Olympiads in Informatics, 2017, Vol. 11, 193–198

DOI: 10.15388/ioi.2017.17

Casual Programming: A Channel for Widespread Computational Education

Shaya ZARKESH
Polyup Inc., USA
email: shaya@polyup.com

1. Introduction

Over the last ten years, casual gaming has seen an impressive rise to the mass market. The smartphone industry currently poses the fastest-growing sector of videogames, with an incredible 23.7% year-over-year growth (Newzoo 2017). Smartphone games present the biggest opportunity to reach a mass market for casual gaming, so an increase in smartphone game popularity is indicative of a greater general shift towards casual gaming.

The question thus becomes the following: how do we harness the megatrend of casual gaming for good? How do we turn smartphone games into useful, educational tools?

At *Polyup*, we aim to answer these questions to empower a global community of creative problem solvers. Taking into account the wild popularity of casual gaming, we set out to find the best path to help primarily older children and teens, but also adults, with their "computational thinking" skills. Computational thinking lies at the heart of new topics becoming indispensable in the information age – subjects like data science, cryptography, informatics, and artificial intelligence. Computational thinking is a way of approaching and analyzing computational problems by honing the skills of pattern recognition, decomposition, abstraction, and algorithm design (Wing 06). In a sense, computational thinking is like critical thinking for STEM – whereas critical thinking focuses on finding relationships between ideas in written text, computational thinking is devoted to finding patterns in numerical contexts. Although the concept is immensely important to many STEM fields like mathematics and computer science, it often goes unaddressed directly in schools, and students are thus left without the ability and creativity to solve more difficult computational problems on their own. By creating an attractive casual game where users can lead their own learning, *Polyup* can help fill in the 21st-century skills that most of today's population lacks.

So, enough introduction. What is this mysterious environment that *Polyup* has created? In short, it is the world's first mobile, casual programming environment. While

block-based environments have been available on tablets and computers for years, there has yet to be a major environment optimized for mobile phones. Because of massively constrained screen real-estate, it seems a programming environment on a phone would require a major simplification and reduction of features. However, in the sections that follow, we detail how *Polyup's* gamified programming environment still retains all possible computational tasks and is thus Turing complete.

2. Learning by Teaching

One of the most effective and motivating ways to learn is by teaching others. The story behind *Polyup* capitalizes on just that – it revolves around teaching an AI sidekick named Poly. At first, Poly gloats about all his ability in the computational programming environment; he knows how to do everything from solving quadratic equations to proving Fermat's last theorem! However, Poly's hubris soon leads to his demise, as he attempts a program too computationally complex for his own good, and his internals break, leaving his memory banks empty. From here on, it is the player's job to (re)teach Poly all that he has forgotten, starting from basic tasks like adding numbers to more complicated ones like calculating factorials and building a GCD (greatest-common-denominator) algorithm.

Learning by Teaching is not a novel idea – it is embedded in modern pedagogy, especially given new opportunities posed by digital learning (Biswas 05). However, *Polyup* takes a novel approach to Learning by Teaching by combining it with dynamic scaffolding, the adjustment of difficulty mid-game based on player performance.

Polyup is a level-based system: the player has to solve puzzles by creating a program to achieve a stated output. The levels are of increasing difficulty, but no player will traverse every level. Levels are designed such that multiple levels teach a similar concept; when it is clear the user has mastered a programming concept, they move on to a more difficult one. Otherwise, if the user is unable to pass a level quickly and efficiently, it seems they have not mastered the concept, so further levels on the same concept are given. Such is the backbone of dynamic scaffolding. By implementing dynamic scaffolding in a level-based system, *Polyup* is adaptable to any skill level in programming. Furthermore, by using a non-classical programming paradigm, *Polyup* levels the playing field between experienced programmers familiar with programmatically syntax and those unfamiliar with any programming languages, thus making the educational experience universal.

3. The Environment

One of the most difficult aspects of creating a mobile programming environment is retaining functionality while simplifying the interface to manage a small amount of screen space available. In particular, typing on a phone is cumbersome and frustrating, so we

avoided typing altogether. Like many existing educational programming environments, we thus resorted to a gamified form of block-based programming. However, the similarities to existing environments end there: we have completely rethought the programming paradigm and implemented a novel, simpler user interface.

