SketchIt: Simple Vector Scheme Editor for Mobile Devices with Touch Screen

Vitaly Petrov, Evgeny Linsky
SUAI
St. Petersburg, Russia
{vit.petrov, evlinsky}@vu.spb.ru

Abstract

SketchIt is a simple vector scheme editor for mobile devices with touch screen. The main feature of editor is auto recognition of figures. This helps users to draw an accurate and neatly schemes quickly. The paper describes details of figure recognition algorithm.

Index Terms: Image recognition, Qt, Maemo, MeeGo.

I. Introduction

Nowadays almost everybody, who have been involved in a team work, is familiar with the following problem. During a conference or a meeting there appears a necessity to draw a scheme or a graph. It might be a system architecture, device structure or a project schedule. It should be possible to draw such scheme very fast using mobile device, but the result should be accurate and neatly.

In existing scheme editors (MS Visio, OpenOffice Draw) users select the figure from some predefined set and place it somewhere on the screen. But this is not very fast process. We propose another solution. User just draws sketch of figure on the touch screen using finger or stylus, application recognizes drawings and converts them to vector format (see fig. 1).

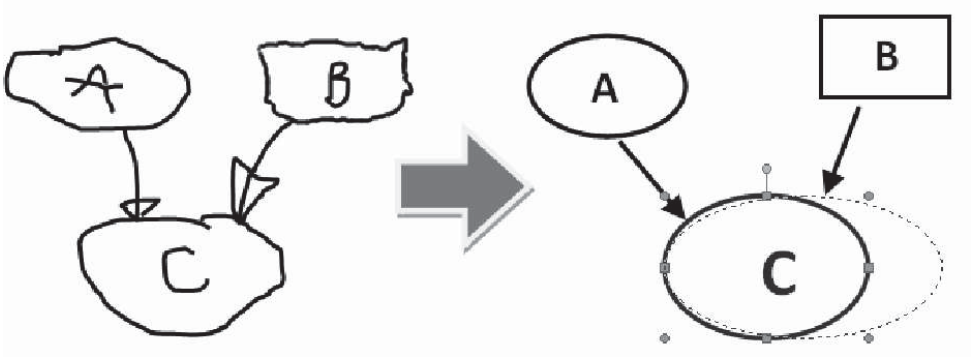

Fig. 1. SketchIt kernel: figures auto recognition

The functionality of the editor includes the following features:

- Auto recognition of figures (basic set: line, arrow, ellipse and rectangle);
- Moving, deleting and resizing of figures;
- Binding text to figures;
- Undo and redo;
- Export to PNG and SVG formats.

The paper has the following structure. The section II describes recognition of user actions: drawing, moving, resizing of figures. The section III reports the details of figure auto recognition. Implementation is discussed in section IV. Section V concludes the paper and gives ideas for future work.

II. User Actions Recognition Algorithm

Before starting image recognition, it is necessary to recognize the user's action. It could be drawing of new figure or moving, resizing, labeling of existing figure. Instead of adding extra buttons ("Set text", "Move", "Resize", "Bind text"), the heuristic algorithm (fig. 2) is proposed. It selects the user's action base on coordinates of the first point and the type of movement. The deleting of figure is performed by complex gesture "cross-hatching" of figure. Such complex gesture is recognized by figure recognition algorithm and described in the next section.

