МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2026
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2026
Proceedings of the Fifty-Fifth Spring Conference
of the Union of Bulgarian Mathematicians
Tryavna, Bulgaria, April 5–9, 2026

BIDIRECTIONAL TRANSLATION BETWEEN TEXT-BASED AND BLOCK-BASED PROGRAMMING LANGUAGES IN THE LOGO TRADITION

Hristo Hristov
Institute of Mathematics and Informatics at the Bulgarian Academy of Sciences,
Sofia, Bulgaria
e-mail: h.hristov@posteo.net

This article presents a free and open-source, minimal implementation of bidirectional translation between text and blocks for a subset of Berkeley Logo, designed for programming turtle robots in mathematics and informatics explorations. The robots are intended to be assembled by students using off-the-shelf electronic components. The programming environment is web-based and can also be used offline. It supports switching between two editing modes – blocks and text – and four turtle-geometry commands: FORWARD and BACK (for relative translation), LEFT and RIGHT (for relative rotation). Its purpose is to provide a foundation for more advanced implementations that build on the blocks–text translation and can also be applied to projects beyond turtle geometry. Blocks mode is easier for beginners, particularly on touchscreen devices, while text mode offers greater flexibility. Switching between modes allows students to work in the way best suited to their skill level and device.

Keywords: turtle, robot, programming, logo, geometry

ДВУПОСОЧЕН ПРЕВОД МЕЖДУ ТЕКСТОВО-БАЗИРАНИ И БЛОКОВО-БАЗИРАНИ ЕЗИЦИ ЗА ПРОГРАМИРАНЕ В ТРАДИЦИЯТА НА LOGO

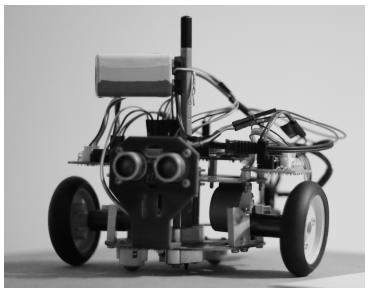
Христо Христов
Институт по математика и информатика, Българска академия на науките, София
e-mail: h.hristov@posteo.net

Тази статия представя минимална имплементация на двупосочен превод между текст и блокове за подмножество на Berkeley Logo, предназначена за програмиране на костенурки-роботи при изследвания по математика и информатика. Тя е свободен софтуер с отворен код. Роботите са предназначени да бъдат сглобявани от ученици с помощта на готови електронни компоненти. Програмната среда

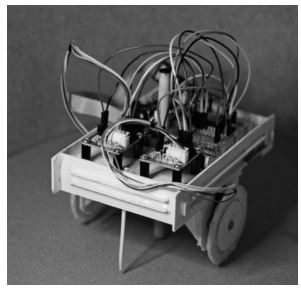
https://doi.org/10.55630/mem.2026.55.117-125
2020 Mathematics Subject Classification: 51-04, 68-04, 97-03, 97P40.

е уеб-базирана и може да се използва офлайн. Тя позволява превключване между два режима на редактиране, блоков и текстов, и поддържа четири команди в костенурковата геометрия: FORWARD и BACK (за относително преместване), LEFT и RIGHT (за относително завъртане). Целта ѝ е да послужи като основа на по-усъвършенствани реализации, които надграждат двупосочния превод между текстово-базирани и блоково-базирани езици за програмиране и могат да се използват и за проекти извън костенурковата геометрия. Блоковият режим е по-лесен за начинаещи, особено на устройства със сензорен екран, докато текстовият режим предлага по-голяма гъвкавост. Превключването между режимите позволява на учениците да работят по най-подходящия за тяхното ниво и устройство начин.

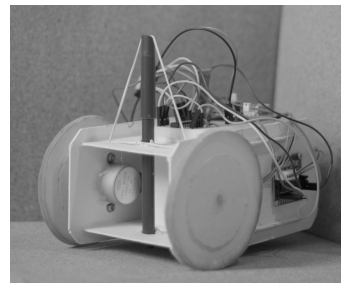
Ключови думи: костенурка, робот, програмиране, лого, геометрия



