L.E.G.O. − An interactive graphics system for teaching geometry and computer graphics

Norma Fuller and Przemyslaw Prusinkiewicz

Abstract

L.E.G.O. is an interactive graphics system for creating, viewing and manipulating two−dimensional geometric figures and three−dimensional objects. The fundamental operations of the L.E.G.O. language form an electronic metaphor of geometric constructions with a straightedge and compass. This is consistent with the primary application of L.E.G.O., i.e. computer−assisted instruction of geometry. L.E.G.O. is also useful when teaching or studying other areas difficult to grasp without good visual aids, such as mechanics and computer graphics. The system can be used both as an interactive environment for experimenting with geometric constructions and as a tool for preparing illustrations.

Reference

N. Fuller and P. Prusinkiewicz: L.E.G.O. − An interactive graphics system for teaching geometry and computer graphics. *Proceedings of CIPS 1986*.

L.E.G.O. – AN INTERACTIVE GRAPHICS SYSTEM FOR TEACHING GEOMETRY AND COMPUTER GRAPHICS

Norma Fuller and *Przemyslaw Prusinkiewicz*

Department of Computer Science
University of Regina
Regina, Saskatchewan, S4S 0A2 CANADA

ABSTRACT

L.E.G.O. is an interactive graphics system for creating, viewing and manipulating two-dimensional geometric figures and three-dimensional objects. The fundamental operations of the L.E.G.O. language form an electronic metaphor of geometric constructions with a straightedge and compass. This is consistent with the primary application of L.E.G.O., i.e. computer-assisted instruction of geometry. L.E.G.O. is also useful when teaching or studying other areas difficult to grasp without good visual aids, such as mechanics and computer graphics. The system can be used both as an interactive environment for experimenting with geometric constructions and as a tool for preparing illustrations.

Keywords: Interactive graphics systems, geometric constructions, constraint-based systems, computer-assisted instruction.

1. INTRODUCTION

In the classroom, simple two-dimensional illustrations can usually be sketched with sufficient precision for student understanding. However, it is difficult for even the most talented instructor to sketch three-dimensional objects and complex two-dimensional figures in real time, using a chalkboard or transparencies, with enough precision to enhance the learning process. The students' understanding must therefore evolve totally from abstract symbolism without an adequate visual model.

This paper describes a system called L.E.G.O. (LISP-based Euclidean Geometry Operations). The fundamental concept of L.E.G.O. is to provide an electronic metaphor for a straightedge and compass. Consequently, L.E.G.O. is particularly suitable for computer-assisted instruction of Euclidean geometry.

The educational applications of L.E.G.O. fall roughly into two categories:

- **The computer as a blackboard.** L.E.G.O. is used by the instructor to illustrate geometric objects and constructions. Such illustrations are more precise and visually more attractive than those drafted on a traditional blackboard.
- **The computer as a virtual laboratory.** The students interact with the system. They create two-dimensional figures and three-dimensional objects using Euclidean constructions, look at these objects from different angles, and introduce modifications. Manipulations reveal general properties of the constructions and provide empirical material for transforming observations into hypotheses.

Apart from the computer-assisted instruction of geometry, constructions can be applied to illustrate selected areas of mechanics (in particular, the theory of linkages) and computer graphics (e.g. 3D modeling and projections).

L.E.G.O. was originally conceived as an interactive system [9]. However, it also can be used to prepare illustrations (plots, slides and prints) suitable for publication purposes. In this case, the real-time interaction is sacrificed for the sake of good quality of rendering.

Technically, L.E.G.O. is characterized by the following features:

- Geometric figures can be referred to by names and used as arguments or obtained as results of functions.
- Functions are defined interactively, by examples. Before a geometric construction is started, selected figures (points, lines, etc.) can be specified as arguments. When the construction is finished, it can be recalled using a different set of arguments.
- Function calls can be nested, allowing the user to easily define recursive figures and objects.
- Three-dimensional objects can be defined, manipulated and viewed.

The idea of using geometric constructions as a basis for an interactive computer graphics system has received almost no attention in the past. This is rather surprising, given the fundamental role of constructions in Euclidean geometry. Only recently has another construction-based system been reported in the literature [2]. On the other hand, L.E.G.O. shares some features and applications with constraint-based graphics systems [3,12,14,15,17].

2. THE L.E.G.O. LANGUAGE

The L.E.G.O. language [8] is a graphical extension of Franz LISP [6,18] and it preserves the LISP syntax. L.E.G.O. and LISP functions can be interleaved in the same program. However, L.E.G.O. maintains its own symbol table and therefore cannot be considered simply as a library of LISP functions. This symbol table contains references to the primitive graphical objects: points, lines, circles, planes and spheres. Associated with these primitives is a set of predefined functions which make it possible to define new objects in terms of the objects already specified. The following functions are essential for developing two-dimensional constructions:

(point *x y new_name*)

Creates a point given coordinates *x* and *y*, and calls it *new_name*. (The term "create" means to produce a new graphic primitive by recording its features in the L.E.G.O. symbol table and by drawing it on the screen.)

(line *point1 point2 new_name*)

Creates a line from a previously defined *point*1 to a previously defined *point*2, and calls it *new_name*.

(circle *center radius new_name*)

Creates a circle given a previously defined point *center*, with the radius equal to a previously defined line *radius*. The circle is called *new_name*.

(intersection *primitive1 primitive2 new_name1* [*new_name2*])

Creates the points of intersection between two-dimensional primitives: points, lines and circles. Intersections with a point can be used to check whether it coincides with another point, or whether it lies on a line or a circle. The actual number of intersections is returned as the value of the function. The value of −1 is returned when intersecting two identical lines or circles.

