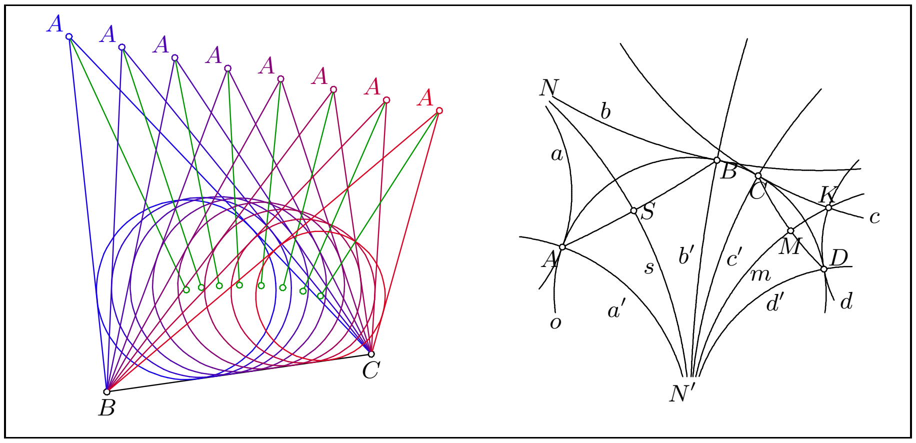


Fig. 9. Inscribed circle for different choices for the point A (left) and hyperbolical geometry figure represented in Poincare's disc model (right)

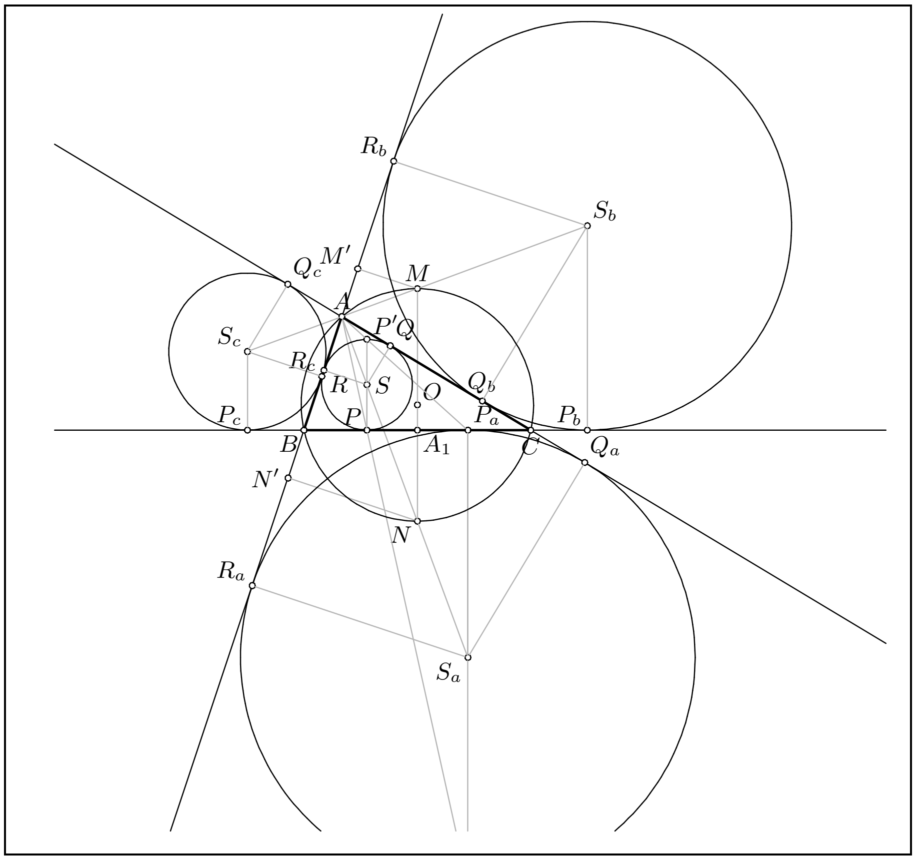


Fig. 10. Triangle with some characteristic points

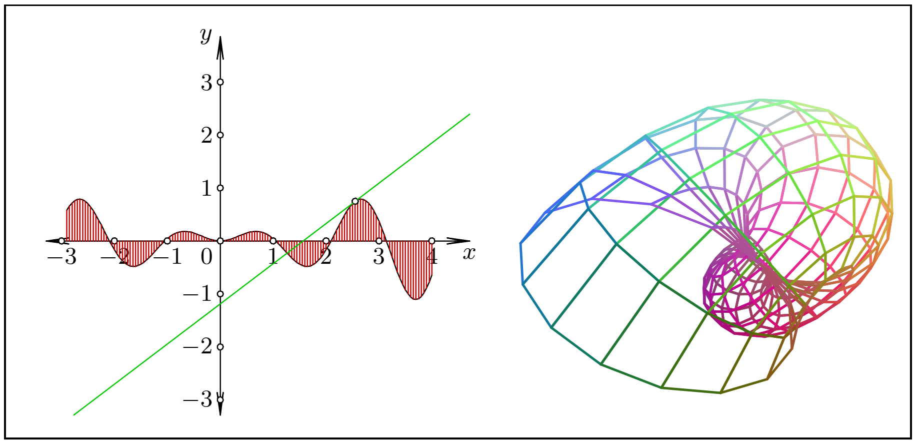


Fig. 11. Function and its derivation (left) and model imported from JavaView (right)