TrollEdit - New Way of Source Code Editing

Andrej FOGELTON*, Ondrej KALLO*, Peter ONDRUŠKA*
Martin PALO†, Jakub UKROP*

Slovak University of Technology
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
ufopak@googlegroups.com

Editing a source code is asignificant part of a programmer's job. Most of the time programmers modify and debug existing code rather than create new code; this refactoring of source code always takes a lot of time. *TrollEdit* accelerates the casual way of source code editing by providing users with graphic view of the inner structure and logic of programs. Additionally, main principles of literate programming are supported, meaning that source code documents can contain fully formatted documentation.

Block hierarchies and their visualization are the main ideas underlining this project. Source code written in any of textual programming language consists of number of lexical, syntactic and semantic units – logical "chunks of code" or blocks. Enriched language grammars implemented using the *Lua* based *LPeg* [1] pattern matching library are used to parse the source code into an abstract syntactic tree (AST). Support of any programming language or any formal language is limited only by availability of its grammar, which can be easily transformed into the *LPeg* syntax. Every node in the AST represents what we call a block.

By using block hierarchies based on AST rather than simple syntactic rules of a language, our editor can offer features not common in most source code editors. These affect refactoring tasks such as fast selection of blocks without need to select blocks by cursor, enhanced drag-and-drop within a program by snapping to available spots or customized code folding by replacing folded blocks by desired summary information etc.

Since everything within a program is a block, some may end up containing too many sub-blocks or too small to be useful e.g. a single semicolon. Therefore, it is essential for users to be able to control and shift the level on which the current editing is going

---

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

IIT.SRC 2010, Bratislava, April 21, 2010, pp. 1–2.

on according to their momentary needs. Figure 1 illustrates the concept of the block hierarchy on a sample C code fragment.

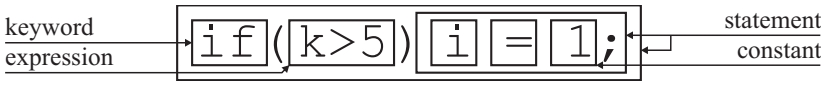

Figure 1. Example of block recognition

TrollEdit is based on the graphics view framework provided by *Qt* toolkit. Blocks can contain graphic items which can be have different shapes, font or even images, rich text, UML diagrams etc. Every block supports layout, movement, drag-and-drop and text editing features. By providing constraints on where certain items can be dropped we can maintain a structure of the document, since it is not desirable to be able to drop anything anywhere. Blocks have to be moved with their descendants and resized according to size and layout of their descendant blocks. A programmer can fold blocks, which are naturally bound (functions, cycles, etc.) or define own blocks using menu items.

By relying on the former mentioned display and layout capabilities of the block layout system, *TrollEdit* can be used as a document editor for writing documentation directly into the code. Similarly to standard language comments these blocks will be omitted from the executable code, so code and documentation can coexist without any side effects. This was inspired by the literate programming idea conceived by Donald E. Knuth [2]. As opposed to the original approach no special tags or macros are necessary to separate the source code from documentation. To achieve better readability ”documentation blocks” support full text formatting (font size, bold, italics, colour, etc.). Moreover, the user can choose to generate different outputs consisting only of the selected code and documentation blocks.

TrollEdit is an aspiring *OpenSource* project with the specific aim to create a multi-platform editor based on the *Qt* toolkit. It is designed with extensibility and flexibility in mind, for example new languages can be added at any time by providing their *LPeg* grammars. When combined with a debugger blocks can also hold various helpful data, for example block execution count. Potential visual effects in code blocks are not limited to the standard syntax highlighting as more visual feedback can be achieved e.g. colour coding algorithm cycles according to depth or complexity.

References

[1] Ierusalimschy, R.: A text pattern-matching tool based on Parsing Expression Grammars. *Softw. Pract. Exper.*, 2009, vol. 39, no. 3, pp. 221–258.

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