

# A Sketch-Based System for Teaching Geometry

Gennaro Costagliola\*, Salvatore Cuomo<sup>◇</sup>, Vittorio Fuccella\*, Aniello Murano<sup>◇</sup>  
gencos@unisa.it, salcuomo@unina.it, vfuccella@unisa.it, murano@na.infn.it

\*Università di Salerno  
Via Ponte Don Melillo  
84084 Fisciano (SA), Italy

<sup>◇</sup>Università di Napoli Federico II  
Via Cinthia 26  
80137 Napoli, Italy

## Abstract

*Interactive Whiteboards (IW) have been massively introduced in schools. While the benefits of these devices in the learning process are well known, the dedicated software lack of functionality. In particular, most of the existing tools for IWs are an adaptation of classical software used on Personal Computers, mainly based on buttons and menus. The objective of the research presented in this paper is to completely re-think the interaction paradigm of the software for the IW, in such a way that the input is only composed of pen draws and hand-touches. In order to test this idea in practice, we apply it to the case of the teaching of geometry. Hence, more specifically, in this paper we propose a smart sketch-based tutoring system for IW that combines the recognition of hand-drawn geometrical shapes to the use of multi-touch gestures to support the editing of the drawn shapes.*

## 1 Introduction

Interactive Whiteboards (IW) are electronic devices having the dimension of a traditional whiteboard, on which it is possible to draw by using virtual pens or the touch screen system. The most advanced of them are enhanced with further capabilities, such as the possibility of supporting both interaction types and distinguishing among the media used for the interaction, making it possible to write with a pen, move objects with a finger and erase with the palm of the hand.

On one hand, a strong increase in the spread of IWs is expected in the future years. In 2006, the *Italian Ministry of Education* launched a plan for introducing 10'000 IWs in schools. With the present trends, a number of 5'300'000 IWs installed will be reached on a world-wide scale within 2015. In proportion, one IW out of 7 classes [8]. On the other hand, one of the drawbacks of these devices is the lack of software tools which can fully operate on them in order

to support teaching. Most of the available software tools, in fact, are still built upon the traditional WIMP (Window, Icon, Menu, Pointing Device) interaction style [4]. Even the most recent of them cannot take advantage from all of the advanced capabilities cited above.

In this paper we present a work in progress aimed at enhancing the interaction with IWs. Our work is focused on the support of teaching geometry in middle school. The teaching of geometry requires a massive use of annotated diagrams. In particular, angles, circles, and closed polygons have to be drawn and they can be annotated with labels representing the identifier or the value of a segment or an angle. Furthermore, the syllabi for middle schools include the explanation on how to calculate areas and perimeters of polygons.

Traditional geometrical software tools, such as *CABRI* [1], allow to draw geometrical shapes and to edit them. Nevertheless, they use the traditional WIMP paradigm, which, as remarked in [6], requires a significant amount of mode switching and loss of fluidity within the interface. Besides the time loss in the use of the traditional paradigm, it also requires a good acquaintance of the teacher with its menus and tools in order to perform a live lesson. As reported in [5], in many fields, the use of a sketch-based interaction, besides being more intuitive, can significantly improve the efficiency with which the same tools are used.

The prototypical system we propose can effectively take advantage from the capabilities of the more recent IWs, included the use of pen-based and multi-touch interactions. In particular, it is an intelligent sketch-based tutoring system for IW that combines the recognition of hand-drawn geometrical shapes to the use of multi-touch gestures to support the editing of the drawn shapes. It supports the recognition of angles, circles and regular polygons. The recognized shapes can be annotated and the annotated elements can be referred to in simple hand written expressions. These expressions can be used to set the value of the elements and to calculate measures of interest such as the area and the

perimeter of a drawn shape.

The rest of the paper is organized as follows: next section contains a brief literature survey on the most recent tools described in the scientific literature for aiding teaching through sketch-based techniques; Sections 3 and 4 are devoted to describe the system’s features design, respectively; in Section 5, a prototypical implementation of the system is described; lastly, some final remarks and a brief discussion on future work conclude the paper.

## 2 Related Work

Some prototypical software tools based on sketch technology for enhancing the teaching of disciplines in schools have been recently introduced. This is the case of math [6], geometry [4] and physics [3].

One of the first attempt for aiding mathematical problem solving is the *MathPad*<sup>2</sup> tool [6]. The tool supports a set of operations, including the recognition and the solution of hand written mathematical expressions, the recognition and the animation of sketched diagrams, as well as the innovative capability of letting the user associate expressions to diagrams. After an association is made, changes in mathematical expressions can be reflected as changes in the diagram and vice versa.