The operation of intersection requires particular attention. It may create two intersection points and the user must know which point of intersection will be called *new_name*1, and which one – *new_name*2. In L.E.G.O. the points of intersection are distinguished on the basis of the oriented angles between the intersecting primitives. Consequently, the correct selection of the points of intersection is preserved when translating or rotating the construction.

In order to illustrate key features of the L.E.G.O. language, let us consider some simple programs. They can be developed noninteractively (using a text editor) or interactively. In the latter case, each statement entered to the system is immediately executed to provide visual feedback. The first program creates line *L* defined by points *A* and *B*, and bisects *L* with line *P* perpendicular to *L*.

Program 1.

```
1  (point 400 370 A)
2  (point 600 470 B)
3  (line A B L)
4  (circle A L C1)
5  (circle B L C2)
6  (intersection C1 C2 X1 X2)
7  (line X1 X2 P)
```

The construction described by this program is shown in Fig. 1.

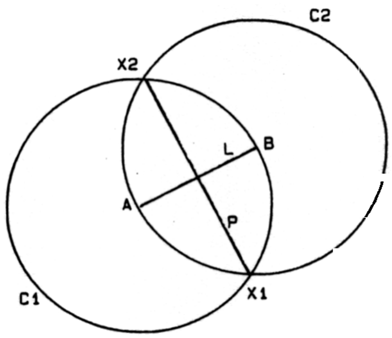


Fig. 1. Bisecting a line in L.E.G.O.

A geometric construction can be specified as a function using function definition functions **define_function** and **end_function**. For example, in order to specify the construction to bisect a line as a function, the statements:

```
(define_function bisect (A B) (P))
...
(end_function)
```

should have been typed after lines 2 and 7 of Program 1, respectively. The statements in lines 3-7 would then constitute the function body. Parameters of the **define_function** function indicate that the new function **bisect** shall be called with two arguments referring to previously defined primitives (*A* and *B*), and will create a new primitive *P* as a result. Note that line *L* will become local to the function **bisect** and therefore should not be referred to outside the body of this function. The function **bisect** can be used, for example, to construct the circumcircle of a given triangle *ABC* (Program 2 and Fig. 2).

Program 2.

```
1  (bisect A C P)
2  (bisect  B C Q)
3  (intersection P Q X)
4  (line X C R)
5  (circle X R Z)
```

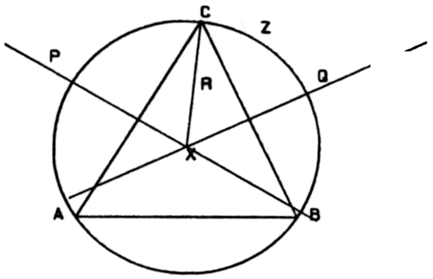


Fig. 2. Construction of the circumcircle of a triangle.

L.E.G.O. functions can be called recursively. Assume that the user-defined function (**midtriangle** *A B C D E F*) creates a triangle given vertices *A,B,C*, and returns the mid-points of the edges: *D,E,F*. Using **midtriangle**, the Sierpiński gasket [13] (Fig. 3) can be defined as follows:

Program 3.

```
1  (point 220 150 A)
2  (point 790 150 B)
3  (point 505 643 C)
4  (define_function gasket (A B C))
5  (midtriangle A B C D E F)
6  (write_function)
7  (if (> (distance A B) 40) then
8        (gasket A D F)
9        (gasket B E D)
10       (gasket C F E))
11 (end_function)
```

Line 6 calls function **write_function** which temporarily writes the function currently being defined. This is a necessary statement before this function can call itself. Line 7 illustrates the mixing of LISP and L.E.G.O. functions. A predefined L.E.G.O. function **distance** is used in conjunction with the LISP macro **if...then** to control the termination of the recursive calls.

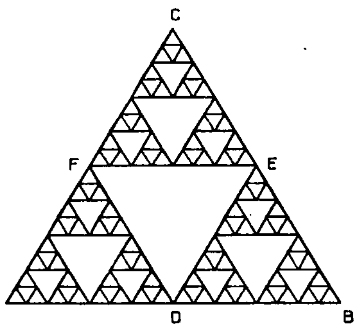


Fig. 3. The Sierpiński gasket.

In order to develop three-dimensional constructions, the functions **point**, **line** and **intersection** described before are extended to operate on three-dimensional primitives. Additionally, the following functions are defined:

(**ppp_plane** *point1 point2 point3 new_name*)

(**pl_plane** *point line new_name*)

(**ll_plane** *line1 line2 new_name*)

Each of these functions enters plane *new_name* to the L.E.G.O. symbol table. The plane is specified by three non-collinear points, a line and a point not on the line, or two intersecting or parallel lines, respectively. The plane is not displayed. (In order to present the plane visually, the user must draw on it an appropriate two-dimensional figure, for example a rectangle.)

(**circle** *center radius plane new_name*)

Creates a circle on a previously defined *plane*, given the *center* and the *radius* of the circle. The circle will be called *new_name*.

(**sphere** *center radius new_name*)

Creates a sphere given a previously defined point *center*, with the radius equal to the length of a previously defined line *radius*. The sphere is called *new name*.

An example of a three-dimensional construction is given by Program 4. It creates a wire-frame model of a regular tetrahedron, given an equilateral triangle *ABC* (Fig. 4).

Program 4.

```
1  (line A B R)
2  (sphere A R Sphere1)
3  (sphere B R Sphere2)
4  (sphere C R Sphere3)
5  (intersection Sphere1 Sphere2 Circle)
6  (intersection Sphere3 Circle D)
7  (line A D L4)
8  (line B D L5)
9  (line C D L3)
```

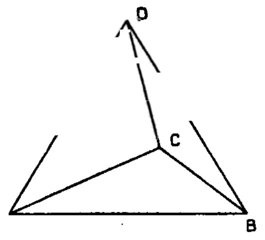


Fig. 4. A regular tetrahedron. (The spheres used for construction are not shown.)

