SHORT COMMUNICATION

CAPTURING CONGRUENCE WITH A TURTLE

RINA ZAZKIS[1] AND URI LERON

ABSTRACT. This note compares turtle geometry and Euclidean geometry with respect to their treatment of similarity and difference of plane figures. It is observed that while the Euclidean notion of congruence faithfully captures a common perception of "sameness", the turtle expression of this idea is too weak. To deal with this insufficiency we add a new turtle operation, FLIP, which turns the turtle upside down. This brings the turtle's power to express invariance of shape up to Euclid's.

The problem and its solution are viewed briefly from the perspectives of mathematics, computer science and education. The mathematical discussion compares the turtle group and the Euclidean group. The computational discussion focuses on the issue of "expressive power" of a language and how it may be enhanced. The educational discussion suggests a classroom implementation of the above ideas.

INTRODUCTION

This note compares turtle geometry and Euclidean geometry with respect to their treatment of similarity and difference of plane figures. It is observed that while the Euclidean notion of congruence faithfully captures a common perception of "sameness", the turtle expression of this idea is too weak. To deal with this insufficiency we add a new turtle operation, FLIP, which turns the turtle upside down.[2] This brings the turtle's power to express invariance of shape up to Euclid's.

On a more sophisticated level, our solution is viewed from two complementary perspectives. From the perspective of higher mathematics, the addition of FLIP is viewed as enlarging the pool of available turtle operations. Formally, it is an extension of the *turtle group* (the group of turtle operations) to one that is isomorphic to the Euclidean group (the group of plane isometries). This is related to Klein's *Erlanger Program*, in which various geometries are classified via groups of transformations and their invariants.

From the computer science perspective, this problem reflects an insufficiency of the turtle *language*. The addition of the new command FLIP is seen here as enhancing the *expressive power* of the language, so that it may faithfully capture our notion of invariance of shape. This represents an important trend in modern computer science (Abelson and Sussman, 1985): rather than lowering our descriptions down to the limitations of a given programming language (in effect, using various programming tricks), we strive to enrich the language till its expressive power is up to the given task.

Educational Studies in Mathematics **22**: 285–295, 1991.

Finally, we describe a classroom implementation of the foregoing ideas. It is based on an activity which is simple enough to suit virtually all ages, but at the same time is sophisticated enough to demonstrate the above-mentioned insufficiency of turtle geometry. The proposed enhancement to the turtle language can then lead, depending on the level and interests of the students, to discussions of geometry, transformations, groups, programming languages and issues of style in programming and problem solving.

THE HILBERT CURVE

We begin by considering a turtle description of the well-known Hilbert curve, as given in Abelson and diSessa (1981, pp. 96–98).

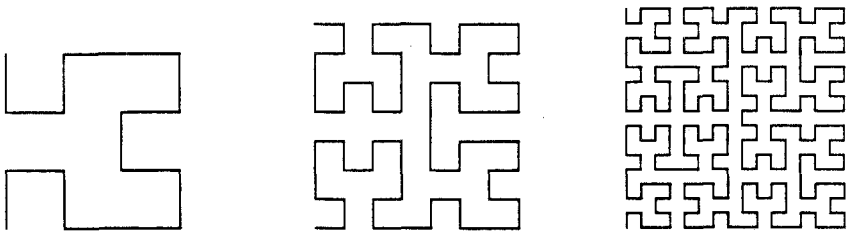

Fig. 1. The Hilbert curve.

