Physical Programming Language

Akshay Baweja

Abstract— This paper explains the physical programming language that helps its user understand the fundamentals and concepts used in a programming language. It uses LEGO-style blocks and colors to help kids learn the basics of programming and aims to increase the learning effectiveness of programmers. The Physical Programming Language takes the form of a kit that comprises different sub-modules like variable blocks, value blocks, marker blocks, display blocks, loop blocks, conditionals blocks, and operator blocks. The physical programming language aims to provide a tangible and tactile learning in a physical environment. It also re-imagines how we think of programming language today. Instead of a traditional programming setup that involves a minimum of a keyboard and a screen to function, it uses physical blocks. It helps the user visualize how a program would look in an actual physical space than inside a storage disk.

Index Terms— Programming; Playful Learning; Education; Flowcharts; Logic Development; Experiential Learning; Code Blocks;

1. INTRODUCTION

As we continue to innovate, technology is set to advance even further and we will continue to include more advanced devices as part of our daily lives. Also, we are incorporating more complex programs and structures which ease our way of living. As we imagine programming to be, sitting in front of the screen, typing out lengthy code files, and uploading them to servers. But, **can we visualize a program without screens?** An interview with twelve people from different professional backgrounds revealed that seven out of twelve left programming during their initial hands-on coding experiences. At the same time, two of the remaining five said it took them time and hours of tutorials to understand the programming language they were trying to learn. While the remaining three said, they were able to grasp the coding concepts in their initial learning stages. This raises a question of **can we make learning code a playful experience?**

1.1 Why is learning programming important?

Learning how to program not only helps kids learn to code but also provides them with various other advantages. These benefits may include improvement in their problem-solving ability, along with increased thinking ability. Programming gives children a challenge and helps them develop resilience. As Margaret Mead said: "*Children must be taught how to think, not what to think.*" [8] Learning how to program indeed teaches them how to think instead of what to think, which is the most critical part of learning as it helps grow our intellect and understanding of various systems and concepts.

Research into learning methodologies [2], [11] that helps children learn and grasp concepts showed that the following methods show promising results –

- Learning through poetic narrations – Poems have a certain rhythm, which attracts attention and retains in memory.
- Learning through games – a thoughtfully designed game attracts attention because of sounds, contrasting colors, and visually appealing graphics. Thus, a game can help teach one or more concepts in a better way.
- Learning through playful experiences – a playful experience acts as an aid to learning as it helps a kid learn certain things they are experiencing in a physical world, which helps them understand better as they are interacting and experimenting with it [10].
- Learning through storytelling – Stories are something we all love to read or hear. Stories invoke a sense of curiosity and forces to understand and know what's coming ahead. This rising curiosity facilitates learning as it helps in thinking of multiple possibilities around a thing, which in turn boosts their thinking ability.

- *Akshay Baweja is currently pursuing Master of Fine Arts degree program in Design and Technology from Parsons School of Design, The New School, New York, United States of America, PH - +1 (347) 614-7472. E-mail: akshaybaweja@newschool.edu*

1.2 Initial Idea and Concept

The initial idea and concepts revolved around the question of **how can learning to code be made fun and easy for children to step into the world of code?** The notion of making code easy to understand led to the drafting of initial goals that must be set to achieve the desired learning experience. These goals are –

- It should be easy to understand.
- It should include at least basic programming concepts.
- It should have elements like variables, operators, expressions, loops, inputs, and outputs.
- It should be self-explanatory yet straightforward.
- It should be visually appealing to kids.

Taking LEGO Blocks as an inspiration, the initial concept of *Physical Programming Language* is drafted. It uses colorful blocks to create a playful experience. Each block denotes a code element. These elements when put together create a piece of code. The blocks are conceptualized to be colorful and textured, creating a unique identity of each code element. The initial and last block would be 'BEGIN' and 'END' block, respectively. The 'BEGIN' block consists of a small screen that will be used to print output in case of an error free code otherwise will be used to log errors, whereas the 'END' block will mark the end of the program.

2. System Features

The system design provides a playful and tangible learning experience using visually appealing code blocks to help children understand the basic programming concepts. An ideal audience for the Physical Programming Language would be children beginning to learn to code. It can be used by a single user or multiple users to collaborate and make a program. This also teaches the importance of *working collaboratively* with fellow partners [4].

2.1 Basic Structure of Physical Programming Language

Fig 1. Variable Block for Physical Programming Language

2.1.1 Variables

Unlike other programming languages, physical programming language uses different colors to represent variables. Each unique colored block represents a unique variable (as shown in figure 1). The block looks like a LEGO block with the option to combine different blocks from all four sides. The circular openings and indents help blocks to fit in together with one another.