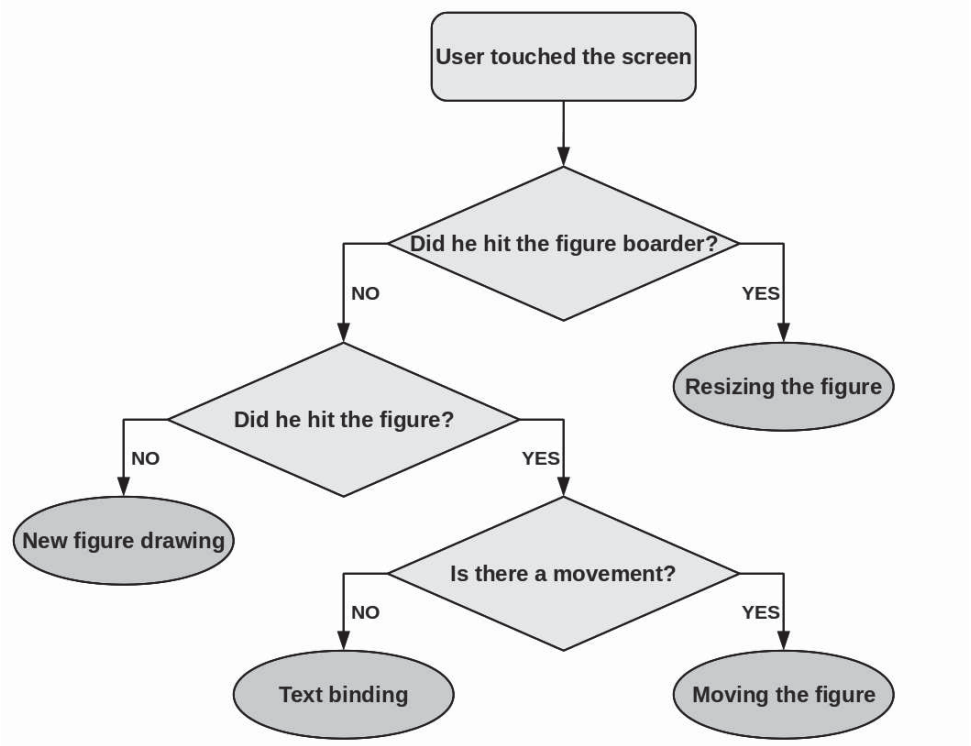

Fig. 2. Actions recognition algorithm

When user touch a point, that belongs to two or more figures, the following procedure is used. The touch of border has higher priority than general touch of figure. If priority is equal for several figures, the older one will be chosen.

III. Image Recognition Algorithms

For the recognition algorithm, the curve, drawn by user, is a sequence of points with two coordinates: x and y. To recognize the image, the 3-steps algorithm is used. Firstly, from the sequence of point the contour of the figure should be detected (subsection III-A). Then, using the sequence and contour the figure type is determined (subsection III-B). Finally, base of detected figure type the appropriate recognizer of figure parameters is executed (subsection III-C).

A. Contour Detection

The contour detection algorithm is very simple. It is necessary to find two points: $A(x_{min}, y_{min})$ and $B(x_{max}, y_{max})$.

B. Figure Type Recognition

The figures type recognition procedure has a tree structure and is shown on the fig. 3. In our application to delete figure one should cross-hatch it (fig. 4). Hatching is polyline, so it is recognized using the similar algorithm as lines.