Their first version is composed of two "mirror" procedures, LHILBERT and RHILBERT that recursively call themselves and each other.

```
TO LHILBERT :SIZE :LEVEL
  IF :LEVEL = 0 [STOP]
  LT 90
  RHILBERT :SIZE :LEVEL-1
  FD :SIZE
  RT 90
  LHILBERT :SIZE :LEVEL-1
  FD :SIZE
  LHILBERT :SIZE :LEVEL-1
  RT 90
  FD :SIZE
  RHILBERT :SIZE :LEVEL-1
  LT 90
END
```

```
TO RHILBERT :SIZE :LEVEL
  IF :LEVEL = 0 [STOP]
  RT 90
  LHILBERT :SIZE :LEVEL-1
  FD :SIZE
  LT 90
  RHILBERT :SIZE :LEVEL-1
  FD :SIZE
  RHILBERT :SIZE :LEVEL-1
  LT 90
  FD :SIZE
  LHILBERT :SIZE :LEVEL-1
  RT 90
END
```

Their second, more elegant, version takes advantage of the similarity between LHILBERT and RHILBERT, uniting them into a single procedure. This elegance, however, is achieved at the cost of introducing a rather unnatural programming trick, namely an additional variable called PARITY that takes on the values 1 and -1 and, accordingly, keeps or reverses directions.

```
TO HILBERT :SIZE :LEVEL :PARITY
  IF :LEVEL = 0 [STOP]
  LT 90 * :PARITY
  HILBERT :SIZE :LEVEL-1 -:PARITY
  FD :SIZE
  RT 90 * :PARITY
  HILBERT :SIZE :LEVEL-1 :PARITY
  FD :SIZE
  HILBERT :SIZE :LEVEL-1 :PARITY
  RT 90 * :PARITY
  FD :SIZE
  HILBERT :SIZE :LEVEL-1 -:PARITY
  LT 90 * :PARITY
END
```

THE PROBLEM: CAPTURING SAMENESS

In Euclidean geometry, the idea of "sameness" in regard to plane figures is represented faithfully by the notion of *congruence*: congruent figures usually fit our perception of "same figure, differently placed". In terms of transformation geometry, congruence is effected by an isometry of the plane: two figures are congruent if and only if they can be brought to overlap by an isometry, that is, a combination of translations, rotations and line-reflections. In turtle geometry, the same notion of "same figures, differently placed" can be given a *procedural* representation: If two figures are "the same" they ought to be describable by a single turtle procedure (a combination of FORWARDs and RIGHTs). The two figures in question could then be drawn by the same procedure, varying only the initial turtle state i.e., its position and heading.[3]

As the Hilbert Curve example demonstrates, however, this is not always the case: two different procedures, RHILBERT and LHILBERT, are needed in turtle geometry in order to describe congruent mirror shapes. In practice one can of course solve the problem by using programming tricks

such as the PARITY variable above, but this only highlights the insufficiency of the *geometric* means available. (See the Classroom Implementation Section for a much simpler example demonstrating the same problem.)

A SOLUTION

We add a new turtle operation **FLIP**, which *turns the turtle upside down.* Thus the effect of FLIP is to switch the turtle's left and right. We can also say that FLIP changes the turtle *state* from "face down" to "face up" and vice versa.

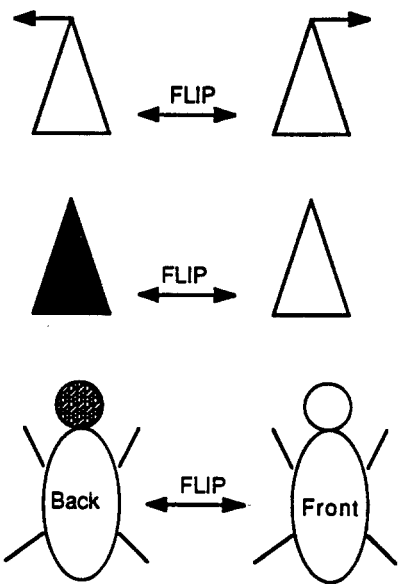

Fig. 2. The FLIP operation.

For a turtle living in the computer screen, "face down" actually means "face to the screen", so that the turtle's left is the same as the user's. On the other hand, when the turtle is in the "face up" state, i.e. facing the user, its lefthand side coincides with the user's right.