2.1.2 Values and Operators

Values and Operators in the physical programming language are represented by translucent amber-colored blocks having the value or operator embossed with white color on the block's surface. The distinctive amber color helps in visually separating the value and operator blocks from other blocks. (refer to figure 2).

The *value blocks* are available from 0 to 9 while *operator blocks* include addition (+), subtraction (-), multiplication (×), division (÷), exponential (^), modulus (%), greater than (>), less than (<), equality (=), inequality (,), logical AND (&) and logical OR(|).

It would be interesting to note that the use of greater than, less than, equality, inequality, logical AND, and, logical OR is limited to conditionals only. For equating two systems, the '*equal to*' sign is assumed between the first block and the second block of a line.

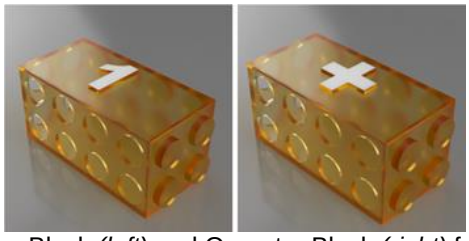

Fig 2. Value Block *(left)* and Operator Block *(right)* for Physical Programming Language. The value block denotes a value of '1' while operator blocks showcases 'addition (+)'.

The two or more value blocks are put together horizontally, i.e., either on the left or on the right, they auto-adjust their decimal position. Suppose we have a value block with a value of 7 and put a value block with a value of 9 on the right side of block 7, then the total value becomes 79.
Both the value and operator blocks can only have valid attachments on their side, i.e., either left or right. They do not accept any parameter on their top or bottom.

2.1.3 Loops

In Physical Programming Language, loops are divided into two sections, marking the beginning and end of the 'LOOP' block. The LOOP BEGIN block marks the start of the loop to indicate the number of times loop is required to run that is denoted by adding a *value block* on right side of the LOOP BEGIN block as shown in figure 3 while the LOOP END block marks the end of sub-program which needs to be iterated by the loop. Figure 4 shows the LOOP END block is marked only 'LOOP' while the LOOP BEGIN block had an additional marking of 'TIMES.' This marking denotes the requirement of an additional value block. The Loop End Block is also closed on sides. Hence, making the two visually distinctive.

2.1.4 Conditionals

Conditionals in the physical programming language are very similar to the concept of loops; it is divided into three sections, namely, 'IF,' 'ELSE,' and 'END IF.' The IF block, just like the loop block requires additional block(s) to function. Conditional operators (greater than, less than, equality, and inequality) are required to put up conditions needed for the IF block.
The ELSE block may or may not have additional parameters to function. The additional parameters, if added, must be done so by adding an IF block next to ELSE block.

2.1.5 Indentation

An essential part of coding is the formatting of code. Learning how to format code is as important as learning how to code. The blocks are designed to be capable of incorporating indentation in them. If we look at figure 3 closely, we will notice that the LOOP BEGIN block has connection port open only on the bottom right part of the block. Look at figure 7, and we will see how the loop block ensures the indentation of the code within. This imparts learners with a habit of indenting and formatting code as they evolve.

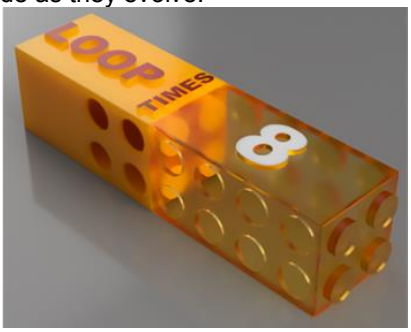

Fig 3. Loop Begin Block with a value block of value '8' on the right, indicating the number of times loop needs to be executed.

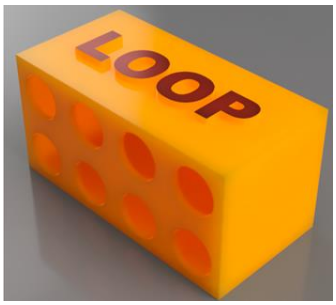

Fig 4. Loop End Block for Physical Programming Language

2.1.6 Learning Algorithms

Algorithms are the heart of coding, and a good coder can

formulate the algorithms irrespective of language and syntax. It is the logic that a coder can think and develop. Physical Programming Language enables users to understand and try building algorithms in a physical environment. The tangibility aids the learning process since users can physically interact with the code blocks in a hands-on experience. Figure 7 illustrates the *Fibonacci Series* that is built using the physical programming blocks. The vibrant colors help in retention so users can retain the code for a longer duration, and also the minimalist design approach reduces the complexity of code [7].