While programs 1 - 4 illustrate the essential features of the L.E.G.O. language, they use but a small fraction of the available functions. In total, L.E.G.O. has approximately 100 predefined functions [8], which can be grouped into nine classes.

1. **Object definition functions** are used to create L.E.G.O. graphics primitives: points, lines, circles, planes and spheres. Functions **point, line, sphere, intersection,** etc. belong to this category. The object definition functions are the fundamental tools for modeling geometric objects in L.E.G.O.

2. **Query functions** provide information about graphical primitives. Two subclasses can be distinguished:
   - *Functions which return a numerical value* (e.g. coordinate of a point, distance between points, length of a line). They are used primarily in conditional statements.
   - *Functions which return a graphic primitive* (e.g. endpoint of a line, center of a sphere, plane con-

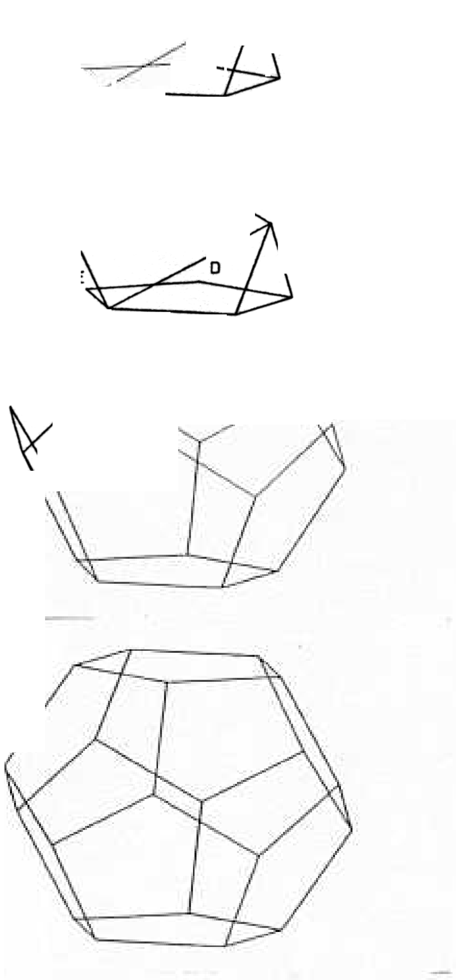


Fig. 6. Construction of a dodecahedron.

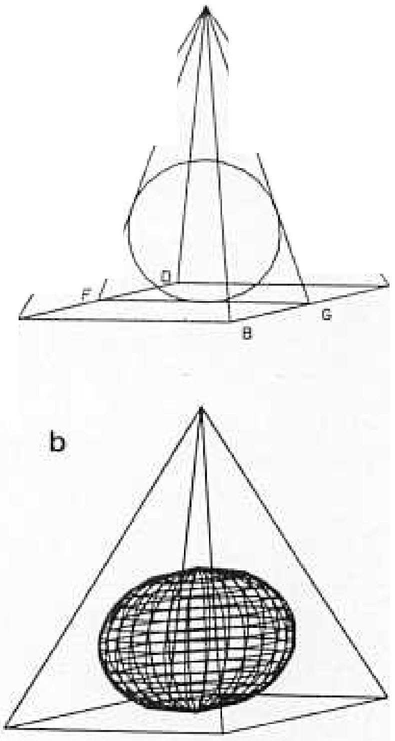


Fig. 7. Construction of the sphere inscribed in a pyrami

pectively, and call the point of intersection *F*. (b) Given ɛ vertices *A*, *B*, *F*, construct pentagon *ABFGH*. (c, d) ntinue constructing pentagons given three vertices until entire dodecahedron is formed.

Examples 1 and 2 indicate the practical importance of function definition mechanism in L.E.G.O. If von audt's construction considered in Example 1 is defined as function, pentagon *ABCDE* in Example 2 can be constructed using a single function call.

Another example of a three-dimensional construction is iven below.

Example 3. Figure 7 shows the construction of a sphere inscribed in a pyramid with a square base. (a) The construction is reduced to that of inscribing a circle in triangle *EFG* formed by the top of the pyramid and the midpoints of two posite edges of the base. (b) The center and the radius of this circle are also the center and the radius of the resulting sphere.

Interactivity is one of the most important features of L.E.G.O. Students can experiment with geometric constructions and study them under a variety of conditions, for example, by providing different sets of data. This helps in the formulation of hypotheses and illustrates theorems.

Example 4. The following three theorems are taken from [4]:

- The internal bisectors of the three angles of a triangle are concurrent and define the center of the incircle of this triangle (Fig. 8).
- (Varignon's theorem) The figure formed when the midpoints of the sides of a quadrangle are joined in order is a parallelogram (Fig. 9).
- (The nine-point circle) The feet of the three altitudes of any triangle, the midpoints of the three sides, and the midpoints of the segments from the three vertices to the orthocenter, all lie on the same circle. The center of this circle lies on the Euler line, midway between the orthocenter and the circumcenter (Fig. 10, provided Brad Longworth

taining a circle). They are useful when arguments other than points are passed to functions.

3. **Drawing functions** are used to display simple figures such as alphanumerical symbols, arcs of circles and filled polygons. These are not considered as L.E.G.O. primitives and, consequently, cannot be passed as arguments to the function **intersection**.
4. **Presentation definition functions** are used to control the appearance of graphical objects on the screen. Examples of controlled features are listed below:
   - *Visibility of primitives.* Auxiliary construction lines may be removed from the final picture.
   - *Display of primitive names.* In some applications, such as the presentation of geometric constructions for educational use, primitives should be labeled. In other cases, such as the modeling of realistic scenes, the display of names should be suppressed.
   - *Color, width and style of lines.*
   - *Color, size and type of fonts.*
5. **Function definition functions** form a class which contains **define_function** and **end_function**, already described.
6. **Viewing functions** are used to divide the screen surface into separate viewports, define parameters of the projection, rotate objects in space, etc.