We now use FLIP to give a third description to the Hilbert curve, which seems to us to represent more faithfully the geometrical relations involved. Instead of distinguishing right from left, or introducing extraneous tricks, we observe that FLIP LHILBERT FLIP has actually the same effect as RHILBERT. Thus the basic similarity of the two can be captured in our

enriched language as:

```
TO HILBERT :SIZE :LEVEL
  IF :LEVEL = 0 [STOP]
  LT 90
  FLIP HILBERT :SIZE :LEVEL-1 FLIP
  FD :SIZE
  RT 90
  HILBERT :SIZE :LEVEL-1
  FD :SIZE
  HILBERT :SIZE :LEVEL-1
  RT 90
  FD :SIZE
  FLIP HILBERT :SIZE :LEVEL-1 FLIP
  LT 90
END
```

NOTE ON FORMALITY

We have adopted the same informal, anthropomorphic approach in introducing the new turtle operation FLIP, as is customary in the Logo literature. However, this is easily formalizable by considering turtle operations to be transformations of the "turtle plane" – the set of all turtle states (x, y, h, f). Here x and y are the turtle's coordinates, h its heading and f its face-state, which can assume the values "up" or "down". FLIP is now defined as the operation that changes the value of f from "up" to "down" and vice versa, leaving the other components of the state unchanged.

THE TURTLE GROUP: A VIEW FROM HIGHER MATHEMATICS

According to Klein's "Erlanger Program" geometries are classified via groups of transformations and their invariants. Thus Euclidean (plane) geometry is characterized by the group of all plane isometries – combinations of translations, rotations and line-reflections – which we shall call the *Euclidean group*. Similarly, turtle geometry is characterized by the group of all turtle operations – combinations of FORWARDs and RIGHTs – which we shall call the *turtle group*. From our discussion of turtle representations of the Hilbert curve, one suspects that the turtle group is in some sense "smaller" than the Euclidean group. Indeed, it can be shown (Leron

and Zazkis, in press) that the turtle group is isomorphic to the subgroup of *direct* isometries, i.e. combinations of translations and rotations (no reflections). However, the *extended turtle group*, with FLIP adjoined, is isomorphic to the entire Euclidean group. This is what we meant by saying in the introduction that FLIP brings the turtle's power to express invariance up to Euclid's.

In our isomorphism, incidentally, RIGHTs correspond to rotations about the origin, FORWARDs correspond to translations along the y-axis, and FLIP corresponds to reflection in the y-axis. Translations in general correspond to "heading-preserving" turtle operations, i.e. operations of the form ⟨RIGHT a FORWARD b RIGHT -a⟩. Similarly, general rotations and reflections correspond to suitable *conjugates* of the RIGHT and FLIP operations.

EXPRESSIVE POWER: A VIEW FROM COMPUTER SCIENCE

From a computer science perspective, the significance of our FLIP lies mainly in its *linguistic* aspects. Whereas the mathematical significance was seen to lie in the extension of the group of turtle operations to achieve greater range of invariance, here we are more concerned with the extension of the language to achieve greater *expressive power*. It should be emphasized that this does not mean that things can now be expressed that couldn't before. What is meant, rather, is that things can be *better* expressed than before, in the sense that we can express them in the simplest and most natural language for the given context. Unlike mathematics, where one is free to choose one's language to suit one's purpose, in computer science, unless special care is taken, one is often forced to distort one's perceptions to suit a given "general purpose" programming language. (Cf. the wide-ranging discussion of expressive power and abstraction in Abelson and Sussman (1985).) In this sense, the addition of FLIP enables us to better express invariance of shape in the turtle language, thus increasing its expressive power.

FLAGS AND BOARDS: A CLASSROOM IMPLEMENTATION

The following is a brief description of a classroom activity based on the above ideas, that we have actually carried out, in one form or another, with many groups of students ranging from elementary school to college.

(a) *Similarities and differences.* The activity, which is based on small-group work alternating with classroom discussion, starts with the follow-

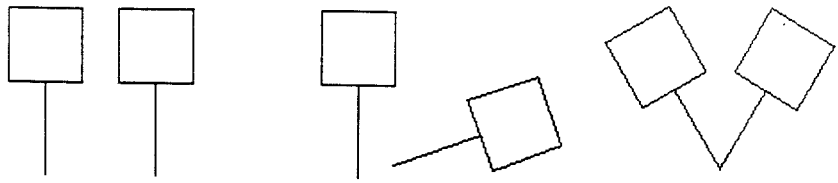

Fig. 3. Congruent boards.

ing question: What is similar and what is different about the patterns in Figure 3?