(a) Leo v1



(b) Leo v2



(c) Leo v3

Figure 1: Handmade turtle-robots

1 Introduction

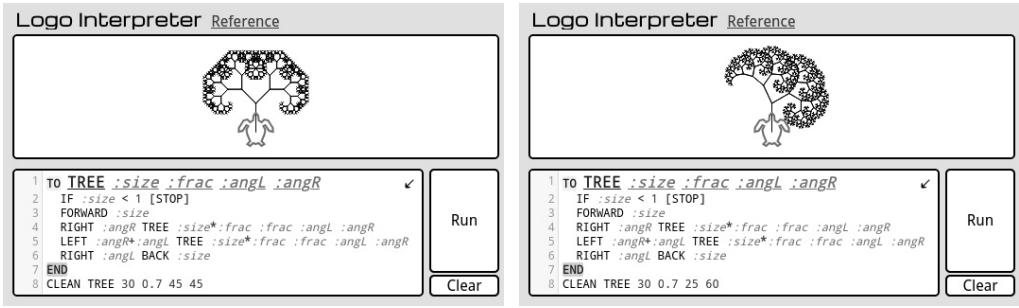
1.1 Constructionism, Logo and turtle-robots

Back in 1971, when computers were not yet widely available at schools and homes, Seymour Papert and Synthia Solomon wrote the article "Twenty Things To Do With A Computer"[5], envisioning computers being programmed by students, not vice versa. This article's 50th anniversary has been celebrated with a special collection of articles describing good educational practices in various countries, including Bulgaria [7].

There are two possible directions of the programming interaction between a child and a computer [5, p.1], leading to two approaches to education – "Constructionist approach" and "Instructionist approach". The "constructionist" approach led to the development of Logo, which became not only a programming language, but also a computer environment for explorations, philosophy and even a culture [3].

Papert characterizes the turtle as an "object to think with" [4, p.9]. Accordingly, he employed robots that can trace any geometric shape using just two commands: "FORWARD" and "RIGHT". And he called these robots *turtles ... in honor of a famous species of cybernetic animal made by Grey Walter, an english neurophysiologist.*[5, p.3]. Figure 2 presents variations of tree fractals using recursion, a conditional statement and the two

commands FORWARD and RIGHT, plus two additional commands BACK and LEFT (to avoid negative numbers as inputs).



(a) 0.7, 45, 45 (b) 0.7, 25, 60

Figure 2: Tree fractals drawn in the jslogo[10] interpreter with different parameters

1.2 Block-based and text-based

Block-based programming environments are specifically designed to help novices tinker and explore to learn programming. By using blocks over text-based languages, these environments reduce the learner's cognitive demands by limiting possible syntactic and grammatical mistakes and focusing only on developing algorithmic thinking [1, p.1]. Wrapping blocks with another blocks makes the program's structure clear and easy to understand. Blocks can also be labeled with pictures, instead of text, which allows children to learn programming before they learn reading and writing. Especially before they learn typing on a keyboard. Blocks are also easier to use on a smartphone or tablet, because these devices have a touchscreen, but lack a keyboard. Dragging blocks on such hardware is generally more accessible than typing characters.

Text-based languages require reading and typing on a keyboard (even if it is a virtual one, displayed on the screen). Young children may not yet have sufficiently developed these skills to use them effectively in the acquisition of another skill, such as programming. Consequently, the learning curve may become steep, which can lead to discouragement for some learners, particularly those who already have limited confidence in their abilities in other domains. Developing proficiency in programming can provide a foundation for learners to build confidence in their ability to acquire other skills, particularly when the learning experience is engaging and meaningful. This sense of self-efficacy may transfer to other areas, including problem-solving, communication, and emotional intelligence. Therefore, minimizing unnecessary barriers to programming can be an effective strategy in math and computer science education.

Once initial barriers are removed, overly "user-friendly" interfaces can limit deeper understanding, particularly for older students skilled in typing or motivated to master it. In text mode, copying, pasting, and multi-line editing are faster than with blocks, and experienced users can navigate code efficiently in editors like Vim[1] or GNU Emacs[2],

[1] `https://www.vim.org/`
[2] `https://www.gnu.org/software/emacs/`

with the entire source visible unless sections are folded or separated.

Text-based programs can be exported simply by copying the source code, while blocks often require a specific file format or multiple screenshots for large programs. Syntax in text languages can be complex, with structure determined by brackets, quotes, or indentation, which vary across languages. Blocks make these structures more visible, supporting debugging and comprehension. Each approach has strengths and limitations, and switching between them allows learners to focus on the program itself rather than the mechanics of programming.