In [4], authors present a system which recognizes hand-drawn figures and hand-written proof scripts. Figures and proofs are written by the teacher in two separate views of the system interface. As for *MathPad*<sup>2</sup>, one of the most innovative capability is that of accurately establishing the correspondence between geometric components and proof steps: parts of the figures are highlighted as soon as they are used in the proof.

Similar to *MathPad*<sup>2</sup>, but enhanced with a custom physics engine, is the work described in [3]. The tool provides animation of student-drawn sketches based on the associated mathematics. Such an animation allows students to have a better understanding of physics concepts and can also help students in intuitive verification of their answers to problems.

The surveyed tools have scopes different from the one which has inspired our work. Among those cited, the most related to ours is the work done by Jiang et al. [4]. However, this tool is more oriented to theorem proving and does not fit the needs of middle school programs. Furthermore, it is not specifically designed to support IWs and does not take advantage from multi-touch and multi-modal interaction.

## 3 System Features

The tool we propose is an intelligent sketch-based tutoring system for IW that combines the recognition of hand-drawn geometrical shapes to the use of multi-touch gestures, in order to support the editing of drawn shapes. The recognition of annotations and short expressions is also supported.

The features of the system have been calibrated upon the middle school syllabus of Italian School. The syllabus published by the *Italian Ministry of Education* [2] foresees the use of audiovisual equipment to support the teaching of sci-







Interaction Type	Interpreter	Output
Pen-based 	Shape Recognizer	Shape  Annotation 
	Expression Recognizer	Expression 
Finger-based 	Gesture Recognizer	Gesture 

Figure 1. System architecture.

ence. In particular, for the teaching of geometry, its use is recommended, together with the recourse to drawings and diagrams. The use of experimental activities is foreseen in geometry as well as in other disciplines. According to the syllabus, the study of geometry must profit from a non-static presentation of the geometric figures, emphasizing their properties by showing their transformation.

More in detail, the syllabus for geometry includes some basic concepts, such as the study of plane and solid figures, the calculation of areas, perimeters, length of segments and magnitude of angles, the Pythagorean theorem. The processes of building figures is also included and foresee the use of tools such as ruler, set square, and compasses. More advanced topics include the Cartesian coordinate system and the Euclidean transformations.

The proposed system supports a set of features including the recognition and manipulation of angles and plane geometrical figures. The support of solid shapes and Cartesian coordinate systems is foreseen for future enhancements.

## 4 System Design

The system is composed of three different modules, as graphically summarized in Figure 1: (i) a *Shape Recognizer* which recognizes the sketched geometrical shape and annotations performed on it; (ii) an *Expression Recognizer*, which recognizes simple expressions referred to elements of the drawn shapes; (iii) a *Gesture Recognizer* which recognizes gestures performed on the shapes in order to edit them. The input is dispatched to the right recognizer, according to the type of interaction (pen-based or finger-based). Expressions can be distinguished from shapes and annotations, since they are performed on separated views.

### 4.1 Shape Recognizer

The *Shape Recognizer* supports the recognition of angles, circles and regular polygons. In our initial prototype, a shape must be completed in a single pen stroke. As soon as a stroke is entered, a *cuspid detection* procedure is run, in order to obtain the count and the positions of cusps in the stroke. The stroke is resampled in a sequence of equally

spaced points. A stroke can be firstly classified as a *segment*, an *angle* or a *shape* on the basis of the number of cusps it contains (0, 1, > 1, respectively). A further check for *closure* is applied on strokes classified as *shapes*. This is performed by checking that the distance between the last and first points of the stroke is smaller than a given threshold. If this check fails, the stroke is left unrecognized and it is automatically erased from the screen. A stroke classified as a *shape* is further analyzed in order to correctly classify it as a *circle* or a *polygon*. This step is performed by evaluating a measure of circularity as reported in [3]: the standard deviation of the angle subtended at the stroke's centroid by each line segment in the stroke is computed and for values smaller than a threshold, the stroke is classified as a *circle*, otherwise it is classified as a *polygon*. A *polygon* is further classified in a *triangle* (*equilateral*, *isosceles* or *scalene*), a four-sided shape (whose type can be *square*, *rectangle*, *diamond* or *rhomboid*), or an n-sided regular polygon (*pentagon*, *hexagon* and *octagon*).

As soon as a stroke is classified, it is immediately beautified. In past research on sketch recognition, beautification has not always been regarded as a desirable feature. In particular, Plimmer and Apperley [9] found that in the early stages of a design process, the users appreciated that the look and feel of a rough sketch was maintained. Nevertheless, the above findings were recorded on the recognition of user interface diagrams. These diagrams can contain many elements and the precision with which single elements are drawn is not essential for their understanding. In our case, instead, a single shape is entered and a higher precision is required, so we regarded beautification as desirable. The procedures for recognition and beautification of the entered strokes are based on the algorithms described in [7].