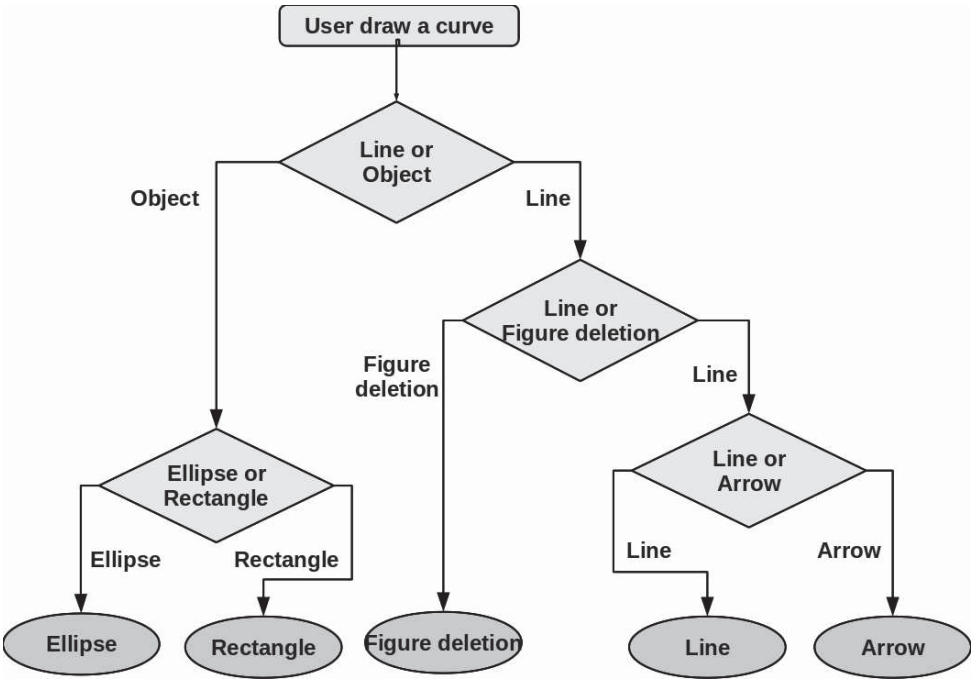

Fig. 3. Figure type recognition algorithm

Firstly, it is necessary to chose the figure class: a line/arrow, or a convex object. A convex object is usually drawn as follows. User starts a curve somewhere on the screen, move a stylus/finger for a while and, finally, come back to the start point. It means that first and last points are situated near each other. So as a criterion there can be used a fraction

$$E = \frac{|x_{begin} - x_{end}| + |y_{begin} - y_{end}|}{w + h},$$

where x_{begin} and y_{begin} are coordinates of the first point, x_{end} and y_{end} — of the last one, w is a contour width, and h is a contour height. After calculation, E can be compared with a defined coefficient (in the current version it is equal to $\frac{1}{3}$). If E is lower, the drawing is considered to be an object, otherwise, a line.

Then, if it is a line, going from up to down, and there are more than 5 direction changes, it is hatching. Hatching means that figure below the curve should be deleted. Otherwise, curve is supposed to be a line or an arrow. To chose between these two types, it is proposed an algorithm, calculating the number of turns for more than 90 degrees. When drawing has an arrowhead, user makes such a turn at least 3 times. And this is a formal criterion for choice between line or arrow.

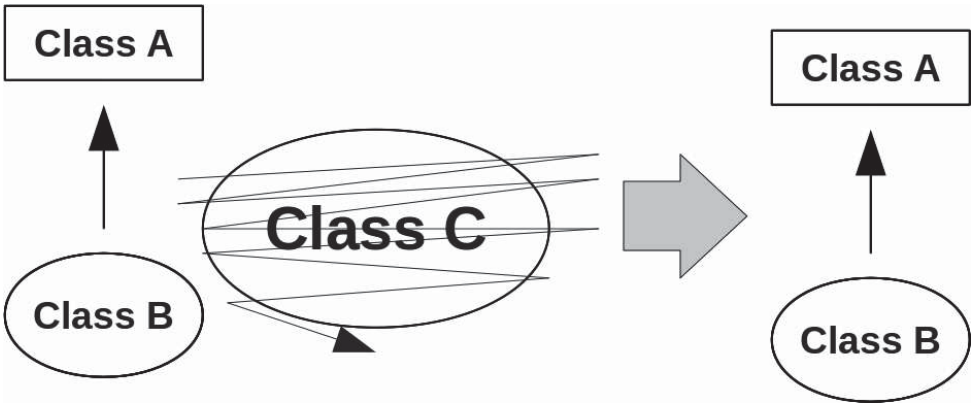

Fig. 4. Figures deletion sketch

The method for distinguishing the object type (ellipse or rectangle) is computation of the distance from each point of the curve to the contour (fig. 5). The distance is calculated using the following formula:

$$H = \frac{1}{N} \cdot \sum_{i=0}^{N} (\frac{L_x^{(i)}}{w} + \frac{L_y^{(i)}}{h}).$$

The smaller this value is, the more the object looks like a rectangle. The current threshold for H is 0.3.