3 Learning System

The Physical Programming Language is a seamlessly integrated system that combines various fabrication techniques, electronics, and firmware development, enabling users to have a playful experience in learning programming.

3.1 Aesthetic Look and Feel

The physical form is built up using different colored acrylic sheets with a sheet thickness of 3mm. While the value and operator block use a matte, translucent acrylic, while other blocks use a glossy, opaque acrylic. Each block has a dimension of *120mm* × *60mm* × *60mm*. Each block consists of small disc neodymium magnets on circular cutouts and extrusions. These extrusions help the block form a correct connection with other blocks and prevent the wrong orientation of the block [6].

3.2 Hardware Implementation

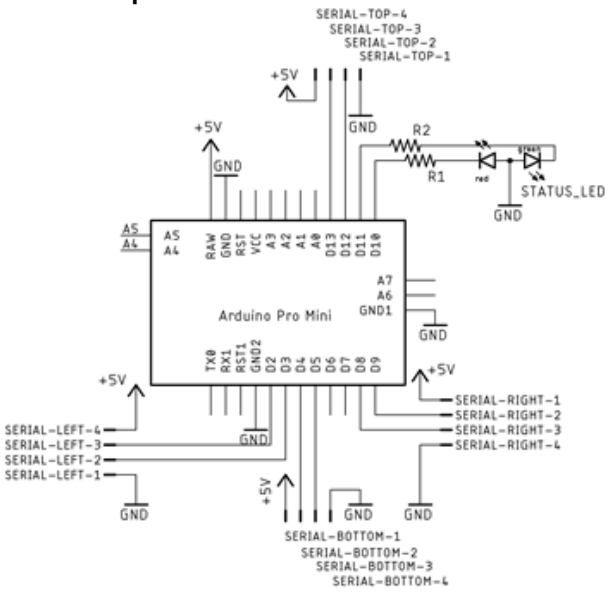

Fig 5. Schematic of Physical Programming Block

Each block in Physical Programming Language implements *serial communication* to communicate with adjacent blocks. Therefore, each block consists of four serial ports on four sides of the block, i.e., top, bottom, left, and right. Since neodymium magnets are used in forming connections with other blocks, they also conduct electricity. This makes them suitable for using the serial connection. For implementing the serial communication, an ATMega328 microcontroller, an 8-bit AVR RISC-based microcontroller having 32KB ISP flash memory, 1KB EEPROM, and 2KB SRAM [9], based custom printed circuit board (PCB) is used. The custom printed circuit board is designed around the Arduino Pro Mini [1] module, which embeds the ATMega328 microcontroller. Apart from Arduino Pro Mini, the PCB houses four 4-pin connectors on the four sides which connect to the neodymium magnets on the outer shell of the block and further connect to other blocks. Each 4-pin connector corresponds to VCC, GND, RX, and TX, as shown in the schematic in Figure 5.

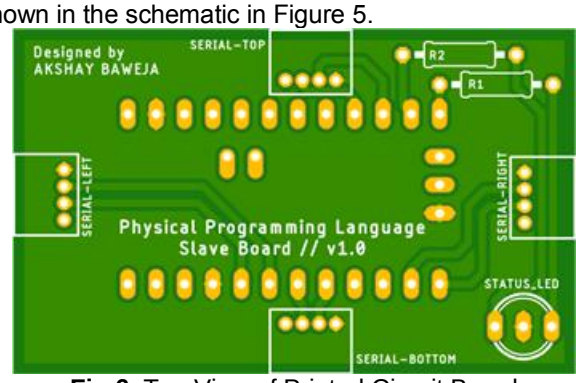

Fig 6. Top View of Printed Circuit Board

3.2.1 Display

The top block, i.e., 'BEGIN' block houses a *480px* × *320px*display, which displays the connected blocks along with the output. Moreover, the display also explains each code line and how the computer processes each line. The display is an IPS 3.5" display that is seamlessly integrated with the LEGO style programming blocks. The display provides the freedom to play around and experiment with the blocks. Hence, it plays a significant role in the process of learning to program.