   Interaction supporting functions make it possible to remove or modify previously defined primitives. These functions are particularly useful when developing constructions interactively, since each statement entered to the system is immediately executed and it cannot be subsequently altered by editing.
8. **System functions** are used for file manipulation (such as function loading), to configure the system for a particular type of graphics output device (such as a plotter), etc.

   Debugging functions provide information about primitives stored in the symbol table, actual viewing parameters, etc.

3. APPLICATIONS OF L.E.G.O.

3.1. Computer assisted instruction of Euclidean geometry.

The fundamental concept of L.E.G.O., mimicry of constructions with straightedge and compass, obviously makes the system suitable for presenting constructions of Euclidean geometry. Due to the accuracy of computer calculations, exact drawings can be easily obtained. This is in contrast to the approximate drawings made at the chalkboard and sometimes found in publications. Additionally, the progress of a construction in time can be shown. It can be studied directly in front of the monitor, or presented as a sequence of snapshots.

Example 1. Figure 5 illustrates von Staudt's construction of the regular pentagon [1]. (a) Construct a square *ABCD* and connect the midpoints of the opposite edges with lines *EG* and *FH*. Inscribe circle *M* in *ABCD*. (b) Extend line *AB*, and mark on it point *S* so that the distance $|AS|$ equals three times distance $|AE|$. Let line *CS* intersect circle *M* at points *P* and *Q*. (c) Join *P* and *Q* to the point *E*, and let *P′*, *Q′* be the intersections of lines *EP* and *EQ* with line *FH*. (d) Then the vertices of a regular pentagon are given by point *F* and the intersections with the circle of the lines through *P′* and *Q′* perpendicular to *FH*.

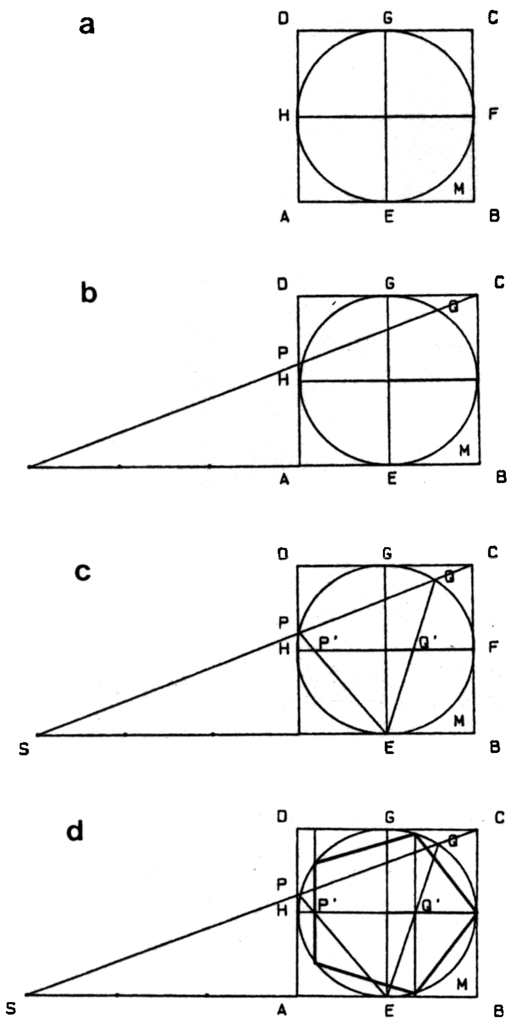


Fig. 5. Von Staudt's construction of a regular pentagon.

L.E.G.O. is particularly useful when illustrating three-dimensional objects and constructions.

Example 2. Figure 6 shows a dodecahedron and illustrates its construction. (a) Let *r* and *R* denote an edge and a diagonal of the regular pentagon *ABCDE*. Intersect three spheres with centers at points *A*, *B*, *C*, and radiuses *R*, *r* and *R*,

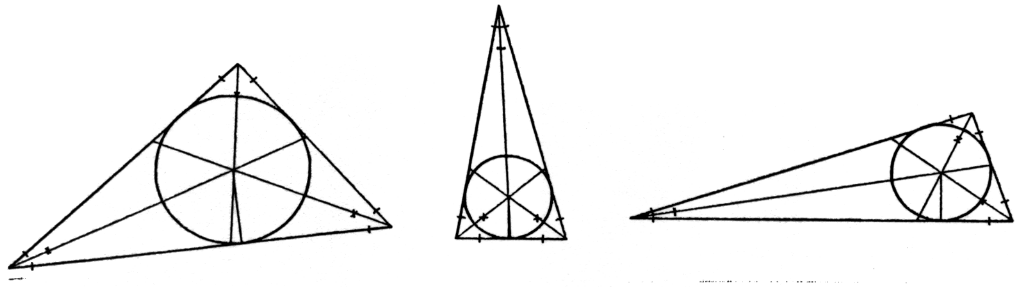


Fig. 8. The internal bisectors of the three angles of a triangle are concurrent and define the center of the incircle of this triangle.

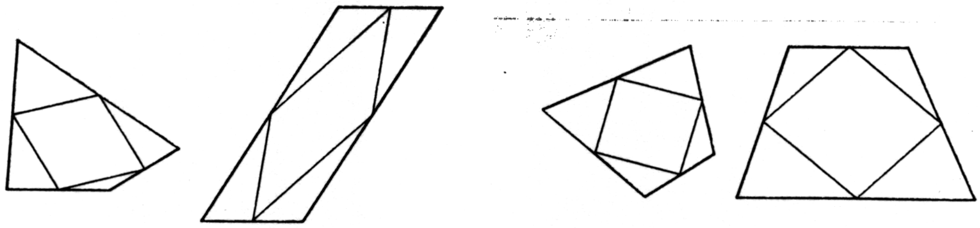


Fig. 9. Illustration of Varignon's theorem.