1.3 The design for the turtle-robot

Papert and Solomon's vision of turtles being affordable for every child [5, p.3] is now achievable and enhanced. Recent studies evidence that robotics in education improves cognitive skills, fosters interdisciplinary learning, and bridges the gap between theory and practice [8, p.1]. In 2025, we developed Leo v2[3] (Figure 1b), designed for students to assemble their own turtle robot at minimal cost while retaining essential turtle-geometry functionality. Using off-the-shelf components, the robot can be programmed offline via a free, open-source compiler compatible with GNU+Linux, OpenBSD, macOS, or Windows, making hands-on exploration of geometric and algorithmic concepts accessible to all learners.

The article "Turtle Geometry with Arduino and Snap! – Developing Interactive Tools for Learning and Experimentation in Mathematics and Informatics" [2] guides students, parents, and teachers through acquiring widely available electronic components, cutting the robot body from cardboard, assembling it, and programming its microcontroller. Schematics, CAD drawings, and 3D models[4] are provided under the GPL v3 license, enabling users to build upon and share their own designs. CAD files also allow batch production of the robot body from cardboard, wood, or metal, using laser cutters, making it feasible for schools, municipalities, or governments to supply all students, though individuals can build their own with minimal support. A student who assembles a personal robot can provide peers with both inspiration and guidance, encouraging the creation of modified, personalized versions. As Papert and Solomon noted in 1971, turtles can have "sense organs". From simple touch, light, and sound sensors to advanced accelerometers and tilt detectors, exemplifying a low floor, high ceiling approach to learning [5, p.3].

1.4 The Previous Step: Blocks → Text Translator

The Leo v2 design [2] includes a translator from the block-based Snap![5] to text-based Arduino[6], enabling its ATmega328P[7] microcontroller to be programmed in both languages. This translator was developed to ease the learning curve for students familiar with block-based programming but less confident with text. The present work extends this effort by implementing bidirectional translation between blocks and text within a small subset of Berkeley Logo[8].

[3] https://hristov-imi-tezo.gitlab.io/vtu-ndmi-2024/

[4] https://gitlab.com/hristov-imi-tezo/vtu-ndmi-2024

[5] https://snap.berkeley.edu/

[6] https://docs.arduino.cc/language-reference/

[7] https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[8] https://people.eecs.berkeley.edu/~bh/v2ch14/manual.html

1.5 Evolution of Turtle Programming

Text-based languages are historically the primary method for programming turtles, originating when the first Logo language was created for text-driven computers. Modern interpreters, such as UCBLogo[9] (offline) and jslogo[10] (online), support virtual turtle programming. Block-based languages, widely used in education, include Snap![5] and Scratch[11] [6, p.31]. Snap! also supports microcontroller programming via the S4A Connector library[12], making it suitable for physical turtle robots. This evolution illustrates how combining blocks and text enables learners to progress from low-floor, high-ceiling visual programming to more advanced text-based programming.

1.6 Current goal

Bidirectional translation between text and blocks has also been implemented in CoffeeScript to program a virtual turtle in Pencil Code[13]. The Droplet library[14], used there, is employed for the same purpose in the implementation provided with this article (Appendix). The current work presents a minimal implementation of bidirectional translation for a small subset of Berkeley Logo, supporting only FORWARD and RIGHT. It serves as a foundation for further developments, including sending programs to a physical turtle robot such as Leo v2; an example of such a more developed environment is presented in Further developments.

2 Materials and methods

The provided implementation for bidirectional translation (Appendix) has two dependencies:

1. Droplet[14] – the JavaScript library used in Pencil Code[13]
2. Ace[15] – a JavaScript text editor library, used by Droplet for the text mode

If they are downloaded in advance, the programming environment can be used offline. The needed files are:

- **ace.js**[16] – the Ace text editor
- **mode-coffee.js**[17] – CoffeeScript mode for Ace
- **worker-coffee.js**[18] – CoffeeScript worker for Ace
- **theme-chrome.js**[19] – the default theme for Ace
- **droplet-full.js**[20] – the Droplet editor