After they have been recognized, the shapes can be annotated. An annotation can be a *number*, a *letter*, or a *dashed line*. The annotation is assigned to the annotated elements by proximity: it must be drawn within a given distance threshold and is assigned to the closest element to which it can be referred. Given the sets of points of both a graphical element and the annotation, the distance is calculated as the minimum of the distances between any two of their respective points. A *number* can be used to set the length of the side of a polygon, the circumference of a circle, or the magnitude of an angle. A special convention is used for *letters*: capital letters must be associated to vertexes and lower-case letters to sides. A *dashed line* can be used to highlight parts of interest of a shape: the bisector of an angle, the height of a triangle, the radius or the diameter of a circle, the apothem of a regular polygon and so on.

## 4.2 Gesture Recognizer

The entered shapes can be edited through the use of multi-touch gestures. The semantics of the gesture set has been thought to enable the most desirable editing functionalities for each shape. First of all, it is possible to enlarge and shrink the size of a drawn shape. This is performed by *pinching inwards* or *outwards* of the shape.

If an angle has been drawn, its magnitude can be changed

through a two-finger *rotate* gesture: a finger must be pointed on the vertex and the other on one of its rays. With the first finger standing on the vertex, the second finger is rotated and the underneath ray is moved jointly with it. Another set of gestures is available for shapes. By keeping the finger pressed on an angle or on a side of a shape and then *swiping* the finger, it is possible to move the element and change the aspect ratio of the shape: with this gesture it is possible to change the type of the entered shape. For instance, a *rectangle* can be changed to a *square* by pressing on one of its short sides and then swiping it to narrow the long sides. A *scalene triangle* can be changed to *isosceles* by moving one of its vertexes.

When editing the shape, visual feedbacks are given to the user to point out that the shape is going to change its type. When an editing operation entails a change in the magnitude of an angle, a small circular arc centered at the vertex of the angle is shown. As soon as its magnitude is close to 90 degrees, a small square is shown in place of it. This is useful to let the user easily set right angles. The contour of a shape is also colored in green as soon as a change in shape type is detected. Numeric labels, with which sides and angles are annotated, are also changed when the shapes are modified. More precisely, their values change proportionally to the extent of the modification. The precision with which the new value is reported is the same of the original value. This holds for the number of decimal digits used to report the length of the side of a shape and the fractions of a degree (e.g., minutes and seconds) used to report the magnitude of an angle.

## 4.3 Expression Recognizer

The *Expression Recognizer* recognizes simple handwritten scripts. The scripts can be composed of different *steps*. In each step, geometrical elements of the drawn shapes can be referred with the letters used to annotate them. An *assignment* statement (with the = operator) can be used to set angle magnitudes and side lengths. Angles can be referred to using both the classical notation (three letters identifying its rays) and the simplified notation using only its vertex. It is worth noting that an assignment can cause a change in the shape proportions or type. In this case, a refresh of the view is triggered.

When the '?' special character is used on right-hand side of the assignment statements, it is automatically substituted by the system, provided that its value can be calculated using the data annotated on the shape or entered in the previous *steps*. *P* and *A* (capital) characters are used to refer to the perimeter and the area of the currently drawn shape, respectively. As soon as an expression is evaluated and recognized as valid, it is converted to printed characters.

The *Expression Recognizer* is a simplification of the one presented in [4]. Firstly, a partition of the hand writing is performed in order to distinguish different symbols in each expression. Then, the symbols are recognized. The set of symbols only includes number, letters and the '=' and '?' special characters. The semantic interpretation is easier than that used in [4], due to the smaller number of avail-

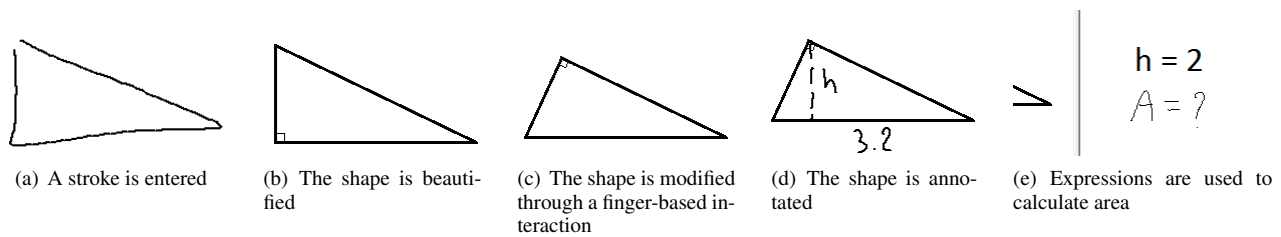


Figure 2. Some screenshots of the system interface

able symbols and operations. In our design, an expression is evaluated as soon as a timer expires from the time of the last input event. If the expression is syntactically correct, the step is executed and a beautified version of the expression is shown in place of the sketched one, otherwise, the system waits for further input.