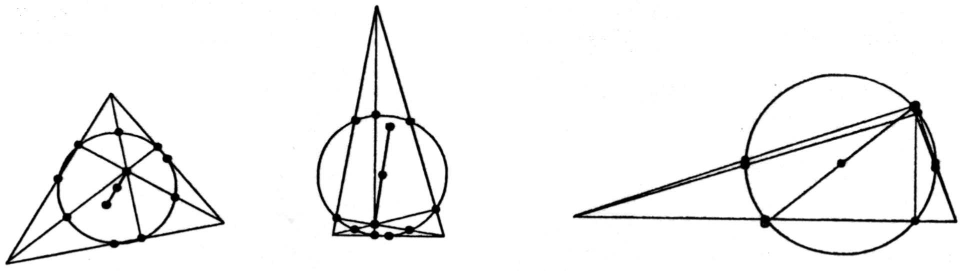


Fig. 10. The nine-point circle.

The notion of a geometric locus is instrumental in geometry, yet many loci are difficult to visualize using traditional methods. An approximation of a locus which is neither a straight line nor a circle involves a repetitive construction of its subsequent points which is tedious if a "real" straightedge and compass were used. On the other hand, repetitive geometric constructions can be easily programmed in L.E.G.O.

Example 5. Figure 11 shows an ellipse and a hyperbole :h were constructed using the following definitions:

An ellipse is a locus of points *A* such that the sum of distances of *A* from two fixed points, called foci, is constant.

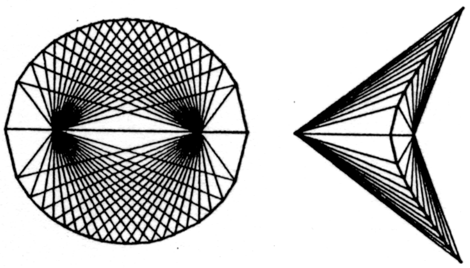


Fig. 11. An ellipse and a hyperbole defined

- A hyperbole is a locus of points A such that the difference of distances of A from two fixed foci is constant.

Another concept involving repetitive constructions is the recursive definition of geometric objects. A simple example of a recursive construction in L.E.G.O. was shown in Fig. 3. Two more complex examples are given below.

Example 6. (provided by Brad Longworth) Figure 12 illustrates the Apollonian gasket [13]. This figure is obtained by recursively constructing the circle tangent to three given circles using the method described in [4].

Example 7. Figure 13 illustrates the Hilbert curve [11] and its three-dimensional extension [16].

Once again, note that constructions shown in Figs. 12 and 13b would be difficult to obtain using a "real" straightedge and compass.

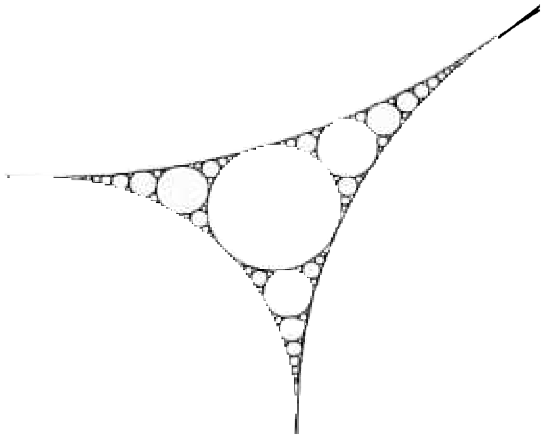


Fig. 12. The Apollonian gasket.

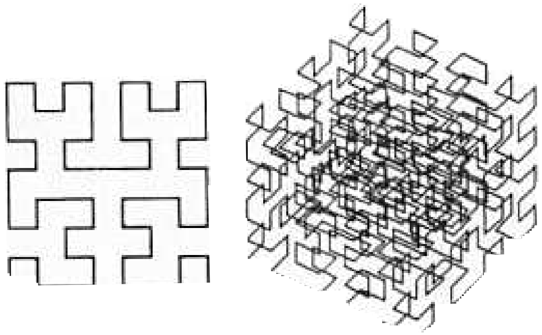


Fig. 13. Two-dimensional and three-dimensional Hilbert curves.

3.2. Applications of L.E.G.O. to computer graphics.

So far, two distinct applications of L.E.G.O. to computer graphics have been identified. The first one is the illustration of projections. Since the projection of a point onto a plane is defined by the intersection of the projector with the projection plane, projections can be easily constructed in L.E.G.O.

Example 8. Figure 14 presents two types of projections: isometric and two-point perspective [7]. Figure 15 illustrates construction of a shadow.

Another application of L.E.G.O. falls into the category of geometric modeling. Repetitive (recursive or iterative) geometric constructions can be used not only to approximate curved lines (Example 5), but also curved surfaces.