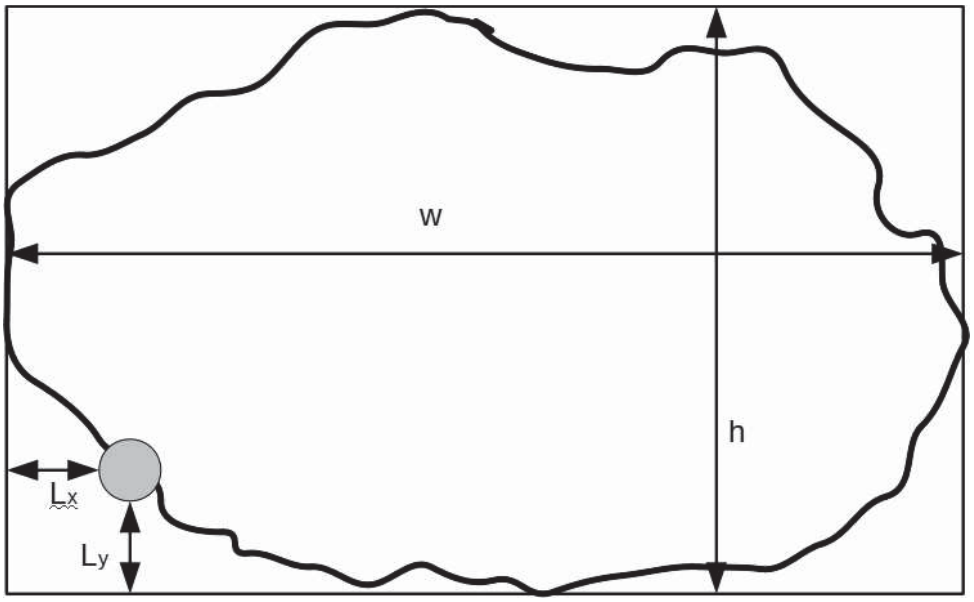

Fig. 5. Ellipse / rectangle chose

C. Figure Parameters Recognition

After the figure type is recognized, its parameters should be calculated. This part is also simple and works as follows (fig. 6).

- *Line:* A — the first point from the sequence; B — the last point from the sequence.

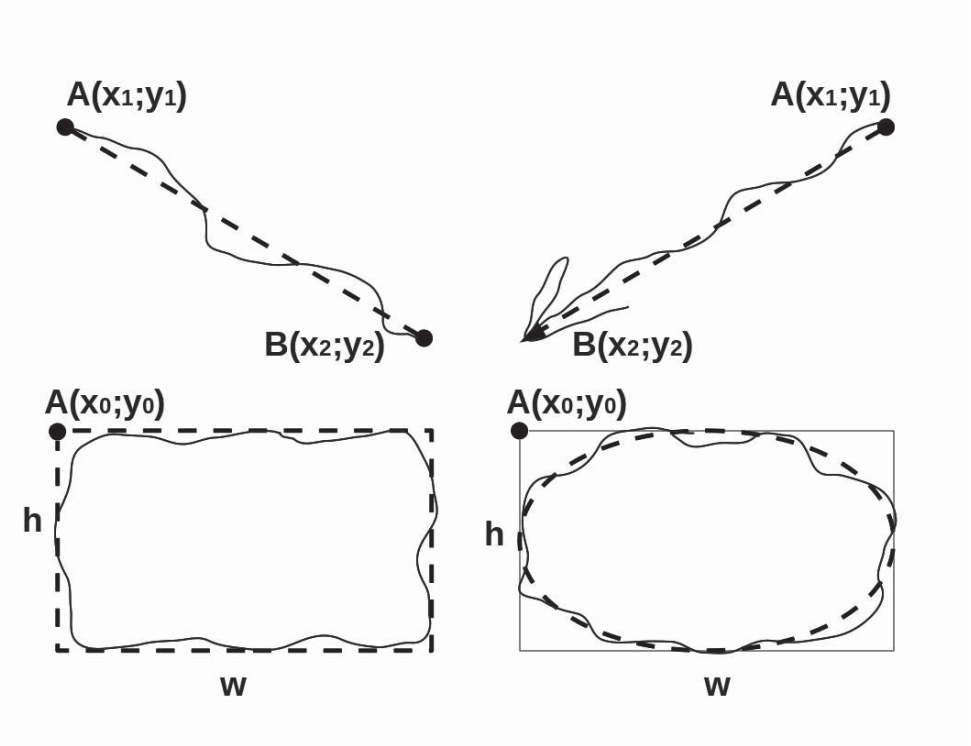

Fig. 6. Figure parameters recognition

- *Arrow:* A — the first point from the sequence; B — the first turn for more than 90 degrees.
- *Ellipse:* x_0 and y_0 — coordinates of the upper left point of the contour; w and h — contour width and height respectively.
- *Rectangle:* similar to ellipse.