3.3 Software Implementation

For software implementation, the *SoftwareSerial library* by Arduino, which is based upon the *NewSoftSerial* library by *Mikal Hart* [5], is used to create four software serial ports to communicate with other block modules. The 'BEGIN' block acts as the master block while the other blocks are considered slave blocks. The serial communications occur at a speed of 115200 bauds per second.

Each module is assigned an ASCII character, which they return whenever the master module requests them. These individual characters help the master module locate the row and position of each block. These positions are then formulated into code lines and processed individually. The Physical Programming Language is an interpreter-based language, i.e., the code is analyzed and executed line by line.

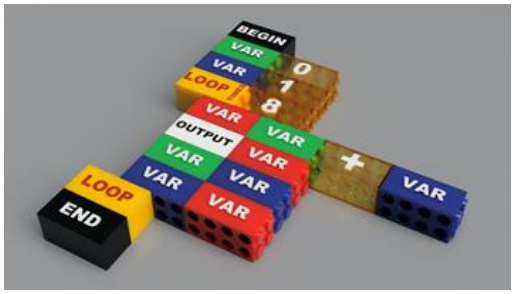

Fig 7. Fibonacci Series using Physical Programming Language

Once these individual characters form a string, the expression is evaluated using the *String Expression Evaluation Algorithm* [3].

6 Conclusion

I present, Physical Programming Language, a tangible programming language, a tangible system that helps beginners learn to program. Compared to other precedents and kits available, the physical programming language has following advantages -

- It bridges the gap between logic building and computer programming.
- It helps the user experience programming in a tangible and fun way.
- By adding different colors on blocks, I aim to make it easier for users to notice and differentiate between different blocks. It helps them better understand how each block functions at different positions.

The Physical Programming Language is still in its early stages and yet to be iterated and reiterated to make it more user-friendly and intuitive. Possible future work directions include experimenting with materials such as silicon and iterating over visual aesthetics like typeface, screen colors, etc. Also, changing the block dimensions will make production more affordable. Moreover, optimizing the printed circuit board design will reduce the serial transmission latency and increase the reliability.

My next step is to continue building the Physical Programming Language, run usability tests with potential users, evaluate the responses, and refine my design and implementation.

Acknowledgement

I would like to thank Kyle Li for helping me experiment and develop the Physical Programming Language. I would also like to thank my friends and family who helped in evaluating and iterating the project and giving their honest feedback. I am also immensely grateful to Nikita Sharma for reviewing the earlier version of this manuscript.

References

[1] Arduino. Pro Mini. (2014). https://store.arduino.cc/usa/arduino-pro-mini. 2020.

[2] Mark J. Brosnan. 1998. Spatial Ability in Children's Play with Lego Blocks. Perceptual and Motor Skills 87, 1(1998), 19–28. DOI: 10.2466/pms.1998.87.1.19

[3] Northern Illinois University Department of Computer Science. 2013. Using a Stack to Evaluate an Expression. (2013). http://faculty.cs.niu.edu/~hutchins/csci241/eval.htm. 2020.

[4] Min Fan, Alissa N. Antle, Maureen Hoskyn, Carman Neustaedter, and Emily S. Cramer. 2017. Why Tangibility Matters: A Design Case Study of At-Risk Children Learning to Read and Spell. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). Association for Computing Machinery, New York, NY, USA, 1805–1816. DOI: 10.1145/3025453.3026048

[5] Agus Kurniawan. 2015. Arduino Uno: A Hands-On Guide for Beginner. PE Press.

[6] Jui-Pin Lin. 2012. Pogo pin connector. (Dec.25 2012). https://patents.google.com/patent/US8337256B1. US Patent 8,337,256B1.

[7] Zohare Manna and Richard Waldinger. 1984. The Logical Basis for Computer Programming: Deductive Reasoning. (1984).

[8] Margaret Mead. 1977. Letters from the field (1 ed.). Harper & Row.

[9] Microchip. 2018. megaAVR® Data Sheet, ATmega48A/PA/88A/PA/168A/PA/328/P. (oct 2018). https://www.microchip.com/wwwproducts/en/ATmega328. 2020.

[10] Maria Moundridou and Alexander Kalinoglou. 2008. Using LEGO Mindstorms as an Instructional Aid in Technical and Vocational Secondary Education: Experiences from an Empirical Case Study, Vol. 5192. 312–321. DOI: 10.1007/978-3-540-87605-2_35

[11] Dora Simões, Margarida M. Pinheiro, and Claudia Amaral. 2014. How the use of Different Teaching and Learning Methodologies Echoes in Learners: The Students' Perspectives. TEM Journal 3, 3 (2014), 262–271.