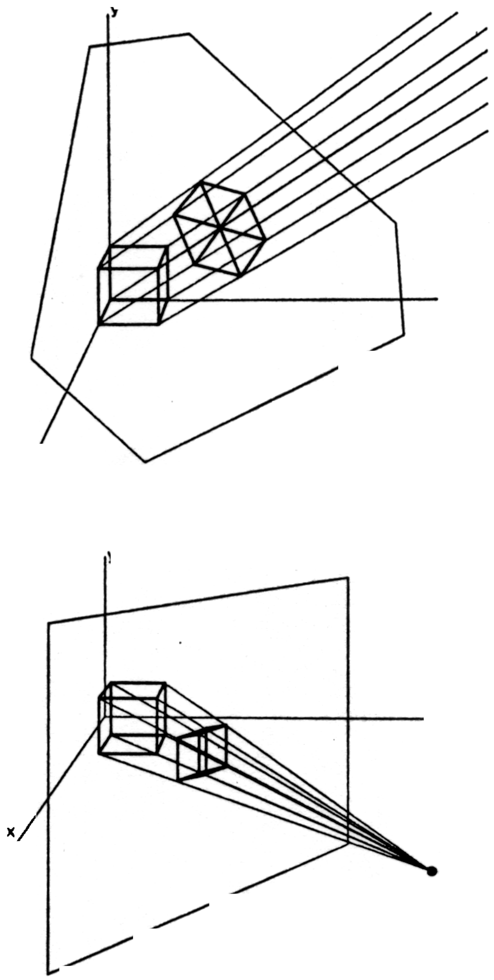


Fig. 14. Two examples of projections.

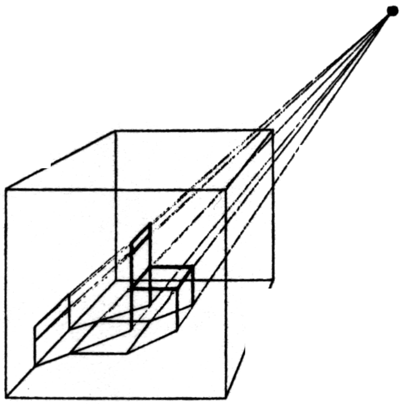


Fig. 15. Construction of a shadow.

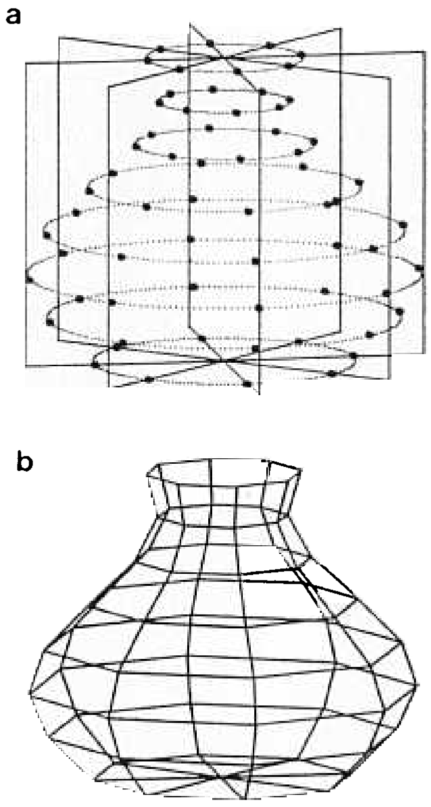


Fig. 16. Construction of a polygon mesh of a vase.

Example 9. Fig. 16a illustrates the L.E.G.O. construction of a polygon mesh of a vase. The vertices of the mesh lie at the intersections of four vertical planes with a sequence of horizontal circles. The final mesh is shown in Fig. 16b.

The modeling of curved surfaces using geometric constructions is interesting not just from the educational point of view. In some cases, the geometric construction of a surface is simpler and more straightforward than other modeling techniques. The concept of geometric modeling using constructions is new and requires a further study.

3.3. Modeling of mechanisms and kinematic analysis.

Mechanisms consist of movable elements (links) connected together in kinematic pairs which put constraints on the motion of the links. The essential problem of kinematic analysis is to determine the relationship between the input and the output motion of a mechanism. This relationship can be very complex and difficult to grasp. Consequently, working models of mechanisms are often necessary to gain a full understanding of the motion [10]. Alternatively, mechanisms can be represented as computer models. The possibility of modeling mechanisms using constraint-based graphics systems was recognized by Sutherland [17] and described as the most interesting application of his Sketchpad. Various types of mechanisms can also be modeled using L.E.G.O. They can be interactively manipulated by the user, or put in motion by a "virtual motor", i.e., a function which moves the input links without user intervention.

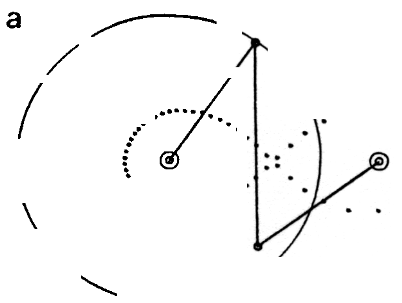
Example 10. The mechanism shown in Fig. 17a is known as James Watt's linkage [5]. If it is put in motion by rotating the left link, the midpoint of the middle link traces a Bernoulli's lemniscate. A "stroboscopic picture" of the linkage (Fig. 18) reveals that the velocity of the midpoint of the middle link varies while the left link rotates at a constant speed. Another mechanism, called Peaucelier's linkage, is shown in Figs. 17b and 19. It is interesting from the historical perspective, as it is the first exact solution to the straight-line motion problem. (This problem consists of converting a circular motion at the input into a linear motion at the output of the linkage [5].)

Example 11. (provided by Wayne Hassman) Figure 20 presents a different mechanism – a simple pulley. The "stroboscopic picture" shows consecutive positions of both loads and reveals the non-linear path of the right load.

4. CONCLUSIONS

L.E.G.O. is an interactive graphics system implementing an electronic metaphor of straightedge and compass. Two-dimensional figures and three-dimensional objects are created using geometric constructions and can be interactively manipulated. L.E.G.O. extends the capabilities of a "real" straightedge and compass in two directions:

- Three-dimensional, iterative and recursive constructions can be performed easily and accurately.
- Once a construction has been defined, it can be manipulated by changing arbitrary arguments.



