Real-time visualization of large vector HP-GL based maps

Maxim Tyulin, Yulii Ketkov
Department of Computational Mathematics and Cybernetics,
Nizhny Novgorod State University,
Nizhny Novgorod, Russia
tyulin@mail.ru, ket@city.ru

Abstract

This paper surveys a software system which is being developed for vector geoinformation systems preview purposes. The software meets the main criteria for such systems – high speed preview – in order to let the user navigate through a map contents without a perceptible delay. It has been achieved, mostly, because a fast indexing algorithm has been developed and implemented. The system is based on the Hewlett-Packard Graphics Language (HP-GL) vector format – public available and widely used in real maps representation.

Keywords: *vector graphics, geoinformation system, HP-GL.*

1. INTRODUCTION

HP-GL is a vector format comprising a set of instructions (or commands). Each command has 4 parts – a mnemonic operation code (two-letter sequences which stand for a particular action), parameters (digits or chars), delimiters (comma, plus or minus signs) and the end mark. For instance, the PD10,-3; instruction means "Pen Down to the (10,-3) point" [1].

Typical digital map can contain up to several thousands of such instructions and it is obvious that we need to interpret all of them when draw the whole image. But in the case of zooming of any part of the map this approach is not acceptable – there are a lot of instructions which will not affect the desired part of the map and therefore it is not necessary to survey them at all. Thus, the problem of detecting commands which need to be processed is the most important for accelerating the map preview process.

2. INDEXING

For this purpose a fast indexing algorithm has been developed. Indexing is a process of marking minimum possible set of instructions which we need to interpret in order to form the selected part of the map. Also, we have to take into account that indexing should be performed much faster than processing of all commands. In general, the process is divided into 3 parts:

- Indexing of instructions which change any output parameters
- Indexing of instructions which draw polylines
- Indexing of all the rest drawn instructions

There are a few commands which just adjust some output parameters (e.g. pen color or width changing, polygon mode opening etc.). All of them are always added to the index array that is they will be always processed. This does not take much time but ensures that we will never miss any output adjustments.

All the instructions which do not draw polylines are indexed in the following way: we check whether there is an intersection of the zoomed part of the map and the boundary rectangle for the instruction – if so, probably, this command affects the selected rectangle, it is also possible that the intersection is not empty but the instruction's image lies outside the rectangle – nevertheless, we add the command to the index array. Both situations are represented in Figure 1.

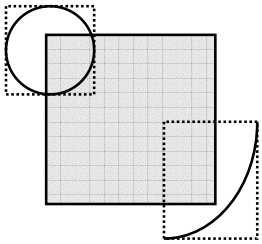

Figure 1: Possible intersections of instructions and the zoomed part of the map.

We do not use the same method for indexing the instructions which draw polylines because widely distributed the situation when the rectangles are intersected but a line will not be really seen on the screen (see Figure 2). Moreover, typical vector map contains thousands of small segments and it will take extremely much time to intersect all rectangles. For this case more sophisticated algorithm has been developed [2].

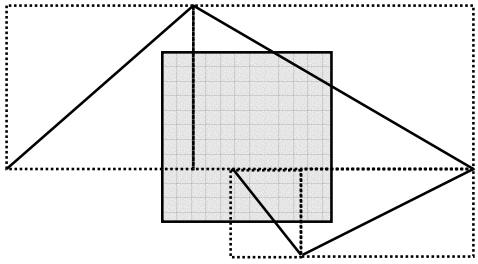

Figure 2: Possible intersections of polylines and the zoomed part of the map.

The zoomed part of the map divides the plane into 9 parts (see Figure 3). Rectangle 4 corresponds to the selected part. First of all we need to determine what part a segment's end point belongs to (we name this characteristic "number"). If coordinates of the point belongs to the rectangle 4 (i.e. the number is 4), then we index this instruction. Otherwise it does not indicate that this command can be skipped, it is still possible that the line intersects the selected rectangle. That is why we need to consider the next point concurrently storing a number of currently calculated point. This allows us to gain time in the future – since each segment's end point is a start point for the next segment we will not have this parameter calculated twice.

0	1	2
3	4	5
6	7	8

Figure 3: The zoomed part of the map divides the plane into 9 parts. The selected part is rectangle 4.