The students' work (with suitable help from the teacher) eventually crystallizes into three types of answers, depending on the language of description.

Using natural language, one may say that all the patterns are composed of two similar "boards", but the relative situation of the two boards in each pair is different.

Mathematically, each pair consists of two congruent shapes, but the transformation carrying one to the other (whose existence is guaranteed by the congruence) is different.

The third kind of description is a procedural one. Here we specify *how to* draw the given pattern, rather than *what is* its structure. For example, here is one possible description using Logo:

```
TO BOARDS1      TO BOARDS2      TO BOARDS3
  BOARD           BOARD           BOARD
  MOVE1           MOVE2           MOVE3
  BOARD           BOARD           BOARD
END             END             END
```

In this description, the mathematical fact that all the boards are congruent is expressed by the computational fact that the same subprocedure BOARD is used to draw all of them. In addition, the similarity in *form* of the three procedures (BOARD MOVE BOARD) reflects the similar *structure* of the three figures. In contrast, the different relative situation of the two boards in each pair is expressed by the three different MOVEs.

(b) *A problem.* The class is next given a slight variation of the above patterns (Figure 4), and the same question is repeated concerning their similarities and differences.

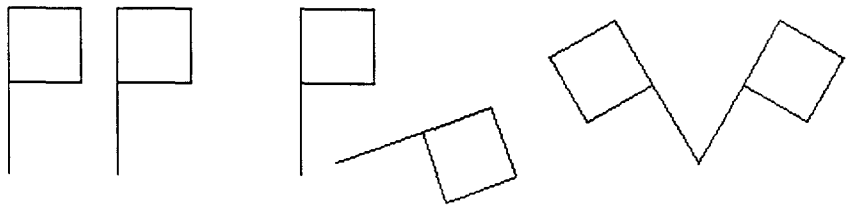

Fig. 4. Congruent flags.

Since the two figures look rather alike, we may expect the same analysis of similarity and difference to hold. In particular, we might expect our turtle description to have the following form:

```
TO FLAGS1      TO FLAGS2      TO FLAGS3
  FLAG           FLAG           FLAG
  MOVE1          MOVE2          MOVE3
  FLAG           FLAG           FLAG
END            END            END
```

To our surprise, we find that while the descriptions in the first two modes (natural language and mathematics) indeed remain the same, the expected turtle description doesn't quite work. Indeed, we need two different FLAG procedures (e.g. RIGHT.FLAG and LEFT.FLAG) to complete FLAGS3. Why is it that the two congruent shapes cannot be described by the same procedure?

This seemingly innocent problem may then lead to a discussion of direct and indirect isometries, the insufficiency of the turtle language in dealing with the latter, and the possibility of extending the turtle language by adding the new FLIP operation.

With this addition to the language, the procedure FLAG3 as defined above will indeed draw the third pair of flags in Figure 4, provided the subprocedure MOVE3 is defined as follows.

```
TO MOVE3
  RIGHT 45
  FLIP
END
```

On a more sophisticated level, our problem can lead to discussion of the turtle group and its relation to the Euclidean group, of Klein's Erlanger Program, and of abstraction as a means for extending the expressive power of the language.

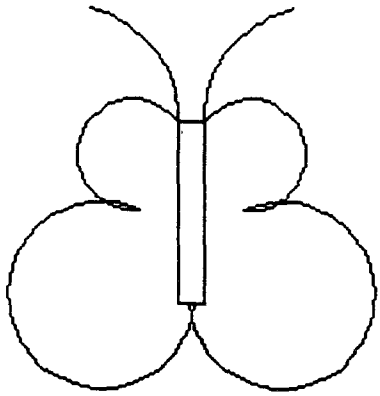

Fig. 5. A butterfly.