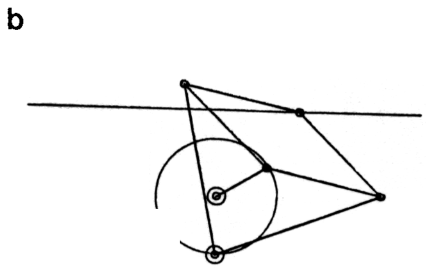


Fig. 17. Examples of linkages: (a) James Watt's linkage, (b) Peaucelier's linkage.

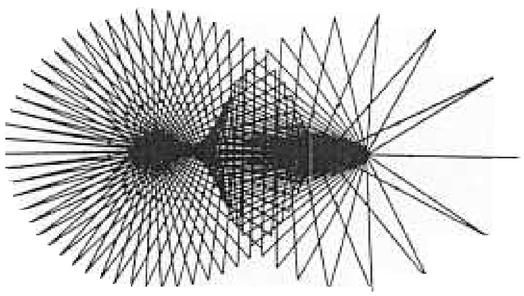


Fig. 18. A stroboscopic view of the James Watt's linkage.

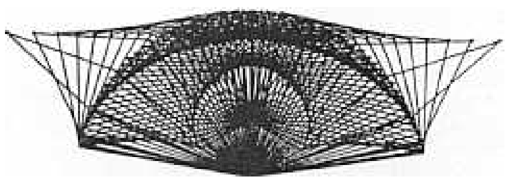


Fig. 19. A stroboscopic view of the Peaucelier's linkage.

L.E.G.O. is particularly suited for computer-assisted instruction of the Euclidean geometry. However, it also can be used in less obvious applications, such as the modeling of curved surfaces and the analysis of mechanisms. The range of practical applications of the construction-based approach require a further study.

Since the beginning of 1985, various versions of L.E.G.O. have been available to computer graphics students at the University of Regina. They found the system very attractive, easy to use, and applicable to many practical problems. Although these opinions were not formally surveyed, they reinforce our conclusion that L.E.G.O. is a viable educational tool with a wide range of applications.

ACKNOWLEDGMENT

This research was supported in part by grant No. A0324 from the Natural Sciences and Engineering Research Council of Canada. This support is gratefully acknowledged.

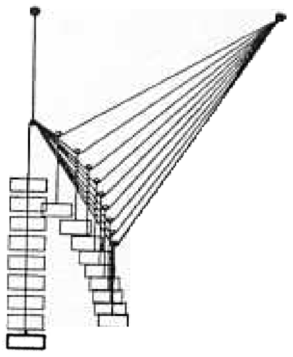


Fig. 20. A stroboscopic view of a pulley.

REFERENCES

[1] Behnke, H., Bachmann, F., Fladt, K. and Kunle, H. (Eds.): *Fundamentals of Mathematics. Vol II: Geometry.* MIT Press, Cambridge, 1983.

[2] Bier, E. A. and Stone, M. C.: Snap-dragging. *Computer Graphics* **20** (4), Aug. 1986, pp. 233-240.

[3] Borning, A.: The programming language aspects of Thinglab, a constraint-oriented simulation laboratory. *ACM Trans. on Programming Languages* **3** (4), Oct. 1981, pp. 353-387.

[4] Coxeter, H. S. M. and Greitzer, S. L.: *Geometry Revisited.* Random House, New York, 1967.

[5] Cundy, H. M. and Rollet, A. P.: *Mathematical Models.* Oxford University Press, London, 1961.

[6] Foderaro, J.: *The Franz LISP Manual.* University of California, Berkeley, 1979.

[7] Foley, J. D. and Van Dam, A.: *Fundamentals of Interactive Computer Graphics.* Addison-Wesley, Reading 1982.

[8] Fuller, N.: *User's Guide to L.E.G.O. - Version 1.0.* Techn. Rep. CS-85-19, Department of Computer Science, University of Regina, 1985.

[9] Fuller, N., Prusinkiewicz, P. and Rambally, G.: L.E.G.O. - An interactive computer graphics system for teaching geometry. *Proceedings, 4th World Conference on Computers In Education.* North-Holland, Amsterdam 1985, pp. 359- 364.

[10] Hain, K.: *Applied Kinematics.* McGraw-Hill, New York, 1967.

[11] Hilbert, D.: Ueber stetige Abbildung einer Linie auf ein Flächenstück. *Math. Annln.* **38**, 1891, pp. 459-460.

[12] Johnson, T. E.: Sketchpad III: A computer program for drawing in three dimensions. In *1963 Spring Joint Computer Conference,* reprinted in Freeman H. (Ed.): *Interactive Computer Graphics,* IEEE Computer Soc. 1980, pp. 20-26.

[13] Mandelbrot, B. B.: *The Fractal Geometry of Nature.* W. H. Freeman, San Francisco, 1982.

[14] Nelson, G.: Juno, a constraint-based graphics system. *Computer Graphics* **19** (33), July 1985, pp. 235-243.

[15] Prusinkiewicz, P. and Streibel, D.: Constraint-based modeling of three-dimensional objects. *Proceedings of Graphics Interface '86 - Vision Interface '86,* 1986, pp. 158-163.

[16] Stevens, R. J., Lehar, A. F. and Preston, P. H.: Manipulation and presentation of multidimensional image data using the Peano scan. *IEEE Trans. Pattern Recognition and Machine Intelligence* **PAMI-5** (5), Sep. 1983, pp. 520-526.

[17] Sutherland, I. E.: Sketchpad: A man-machine graphical communication system. In *1963 Spring Joint Computer Conference,* reprinted in Freeman H. (Ed.): *Interactive Computer Graphics,* IEEE Computer Soc. 1980, pp. 1-19.

[18] Wilensky, R.: *LISPcraft.* W.W. Norton, New York, 1984.