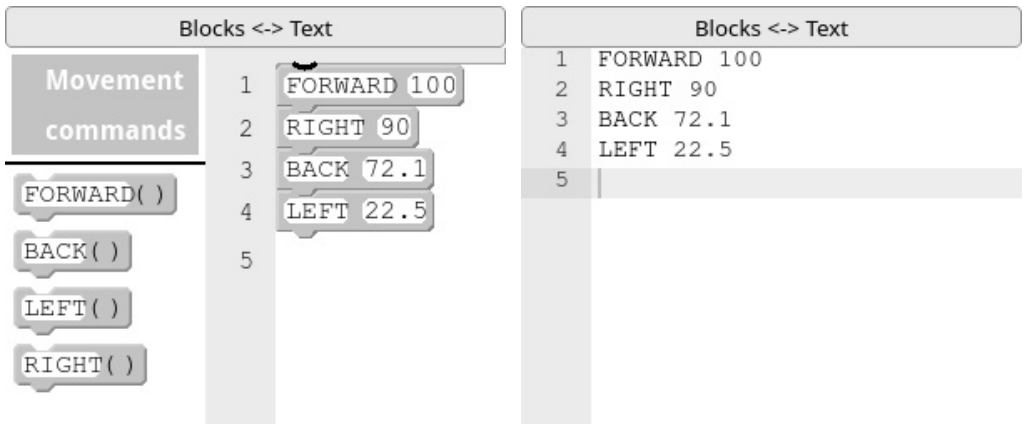
[9] https://people.eecs.berkeley.edu/~bh/logo.html
[10] https://www.calormen.com/jslogo/
[11] https://scratch.mit.edu/
[12] https://snap.creativelearninglab.click/course/view.php?id=2
[13] https://gym.pencilcode.net/draw/
[14] https://droplet-editor.github.io/
[15] https://ace.c9.io/
[16] https://cdn.jsdelivr.net/npm/ace-builds@1.43.5/src-noconflict/ace.js
[17] https://cdn.jsdelivr.net/npm/ace-builds@1.43.5/src-noconflict/mode-coffee.js
[18] https://cdn.jsdelivr.net/npm/ace-builds@1.43.5/src-noconflict/worker-coffee.js
[19] https://cdn.jsdelivr.net/npm/ace-builds@1.43.5/src-noconflict/theme-chrome.js
[20] https://droplet-editor.github.io/assets/droplet-full.js

- **droplet.css**[21] – stylesheet for Droplet

They can also be found in the repository of this article[22].

3 Results

The implementation for bidirectional translation between text and blocks provided with this article (Appendix) can serve as a foundation and a minimal working example. It contains a palette with four blocks – FORWARD, BACK, LEFT and RIGHT (Figure 3a). When dragged to the blocks editor, they receive a default value and become FORWARD 100, BACK 100, LEFT 90 and RIGHT 90. Only blocks attached to the top-left corner are translated to text when the "Blocks ↔ Text" button is clicked (Figure 3b). Clicking again restores the blocks editing mode, translating the text back to blocks and reintegrating any detached blocks.



(a) Blocks mode (b) Text mode

Figure 3: Basic user interface (Appendix)

4 Discussion

The minimal implementation presented in the Results section and Appendix is not sufficient for exploring geometric and algorithmic concepts using turtle geometry, as it lacks both a virtual turtle and a connection to a physical one. Its purpose is to provide a foundation for more developed implementations that build on the blocks–text translation and can also be applied to projects unrelated to turtle geometry. Blocks mode is easier for beginners, especially on touchscreen devices, while text mode allows greater flexibility. Switching between modes lets students work in the way best suited to their skill level and device. UTF-8 icon support enables programs to use icon-based syntax in both modes. Blocks mode also aids debugging by clarifying program structure, particularly when the code is complex or poorly formatted.

[21] `https://cdn.jsdelivr.net/npm/droplet-editor@0.0.2/css/droplet.css`
[22] `https://gitlab.com/hristov-imi-tezo/smb-2026`

Using Droplet or any other library which can work offline allows independence from internet or server connections, ensuring long-term accessibility: future users can explore geometric and algorithmic concepts with these robots under the same conditions, even as technology evolves, assuming electricity is available.

5 Further developments

The provided implementation is minimal, serving as a foundation for further development. It has been extended[23] to program physical turtle robots such as Leo v2, with source code and the microcontroller program shared in the Git repository of this article[24] under GPL v3.

Future work includes supporting a larger subset of Berkeley Logo — procedures, recursion, loops, and conditionals — so students can test programs virtually with UCBLogo before using a robot. Hardware support for PENUP/PENDOWN would allow movement without leaving a trace, and translation between different text languages would enable use of Scheme, Python, JavaScript, Arduino, C++, C, or assembly depending on task or learning goals.