The first major deviation *Polyup* has taken from most existing programming environments is the use of a functional programming environment with Reverse Polish Notation (RPN). The functional programming part means that the environment evaluates expressions immediately instead of executing statements. RPN is a style of programming in which the operator follows the operands (e.g.: 5 2 +), instead of putting the operator between operands like in classical mathematics (e.g.: 5 + 2). Such a notation has a dual purpose: first, it removes the need for parentheses, thus greatly simplifying the programming experience, and second, it exposes the user to a new programming paradigm, making them think of operators as functions. Unlike the typical step-based environments, *Polyup*'s environment truly requires the player to think about the order of operations and how operators and operands will interact.

Polyup's user interface is also unlike most step-based programming languages in that it allows for manipulation of multiple functions on the same page. At the base level, the user interface of Polyscript involves dragging blocks onto a main stack of blocks, called "Poly," and any number of additional stacks. In Fig. 1 is a level in *Polyscript*; note the general layout of a puzzle with a target text on top, a function (stack) in the middle, and a set of draggable blocks available on the bottom.

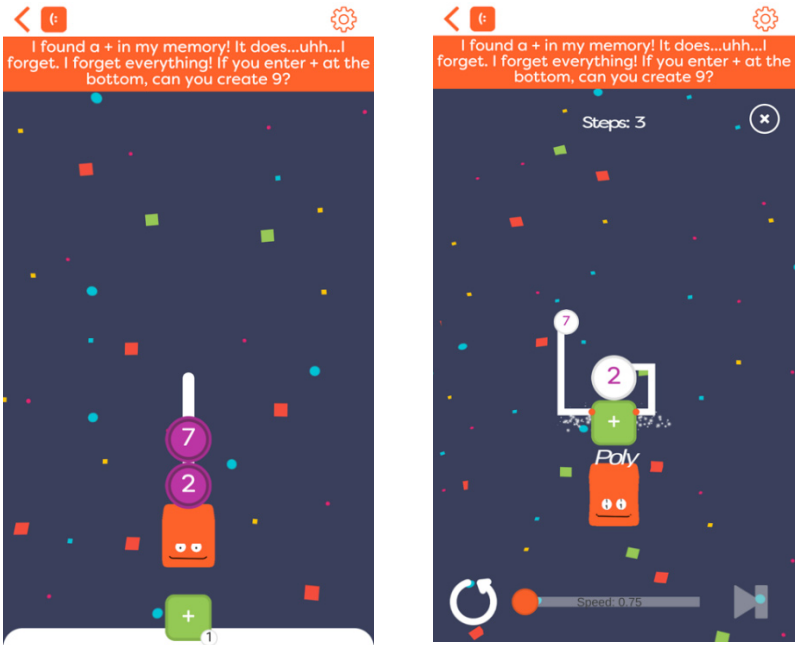


Fig. 1. (left) A simple level in Polyscript (right) the level running.

In Fig. 2 the custom workspace is shown; here, the user has full creative control over the program, with draggable blocks organized into panes.

The panes are laid out as follows:

Number Pane

The number pane contains a system for typing any number or decimal. It contains blocks for the digits 0 through 9, as well as a decimal point; the user can either tap a sequence of these numbers to make a longer number, or drag any block directly to the stack for a number from 0 through 9.

Math Pane

The math pane contains the four basic operations +, –, ×, ÷ as well as more advanced functions like sin and mod.

Variable Pane

The variable pane contains the variable names x, y, and z, as well as the set blocks ->x, ->y, ->z. A set block is an operator that takes in the value above it and sets the respective variable to that value. The variable blocks are operands that are simply references to the values contained in them.

Boolean Pane

The Boolean pane contains the Boolean values, True and False, and the inequality symbols >, <, ≥, and ≤. It also contains the "if", block, an operator that takes in a Boolean value and two functions: one for the "True" case and one for the "False" case.

