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

Lukáš Turský, Jozef Krajčovič, Maroš Jendrej, Marek Brath,   
Ľuboš Staráček, Adrián Feješ [[1]](#footnote-2)\*

Slovak University of Technology in Bratislava

Faculty of Informatics and Information Technologies

Ilkovičova 3, 842 16 Bratislava, Slovakia

tp-team-10@googlegroups.com

**Abstract.** Today editor’s IDE usually use only simple highlighting color without any sign of graphic enrichment features. However enriching the source code through graphic elements can get a better understanding of the structure of given code and thus facilitate the programmer to read and manipulate the code more efficiently, which gave a rise to a TrollEdit. This editor presents a way to work with source code using graphic elements for representation. In this paper we present our experiences and contribution to this editor together with our results about the possibilities of such approach in editing of source code using graphical elements.

# Introduction

**////problem and the state of the art in the respective area are introduced;**

In these days editing of source code can be sometimes very problematic especially when reviewing an unknown one. Here programmer is most of the time just trying to get used to a different way in which code could be written. That is just one of possible problems, which TrollEdit is trying to resolve by using enhanced source code visualization represented in combination with graphic elements.

We have to mention, that this project is based on the existing project research of *Ufopak* team, who originally created this editor in some form and on which we based our project work. Generally our main goal is to further improve provided core functionalities in a specified way, so in the final version *TrollEdit* can be deployed into real environment.

# Motivation and current achievements

The core functionality of *TrollEdit* is based on *Qt framework*, which provides graphic functionalities, in cooperation with *Lua* script language. The proper combination of these two technologies made possible to fulfill all performance challenges so working with editor should be without any restraints and in real time. We can say that enhanced text visualization and representation in block hierarchies were the main ideas describing our motivation.

Generally content of opened file is analyzed and enriched thanks to the language grammars implemented using the *Lua* based *LPeg* pattern matching library, which parses the source code into an abstract syntactic tree (AST). Consequently this content is represented as blocks in a hierarchical order. Here the whole new level of working with text is provided as users can 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, as the idea of *literate programming* by Donald E. Knuth [2] was mentioned, we can easily document parts of source code with comments, which are also represented in a form of text blocks, and in the end we can generate this comments in a form of document.

Based on our research several achievements were done. To the most significant certainly belongs implementation of parallel processing together with optimized processing of AST tree. We introduced the two-mode functionality, which TrollEdit is capable of working in and these are the *text-mode* and *graphic-mode*. In first one, user can work just like in any other text editor he is used on. 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 . This can be printed, exported as PDF or saved for documentation.

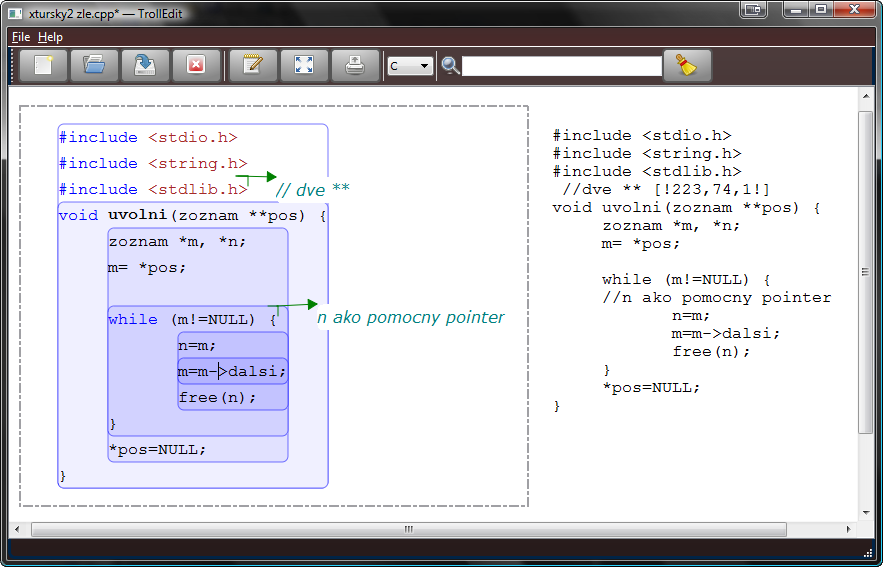


Figure . Visualization of two files opened in editor, one in graphic-mode and other one in text-mode.

Also thanks to the language grammar which describes edited file we can put styles and colors to recognized language types such as variables, loops or functions. So not only is the edited text visualized in blocks, but individual semantic types could be formatted via enclosed .css styles.

# Conclusions

All our contributions have as a main goal to ensure, that this editor is designated with extensibility, efficiency and flexibility in mind. We can always add new grammar to receive support for another language without any invasion to editor, for example just as we are now trying to create grammar for ToDo list implementation.

We want to provide modern user interface for users, which will be easy and intuitive to use. We suppose this editor as a perfect tool mainly for programmers, who want to hold all the information, notes and external files in one place and possibly right next to the source code.

# References

1. //i don´t know which .
2. Knuth, D.E.: Literate Programming, 1992, Stanford University Center for the Study of Language and Information, Stanford, CA, USA, 1992.

1. \* 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 [↑](#footnote-ref-2)