6 Acknowledgments

My deepest gratitude to my mentor Evgenia Sendova for inspiring me to explore and teach mathematics through turtle geometry.

Appendix

index.html[25]

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="lib/droplet/droplet.css"/>
    <style>
      #toggleBtn {
        position: absolute; left: 0; right: 0; top: 0; height: 25px;
      }
      #droplet-editor-wrapper {
        position: absolute; left: 0; right: 0; top: 25px; bottom: 0;
      }
    </style>
  </head>
  <body>
    <button id="toggleBtn" onclick="editor.toggleBlocks()">Blocks <-> Text<
    /button>
    <div id="droplet-editor-wrapper">
      <div id="droplet-editor"></div>
    </div>
    <script src="lib/ace/ace.js"></script>
    <script src="lib/droplet/droplet-full.js">
```

[23] https://hristov-imi-tezo.gitlab.io/smb-2026/ui
[24] https://gitlab.com/hristov-imi-tezo/smb-2026
[25] https://hristov-imi-tezo.gitlab.io/smb-2026/assets/src/index.html

```
    </script>
    <script>
      const movementBlocksArr = [
        { block: "FORWARD()",          // forward movement block
          expansion: "FORWARD 100" }, // default value: 100 mm
        { block: "BACK()",             // backward movement block
          expansion: "BACK 100" },    // default value: 100 mm
        { block: "LEFT()",             // left rotation block
          expansion: "LEFT 90" },     // default value: 90 degrees
        { block: "RIGHT()",            // right rotation block
          expansion: "RIGHT 90" }     // default value: 90 degrees
      ];
      const dropletEditorOptions = {
        // CoffeScript is the selected language for parsing,
        // because it works better with the Logo language syntax
        // than the other languages currently supported by the Droplet
    library
        // (JavaScript, CoffeeScript, HTML, C and partially Python, Java,
    BASIC)
        mode: "coffee",
        palette: [{
          name: "Movement commands",
          color: "blue",
          blocks: movementBlocksArr
        }]};
      const editor = new droplet.Editor(
        document.getElementById('droplet-editor'),
        dropletEditorOptions
      );
    </script>
  </body>
</html>
```

References

[1] Dong, Yihuan and Marwan, Samiha and Catete, Veronica and Price, Thomas and Barnes, Tiffany (2019) Defining Tinkering Behavior in Open-ended Block-based Programming Assignments, Association for Computing Machinery, SIGCSE '19, pp. 1204–1210, https://doi.org/10.1145/3287324.3287437

[2] Hristov, H. (2025) Turtle Geometry with Arduino and Snap! – Developing Interactive Tools for Learning and Experimentation in Mathematics and Informatics, Mathematics, Computer Science and Education, Volume 8 Issue 1, pp. 48-68, https://journals.uni-vt.bg/mcse/eng/vol8/iss1/art7

[3] Papert, Seymour A. (1999) What is Logo? And Who Needs It?. Logo philosophy and implementation. Logo Computer Systems Inc., https://dailypapert.com/what-is-logo-and-who-needs-it/

[4] Papert, Seymour A. (1980) Mindstorms: children, computers, and powerful ideas. ISBN:978-0-465-04627-0, https://www.media.mit.edu/publications/mindstorms/

[5] Papert, Seymour A. and Solomon, Cynthia (1971) Twenty Things To Do With A Computer https://doi.org/1721.1/5836

[6] Perin, Ana Paula Juliana and dos S. Silva, Deivid Eive and Natasha

M. C. Valentim (2023) Investigating block programming tools in high school to support Education 4.0: A Systematic Mapping Study, Informatics in Education, volume 22, issue 3, pp. 463-498, https://doi.org/10.15388/infedu.2023.21

[7] Sendova, Evgenia (2021) An Eternal Source of Inspiration or What the Bulgarian Turtle Told Achilles This Time in Gary Stager, Twenty Things to Do with a Computer Forward 50, Constructing Modern Knowledge Press, 63-92, ISBN 978-1-955604-01-7

[8] Zhang, Deyu and Wang Jiawen and Jing Yanbin and Shen Ao (2024) The impact of robotics on STEM education: Facilitating cognitive and interdisciplinary advancements, Applied and Computational Engineering, volume 69, pp. 7-12, https://doi.org/10.54254/2755-2721/69/20241433