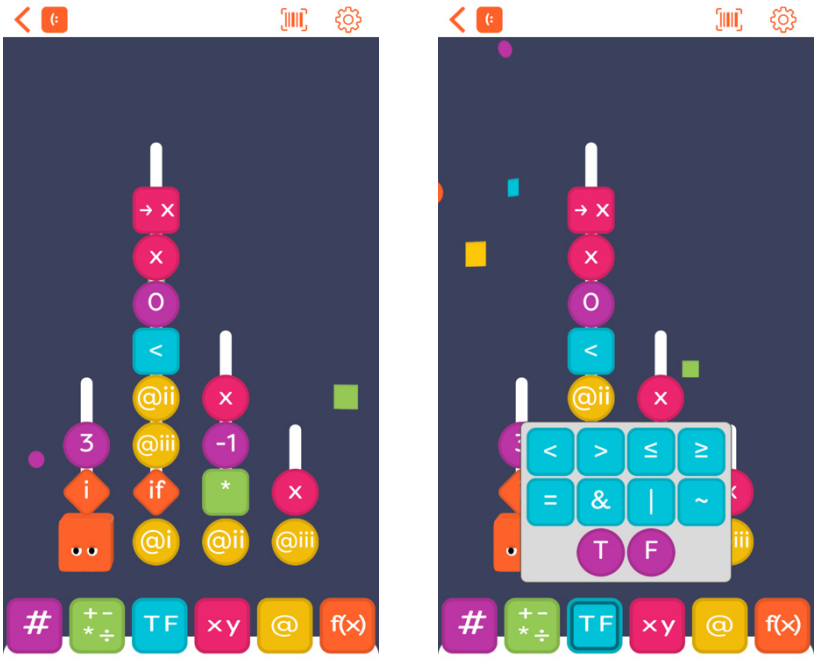


Fig. 2. A more complicated program written in Polyscript.

The magic of *Polyup* lies in its extremely simple compiler. A pointer reference starts at the top of Poly's stack and continually moves down. If it hits an operand, nothing occurs. If it hits an operator, the operator "eats" a number of blocks above it, corresponding to the number of values it takes in (e.g. the + operator takes in 2 values, so the 2 blocks above the "+" are eaten). If the pointer hits a reference to a "file", it adds the file to the stack and continues moving down. The program ends when the pointer hits the bottom of the stack.

4. Expanding the Scope: Socializing the Platform

Any gamified environment will have a hard time attracting users if it does not implement interaction with real humans, in order to motivate users to be the best at the game. *Polyup* thus implements multiple avenues for social interaction in-game. First, there is a puzzle challenge system where a player can create a level and challenge his or her friends to tackle it. Second, we have a live co-editing mode where two players can collaborate on the same puzzle workspace. Lastly, leaderboards for puzzle solving allow players to see their relative rank in problem solving ability, as determined by the dynamic scaffolding algorithm of the app. These avenues for social interaction can make the app far more attractive and create a viral effect, whereby players willingly share the app with friends in order to challenge or collaborate with them.

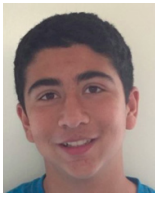
5. Results and Conclusions

Polyup is currently undergoing rapid prototyping of its application and has visited many middle and high schools to receive feedback, much of which has been implemented in its current conception. Responses are overwhelmingly positive. Of 39 students surveyed in the latest major version of the app, the average rating is 7.46. Of the 8 who came from an underprivileged high school, the average rating is 8.13! These results are staggering, and are indicative of the fact that *Polyup* has built an extremely attractive educational programming environment.

We have introduced and reviewed a novel educational programming environment set forth by *Polyup*. By gamifying the programming environment and translating programming to mobile, *Polyup* has created a more attractive and approachable environment to gain computational skills. By partnering with existing educational resources and competitions, *Polyup* has the potential to gain widespread use as a go-to computational learning application.

References

Biswas, G., Leelawong, K. (2005). Learning by teaching: a new agent paradigm for educational software. *Applied Artificial Intelligence*, 19(3), 363–392.
The Global Games Market Reaches $99.6 Billion in 2016, Mobile Generating 37%. Newzoo, Jan. 2017.
Wing, J.M. Computational thinking. (2006). *Communications of the ACM*, 9(3).



S. Zarkesh is a rising senior at the Harker high school in California's bay area. He is an avid coder and math student and has won many regional and national awards in these fields, including national champion at TSA TEAMS (Tests of Engineering Aptitude, Mathematics, and Science) 2016 and the Science Bowl Regional Champion of 2017. Shaya is a co-founder of *Polyup* Inc., a high-tech startup in Silicon Valley founded in 2015.