Next, according to the just calculated points' numbers we minimize total amount of required operations by using the following matrix:

```
int ij[9][9] = {
    {-1,-1,-1,-1, 5, 2,-1, 1, 4},
    {-1,-1,-1, 0, 5, 2, 3, 5, 3},
    {-1,-1,-1, 0, 5,-1, 4, 1,-1},
    {-1, 0, 0,-1, 5, 5,-1, 0, 0},
    { 5, 5, 5, 5, 5, 5, 5, 5, 5},
    { 2, 2,-1, 5, 5,-1, 2, 2,-1},
    {-1, 3, 4,-1, 5, 2,-1,-1,-1},
    { 1, 5, 1, 0, 5, 2,-1,-1,-1},
    { 4, 3,-1, 0, 5,-1,-1,-1,-1} };
```

Columns and rows of the matrix are numbers of previous and current points. Each element means an operation we have to perform with the investigated segment. -1 means that the segment lies outside the rectangle and therefore the instruction can be skipped. 5 means that the segment definitely intersects the rectangle. All the rest values indicate what exactly edges of the rectangle the line can intersect. Finally, only in the case of having values 0 till 4 we check intersections as per that value and if it is not empty we index the instruction.

3. HP-GL VIEWER

The algorithm described above has been implemented in HP-GL Viewer. This software system is being developed at the moment and already supports 28 widely used HP-GL instructions. The user just has to specify an input vector file and the map will be drawn immediately. Map preview section will contain a small copy of the map where the user can observe the currently selected part of the map marked with the red rectangle. In the information field the following is available: the time elapsed since the latest draw operation, a boundary rectangle, current point coordinates, the current axes direction. The direction can be changed easily by selecting another value from the dropped down list – the image will be redrawn accordantly. After the map is drawn the user can zoom any part of the map readily by selecting the desired part with the mouse. For a snap shot of the system see Figure 4.

After having some testing with big HP-GL files (about 10 Gb) it has been found out that a speed of composing the zoomed image almost does not depend on the location of the rectangle. For instance, on a PC having Intel Pentium III 2000 MHz and 256 MB RAM installed, this map is drawn over 3.2 seconds. If we do not use an indexing and want to zoom, for example, any 1/8 part of the map, it will take us 3.2 seconds as well. Having our algorithm applied we accelerate the process up to 0.51 seconds. Obviously, our advantage is arising along with the size of a vector file. When a file is too small we will not have any acceleration compared to the case when a file is so huge that drawing the whole map could make users wait for a while.

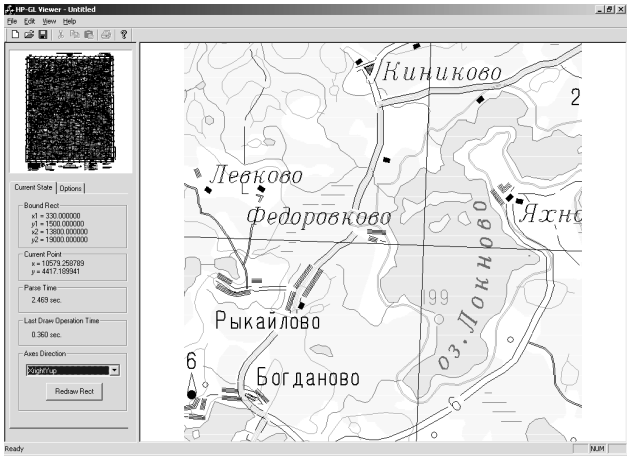

Figure 4: A snap shot of HP-GL Viewer.

4. FUTURE WORK

HP-GL Viewer is a system which is currently being advanced and improved. This process is quite easy to perform due to its object-oriented nature and thought-out internal design [3]. The nearest plans are expanding a set of supported instructions, implementing additional functionality while facing the end user requirements.

5. CONCLUSION

The fast indexing algorithm described in this paper has been successfully implemented and tested in HP-GL Viewer – a software system allowing users to perform fast navigation through a map contents. The software itself is unique since there are no other public available systems providing the same speed and convenience abilities.

6. REFERENCES

[1] *The HP-GL/2 and HP RTL Reference Guide. A handbook for Program Developers.* Addison-Wesley Publishing Company, 1995

[2] Y. Ketkov, S. Kirianov. *Time optimization in the vector large scaled graphic images display* // Proc. of the Conf. of Computer Graph. and Visual. – GraphiCon'2001, p.215-216

[3] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison-Wesley Longman, Inc., 1995.

About the authors

Maxim Tyulin is a post-graduate student of Nizhny Novgorod State University, Department of Computational Mathematics and Cybernetics. His contact e-mail is tyulin@mail.ru.

Yulii Ketkov is a full professor of Nizhny Novgorod State University, Department of Computational Mathematics and Cybernetics; department chief of Research Institute of Applied Mathematics and Cybernetics of Nizhny Novgorod State University. His contact e-mail is ket@city.ru.