(c) *Another example.* To wrap up the activity, additional problems involving FLIP may be considered. For example, suppose we wish to draw the "butterfly" described in Figure 5.

A natural way to describe the butterfly is as composed of a body, two similar pairs of wings and two similar sensors. Since the two "similar" wings, as well as the sensors, are mirror images of each other, our new FLIP command enables us to translate this description directly into a Logo procedure of the form:

```
TO BUTTERFLY
  BODY
  WING FLIP WING
  SENSOR FLIP SENSOR
END
```

NOTES ON IMPLEMENTATION

In order to be able to experiment with the enriched turtle language, a computer simulation may be desirable. Thus, although our main concern in this article is with conceptual aspects of FLIP, we add a few remarks concerning its actual implementation in Logo. Perhaps the simplest way to achieve this is to enter in Logo any standard simulation of 3-D turtle (e.g. Abelson and diSessa (1981, pp. 144 ff.)), define FLIP as ROLL 180, redefine RIGHT as YAW, and ignore PITCH and TRAVEL.

Another way of implementing FLIP is via a suitable redefinition of LEFT and RIGHT. (Most current versions of Logo allow redefinition of

primitives. Alternatively, one can simply define NEW.RT and NEW.LT and ignore the old RT and LT.) The idea is to keep track of the current "flip-state" of the turtle, e.g. in a global variable, and to define FLIP as a procedure that switches the flip-state from "face down" to "face up" and vice versa. The new LEFT will act as the old LEFT when the face is down and as the old RIGHT when the face is up. The new RIGHT is defined similarly.

CONCLUSION

The need for the new turtle operation FLIP arose from the comparison of Euclidean and turtle geometries. Once defined, we have examined the new operation in three contexts: mathematical, where it was seen as an extension of the turtle group to one isomorphic to the Euclidean group; computational, where it was seen as an extension of the turtle language to enhance its expressive power; and educational, where it served as a basis for classroom activities that could lead to all the above themes in an elementary context.

Typically, it was not FLIP itself that has been the center of interest, but the sort of general concepts and structures that could be invoked with it. In particular we saw how FLIP could become a "thinking tool" which enabled us to break away from the constraints inherent in any given programming language.

NOTES

[1] Present address: Department of Mathematical Sciences, Northern Illinois University, DeKalb, IL 60115, USA.

[2] Throughout this note we use "turtle operation" in its mathematical, rather than computational sense. That is, operation as transformation or mapping, rather than a procedure that outputs a computational object. For example, FORWARD 50 is a turtle operation in this sense.

[3] By "the same" we also mean the same size. In order to treat the relation "same shape except for size", we need to replace congruence with similarity in the Euclidean case, and allow the SIZE variable to vary in turtle procedures.

REFERENCES

Abelson, H. and diSessa, A.: 1981, *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*, MIT Press, Cambridge, Mass.

Abelson, H. and Sussman, G.: 1985, *Structure and Implementation of Computer Programs*, MIT Press, Cambridge, Mass.

Coxford, A. F.: 1973, 'A transformation approach to geometry', in *Geometry in the Mathematics Curriculum*, 36th yearbook, NCTM, Reston, Va.

Leron, U. and Zazkis, R.: 1989, 'A turtle view on geometrical transformations (and vice versa)', *Logo Exchange* **7**, 7.
Leron, U. and Zazkis, R.: in press, 'Of geometry, turtles and groups', in Hoyles, C. and Noss, R. (eds.), *Learning Mathematics and Logo*, MIT Press, Cambridge, Mass.
Zazkis, R.: 1989, *Transformation Geometry and Turtle Geometry – a Group Theoretic Perspective*, Unpublished Doctoral Thesis, Technion-Israel Institute of Technology, Haifa.
Zazkis, R. and Leron, U.: 1989, 'Images of geometrical transformations: from Euclid to the Turtle and back', *Proceedings of PME XIII*, Paris.

Department of Science Education
Technion-Israel Institute of Technology
Haifa 32 000, Israel