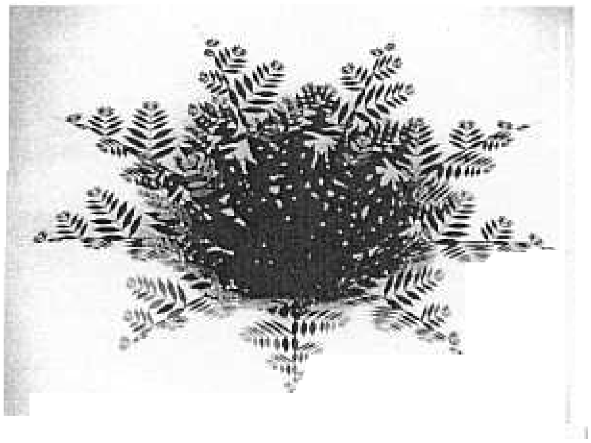


Fig. 7. A plant with small leaves approximated by flat polygons.

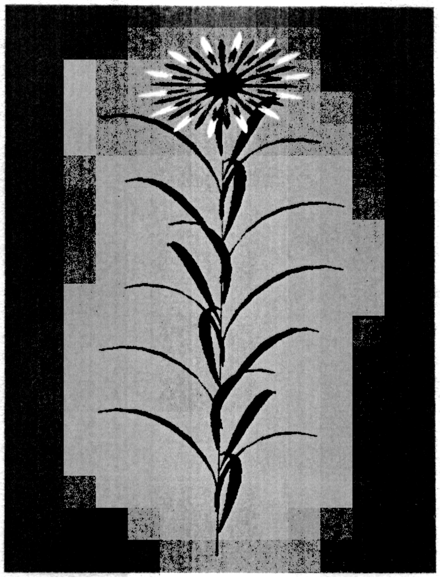


Fig. 8. A plant with curved leaves.

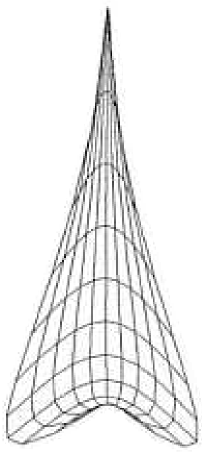


Fig. 9. A Bezier patch modeling a leaf.

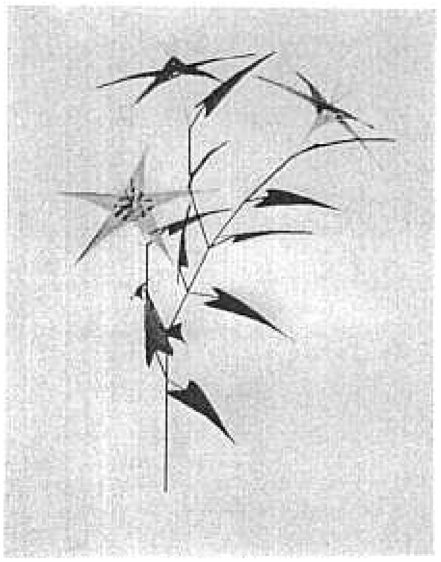
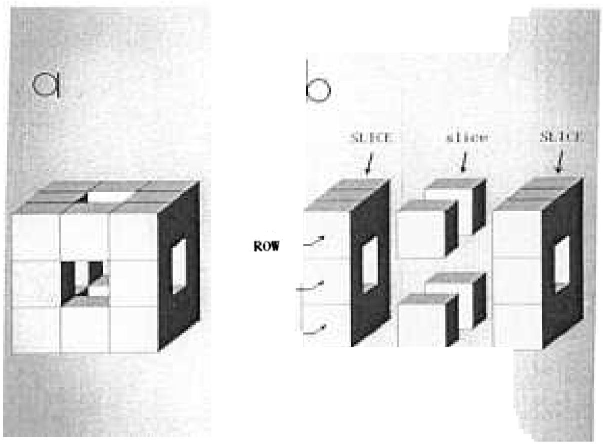


Fig. 10. A plant with leaves and flowers modeled using Bezier patches.



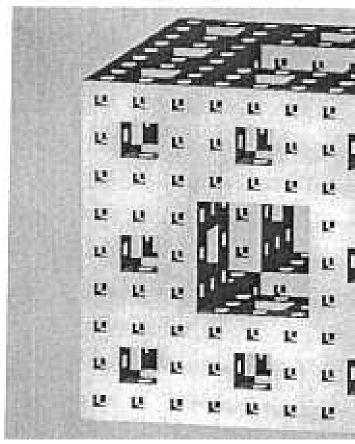
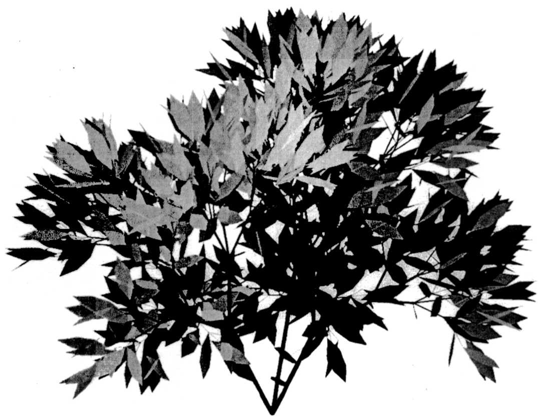
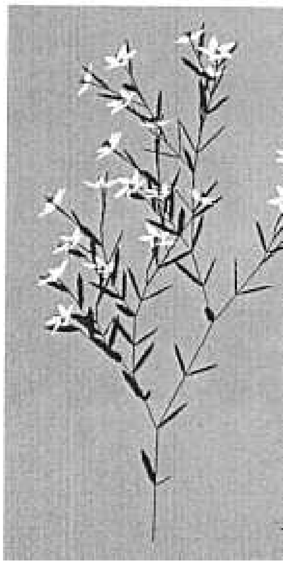


Fig. Construction the Menger

Fig.





.2).

Fig. plant generated