ToyScript
A computer language for teaching coding in the K-12 system

Technical Report
LCC ITI 20-1

September 28, 2020

Francisco Vico
School of Computer Science
University of Malaga

Description

There exist a plethora of computer languages nowadays, arranged according to programming paradigms (declarative, imperative, data-driven...). They serve different purposes, some of which are better fitted for science and engineering, others for web technologies, or even devoted to very concrete applications. Very few were designed with an educational target in mind. Among them, Logo was one of the first and most popular ones, and others have been adopted in universities for illustrating concrete paradigms (like Eiffel for functional programming, PROLOG for logical programming, or Pascal for structured programming). In recent years, block-based visual programming languages (like Scratch or the one used by code.org) have emerged in web-based platforms, as an alternative for teaching to the more traditional text-based languages. Apart from their graphical and colorful nature, what makes them attractive for children and educational environment, there is much discussion on how to move from these dND (drag-and-drop) languages to text-based, and on the bad habits that they instil in users (like lack of debugging actions).

ToyScript is an imperative text-based programming language that is intuitive for children (since it resembles natural language, as opposed to OOP), and integrates knowledge acquired in the classroom. The main purpose is to provide a dynamically typed language, which can be taught in a gradual way, from simple computational concepts to more complex ones.

ToyScript 1.0 syntax is described below in EBNF notation.

```
<letter>      ::= "a" | ... | "z" | "A" | ... | "Z"
<digit>       ::= "0" | ... | "9"
<arithmetic>  ::= "+" | "-" | "*" | "/"
<relational>  ::= "==" | "!=" | "<" | ">" | "<=" | ">="
<bitwise>     ::= "&" | "|"
<negation>    ::= "!"

<natural>     ::= <digit> { <digit> }
<variable>    ::= <letter> { <letter> | <digit> }
<term>        ::= <natural> | <variable>

<expression>  ::= <term> | "(" <expression> ")" |
                  <expression> <arithmetic> <expression>
<condition>   ::= <term> <relational> <term> | <term> <bitwise> <term> |
                  <negation> <term> | "(" <term> ")"

<code>        ::= <instruction> { <instruction> }
<instruction> ::= "up" | "down" | "left" | "right" | "info" |
                  <assignment> | <repeat> | <if> | <while>
<assignment>  ::= <variable> "=" <expression>
<repeat>      ::= "repeat" <natural> <code> "end"
<if>          ::= "if" <condition> <code> [ "else" <code> ] "end"
<while>       ::= "do" <code> "while" <condition>
```

Discussion

ToyScript's syntax imports control flow structures from mainstream imperative languages ('if-else' conditional, and condition-controlled loop 'do-while'). Some instructions are specific, like 'info' (a statement for data input) and the movement commands. The 'repeat' statement is introduced in order to provide a count-controlled loop structure that does not rely on variables (in such a way, repeating an instruction or blocks of instructions can be taught before variables are introduced). Code blocks are delimited with the 'end' label, inspired by dynamic languages like Matlab or Julia (instead of curly brackets, which are more difficult to type for children). Finally, relational and negation operators are included. Bitwise 'and' and 'or' are preferred to the logical '&&' and '||' also to easy typing.

The resulting language is a tradeoff between a simplified type-free subset of JavaScript, where some elements —like curly brackets-delimited blocks, not so intuitive from a natural language perspective, or the 'for loop', efficient, but also cryptic— have been replaced by simpler structures ('end' and 'repeat'). And specific commands have been added to the core syntax, in order to provide an interface with a simulated 2D world, where an agent interacts with objects and other agents. While ToyScript code is intended to solve tasks in such a setup, future versions are intended to promote it towards a general purpose language, incorporating elements from the main programming paradigms.

An implementation of ToyScript is available in the online platform ToolboX.Academy, where it is used to command a robot to solve problems that range from simple movements to complex loops and conditions.