IV. Technical details

The application was written on C++ language with usage of Qt [1] libraries. Currently there are builds for Maemo [2] and Meego [3]. The port for Symbian [4] will be available soon. Screenshot of Maemo version is presented on fig. 7.

SketchIt code consists of 5 main modules:

1. *Model* stores the set of figures and their parameters;
2. *View* is responsible for the image drawing and user input;
3. *Drawer* creates image from the set of figures;
4. *Recognizer* recognizes the figure type and calculates coordinates from the curve drawn by user;
5. *IO* supports SVG and PNG formats.

SketchIt is already available from garage [5] and OBS [6] servers, and will be published in Nokia Ovi Store [7] and Intel AppUp [8].

V. Conclusion

In this work we describe figure recognition algorithms for simple vector editor targeted for mobile devices with touch screen. Algorithm could recognize several user actions (moving, resizing, deleting) and simple figures (arrow, line, rectangle, ellipse).

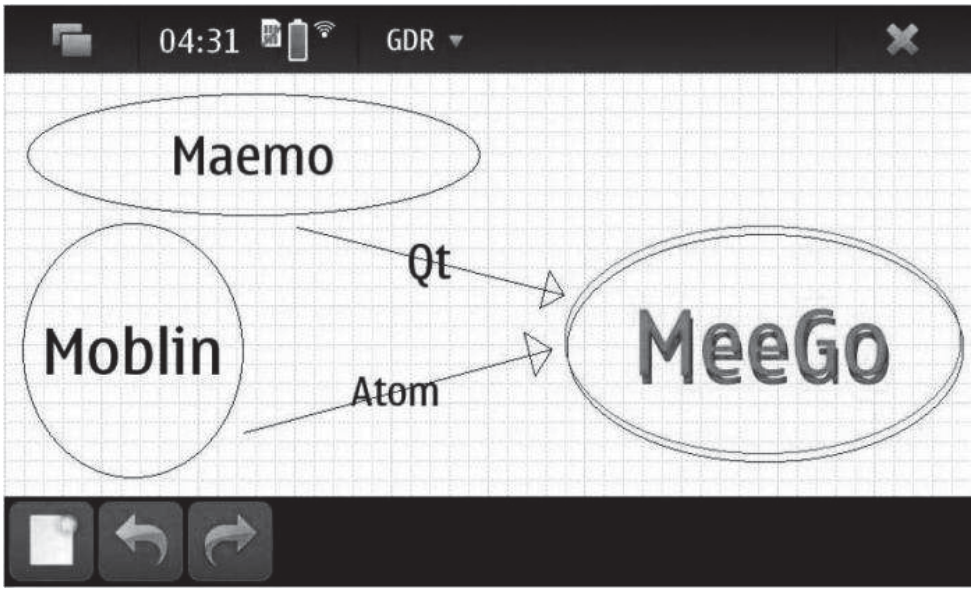

Fig. 7. Current version interface

There are two main directions of SketchIt development:

- integration with web document storage services (e. g. Google Docs);
- adding a possibility to connect and group figures in order to move and resize them together (as it is done in Visio).

REFERENCES

[1] Nokia corporation, "ross-platform application and UI framework Qt," http://qt.nokia.com/products.
[2] Maemo Community, "Official Maemo OS community web site," http://maemo.org.
[3] MeeGo Community, "Official MeeGo OS community web site," http://meego.com.
[4] Nokia Corporation, "Official Symbian OS web site," http://symbian.nokia.com/.
[5] "SketchIt for Maemo web page," https://garage.maemo.org/projects/sketchit.
[6] "SketchIt for MeeGo web page," https://build.pub.meego.com/project/show?project=home:vitpetrov:sketchit.
[7] Nokia Corporation, "Nokia Ovi Store," http://www.ovi.com.
[8] Intel Corporation, "Intel AppUp," http://www.appup.com.