## 5 Implementation

A prototype implementing most of the features of the above described system has been developed in Java language. The interface of the application is vertically divided into two parts. On the left side, figures can be drawn and annotated. The right side is for expressions.

Figure 2 shows some screenshots of the system interface as soon as an action is performed: in Figure 2(a), a stroke representing a right-angled triangle has been drawn; the system substitutes the sketched version with a beautified one (the lower-left vertex is recognized as a right angle, as shown in Figure 2(b)); by pressing and then swiping the finger on the upper vertex the shape is changed and a new right-angled triangle is produced (Figure 2(c)); the triangle is then annotated with a number (3.2) to set the length of its base. Furthermore, the height from the upper vertex is traced out and annotated with the  $h$  character (Figure 2(d)). In the right view, the expression  $h = 2$  is entered. As soon as the expression is recognized, the sketched version is beautified. The user can enter a new expression ( $A = ?$ , as shown in Figure 2(e)), in order to calculate the area of the triangle. As soon as the last expression is entered, the system substitutes it with a beautified expression reporting the value of the area.

## 6 Conclusion

In this paper, we have introduced a new software tool for the IW. The underlying idea is the replacement of the classical WIMP paradigm in favor of a more natural interaction, based only on pen draws and hand-touches. We have applied this idea to the case of the teaching of geometry. The proposed system is a smart sketch-based tutoring system prototype for IW that combines the recognition of hand-drawn geometrical shapes to the use of multi-touch gestures to support the editing of the drawn shapes. In a first informal test with pilot users, we have noted several positive aspects. First, we report an appreciation of the new interaction paradigm, particularly in relation to the ease of use of the tool: it is not required to remember menu layouts to use

specific functionalities. Second, we report an increased fluidity in teaching: indeed, the new interaction paradigm does not compel the teacher to introduce annoying pauses in the lesson, as with classical menu-based applications. Finally, due to the absence of menu and toolboxes, the whole space of the whiteboard is fully available for the lesson.

More formal experiments are necessary to fully evaluate the effectiveness and the efficiency of the system. In particular, as future work, we are planning to carry out an experiment in which specific tasks are executed. These tasks must simulate the execution of traditional and online short training sessions. As a result, we aim at comparing the efficiency of interacting with the IW with both the classical WIMP and the proposed paradigms. We will also evaluate the satisfaction of tutors and learners.

## References

- [1] Cabri ii plus. <http://www.cabri.com/cabri-2-plus.html>, 2011.
- [2] I programmi della scuola media - d. m. 9 febbraio 1979 (in italian). <http://www.edscuola.it/archivio/norme/programmi/media.html>, 2011.
- [3] S. Cheema and J. J. LaViola, Jr. Applying mathematical sketching to sketch-based physics tutoring software. In *Proceedings of the 10th international conference on Smart graphics*, SG'10, pages 13–24, Berlin, Heidelberg, 2010. Springer-Verlag.
- [4] Y. Jiang, F. Tian, H. Wang, X. Zhang, X. Wang, and G. Dai. Intelligent understanding of handwritten geometry theorem proving. In *Proceedings of the 15th international conference on Intelligent user interfaces*, IUI '10, pages 119–128, New York, NY, USA, 2010. ACM.
- [5] L. B. Kara and T. F. Stahovich. An image-based, trainable symbol recognizer for hand-drawn sketches. *Comput. Graph.*, 29:501–517, August 2005.
- [6] J. J. LaViola, Jr. and R. C. Zeleznik. Mathpad2: a system for the creation and exploration of mathematical sketches. *ACM Trans. Graph.*, 23:432–440, August 2004.
- [7] B. Paulson and T. Hammond. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, IUI '08, pages 1–10, New York, NY, USA, 2008. ACM.
- [8] A. N. per lo Sviluppo dell'Autonomia Scolastica. La lavagna interattiva multimediale (in italian). <http://www.indire.it/>, 2011.
- [9] B. Plimmer and M. Apperley. Interacting with sketched interface designs: an evaluation study. In *CHI '04 extended abstracts on Human factors in computing systems*, CHI EA '04, pages 1337–1340, New York, NY, USA, 2004. ACM.