August 22, 2017

Abstract Syntax Tree-Based Code Editor For Mobile Phones

Zev Youra

ABSTRACT SYNTAX TREE-BASED CODE EDITOR FOR MOBILE PHONES

ABSTRACT

A system and a method are disclosed that enable abstract syntax tree-based (AST) code editor on mobile phones. The keys on the keyboard include nodes or small sub-trees of the AST of the code. To insert a new syntax, the keyboard is tapped. The system converts the inserted code into an editable format depending on nature of the code. The syntax includes all the necessary structural parts of the code. To allow the users to enter new code, the system includes placeholders which are empty nodes. The system automatically detects the keys used in the program and the keys used previously and provide the user necessary keys, variables and every language feature used in the programming language. This system and method prevents the user from making syntax errors.

BACKGROUND

Traditionally, code editors treat the program to be edited as a long string of characters. The user types one character at a time, selects one character at a time, and so on. Writing code on a mobile device is difficult due to the limitations of a small touchscreen.

DESCRIPTION

A system and method are disclosed that enable abstract syntax tree-based code editor on mobile phones. The system includes a keyboard with keys representing a node or small sub-tree of the AST (Abstract Syntax Tree). To insert a new syntax into a code structure, the particular key is tapped, which inserts a new node at the location. After inserting a new node, the user's selection is converted to an editable format. The editable format may vary based on the kind of code structure (e.g. after inserting a function call, move selection into arg list, after inserting an if statement, move selection into the test clause).

The syntax is derived from the AST, containing the spacing, commas, semicolons, and other structural parts of the code. To avoid syntax errors, the user's input is modified to translate the intent of the key press into a logical AST, which may be more complex than the simple node replacement (e.g. inserting a comparison operator requires creating a new parent for the selected node rather than just replacing it).

The system enables native editing using a standard keyboard in two cases: editing string or number literals, and naming variables. However, the keyboard may not allow code editing.

The AST-based editor requires a different strategy for insertion of new code. In a program, there is no default "start" location. Placeholders are configured essentially as "empty nodes" on the AST where the user may insert new code. The user may tap between lines of code or to the right of existing code to add a new placeholder between existing lines (which they may then replace with new code). Functions automatically have placeholders inserted in their argument lists. Some language constructs may have optional parts, for example, the "else" clause of an "if" statement. Those are represented as a special tappable placeholder that inserts the clause on tapping the #Keyboard. The system automatically detects the keys used in the program and provides the necessary keys. There must be a key for every variable in the program, every function call, and every language feature in the programming language (if statements, for loops, etc.).

An alternative implementation may include a keyboard with two categories: "auto" or default and "full key set". The "auto" category may contain a set of keys that fit on one "page" that is most likely to be used by the user based on the program and current selection of the user. The system may in the long term, take user behavior into account while including keys in the “auto” keyboard. This is somewhat similar to auto-complete in a typical editor. The "full key set" area may contain everything the user may possibly use, but organized by tabbed category. The combination of auto and tabbed categories may allow most users to work in one tap (on the auto tab) and for more advanced users to quickly get to any key they require in two taps (tab then key).

The user may not select parts of their program character-by-character, but rather can only select a token or group of tokens. The system and method of editing abstract syntax tree-based code on mobile phones prevents the user from making syntax errors.