TrollEdit – different approach in editing of source code using graphic elements

Lukáš TURSKÝ, Jozef KRAJČOVIČ, Maroš JENDREJ, Marek BRATH,
Ľuboš STARÁČEK, Adrián FEJEŠ*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
tp-team-10@googlegroups.com

1 Introduction

Today programmers use editors and IDEs that usually use simple color highlighting without any sign of graphic enrichment features. However enriching the source code with graphic elements can be beneficial for the understanding of the structure of given code and thus lead to better understanding of its structure and meaning for the programmer. This basic observation is the driving idea behind *TrollEdit*.

TrollEdit is an experimental editor that tries to enrich source code with graphical elements for easier manipulation. Source code editing can sometimes be very problematic especially when reviewing unknown code that the programmer is not familiar with. Most of the time programmers are trying to familiarize themselves with the syntax of the source and only then follow to analyze its semantic meaning. *TrollEdit* tries to address both of these steps by enriching the text editor with graphical elements instead of relying on colorized text.

TrollEdit is a running project, which started as a research idea by team *Ufopak*. The team explored the possibility of using abstract syntactic trees instead of simple coloring rules to enrich the presentation of source code and its manipulation. Our goal is to further improve upon the existing core functionalities so *TrollEdit* can be deployed for real development tasks.

2 Motivation and current achievements

The core functionality of *TrollEdit* is based on the use of *LPeg* pattern matching library. Using this library we are able to parse source code according to its grammar into an abstract syntactic tree (AST). This data is then used to enhance the text visualization using the *Qt framework*, which provides the needed graphic functionalities. The proper combination of these two technologies made this editor possible by utilizing the *Lua* programming language and an interface between the two technologies. For performance reasons the current project relies on the much faster *LuaJIT2* implementation.

Using a scripting language and the *LPeg* [1] library we are able to parse the content on any open file into abstract syntactic three (AST) assuming that we have a matching language grammar.

---

* Master degree study programme in field: Software Engineering
Supervisor: Dr.Peter Drahoš, Institute of Applied Informatics, Faculty of Informatics and Information Technologies STU in Bratislava

IIT.SRC 2012, Bratislava, April 25, 2012, pp. 1–2.

Created hierarchical order is then used to visualize and interactively manipulate the structure of the program. Users can than easily control and shift whole blocks just as they are displayed without any usual problems from conventional text editors (text indent, selection etc.).

On top of that, the idea of *literate programming* by Donald E. Knuth [2] is explored as we can easily document parts of source code with comments that can contain rich text content for documentation purposes.

Based on the work done by *Ufopak* we aim to optimize the generation of the AST by utilizing parallel processing and a more efficient way to access the generated data. Among other prominent changes we are introducing, is the ability to switch between graphically enhanced and legacy visualization of the source code. In *text-mode* the editor works as any other common editor and does not interfere with the editing process so productivity of programmers is not affected when writing code. However in second mode user gets the full potential of enhanced editor, where edited text is represented in graphic blocks as we can see on *Figure 1*. This can be interactively manipulated, printed, exported as PDF or saved for documentation.

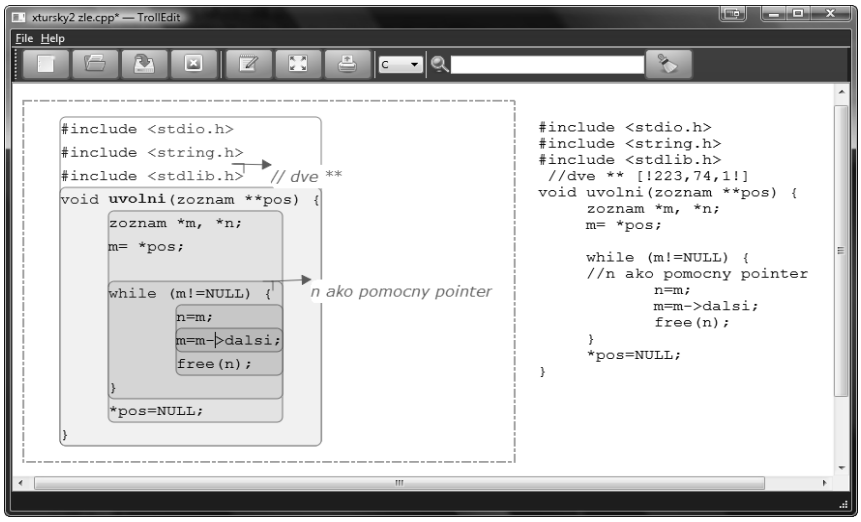

Figure 1.Visualization of two files opened in editor, one in graphic-mode and other one in text-mode.

3 Conclusion

All our contributions are aimed to ensure that TrollEdit will be a practical editor designated with extensibility, efficiency and flexibility in mind. New grammar can always be added for support of new languages without any invasion to editor. For example we can use the editor features to create a grammar for simple ToDo list management as part of the evaluation process. The visual presentation of the editor is also extensible as it relies on style sheets defined using the CSS format.

References

[1] Roberto Ierusalimschy: A text pattern-matching tool based on Parsing Expression Grammars, 2009, PUC-Rio, Rio de Janeiro, Brazil

[2] Knuth, D.E.: Literate Programming, 1992, Stanford University Center for the Study of Language and Information, Stanford, CA, USA, 1992.