# HELPUS: How Entry Level Programmers Use Serialization

Matthew Friedel
MIDN 2/C, USN
*Electrical and Computer EngineeringN*
*U.S. Naval Academy*
Annapolis, Maryland
m212010@usna.edu

***Abstract*—Parallel programming is a powerful tool that can be used to increase performance in most applications, however the skillset required to program in parallel is unavailable to novice programmers. HELPUS is a visual programming language (VPL) that can be configured to directly represent parallel programming models, such as traditional locks in C++ or the Sofritas API. It is presented alongside a series of games that represent various parallel challenges faced in real-world applications. By working through the games using the VPL, novice programmers learn to recognize opportunities for adding parallelism in real-world applications.**

## I. Introduction

SOFRITAS is a parallel programming model that revolves around breaking serial code up into ordering free regions (OFR) that can operate atomically from each other. It is realized in an API that builds upon the memory locks that are present in regular C++ parallel programming. In the original study of SOFRITAS, the main validation was based around its performance and speedup of well-known benchmarks compared to other widely used parallel programming models, however there was no productivity study. The purpose of this project is to measure the relative productivity of SOFRITAS compared to traditional locks by designing a problem solving game for novice and advanced programmers.

### A. Performance vs Productivity

Performance and productivity are the two main measurements of the usefulness of a programming language or tool. Performance is a measure of the processing power that it takes to run a specific calculation, while productivity is a measure of the effort that it takes to create a specific calculation. Typically, the performance or productivity of a programming language is measured in relation to another widely used language. An example that highlights the difference between these is "Simulate a Sine wave over one period in C++ and in Matlab." The resulting C++ program would hands-down use less processing power than the Matlab program while still producing the same result, which indicates that C++ has a higher performance. The effort that it takes to create these two pieces of code, however, would be very different. Creating the C++ code would require time creating multiple loops, while the Matlab code would could be written in very few lines in a fraction of the time, which indicates that Matlab has a higher productivity. In order to measure the productivity of SOFRITAS vs traditional locks, we will have people solve the same series of problems in both of the programming models and compare the effort it took to create those solutions.

### B. Novice Programmers

Previous studies of parallel productivity involve studying post-grad students and programmers with extensive programming experience. This is because the majority of new parallel programming tools are aimed around increasing the performance and productivity of High Performance Computing (HPC) machines such as supercomputers. One of the main aims of SOFRITAS, on the other hand, is to make parallel programming more accessible to programmers at every level. The goal of this study is to measure productivity mainly using novice programmers who have already had some introduction to serial programming. This demographic has the most to gain by having access to a simpler parallel programming model.

One of the main benefits in studying novice programmers will be their use of intuitive thinking. In parallel-programming classes, students are taught basic patterns of parallel problems that make it easier to create solutions. Since novice programmers will not have been taught those series of patterns, the ease of a programming language's syntax to suggest possible solutions will be highlighted in the productivity measurement. Another benefit of studying novice programmers will be to highlight problems that might not appear in advance programmers' code. Consistent problems in novices' solutions will highlight what could be improved in either future versions of SOFRITAS or other parallel programming tools like it.

One of the main challenges in studying novice programmers will be the variance of their skill. Due to the wide range of abilities that subjects can bring to the table, it is possible the a measured difference in productivity could actually be due to the difference in the programmers' ability to think about problems and not the difference in the use of the programming tools. Another challenge in studying novice programmers will be the significant increase in beginner struggles such as syntax errors. This study will account for both of these challenges.

## II. Productivity Measurement Method

### A. Hour of Code Format

The inspiration for the way this project measures productivity comes from hourofcode.com. This website teaches people the basics of programming using a visual programming language (VPL). At the start, only a few specific blocks are given that represent code. As the user solves problems presented by the environment, they are gradually given more blocks and more complicated problems. By the end of an hour, the user has been introduced to most of the major programming concepts that can be applied to any language.

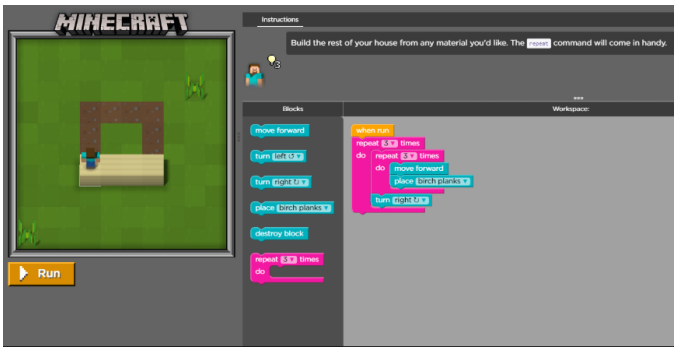


Fig. 1. Minecraft Hour of Code Game

This project will create an environment similar to Hour of Code for teaching parallel programming. It will also use a visual programming language that introduces blocks and problems, however the blocks will be representative of either SOFRITAS or traditional locks. Both languages will be taught using the same set of problems. This will make it possible to measure the effort users of the SOFRITAS VPL must invest to solve those problems compared to users of the traditional locks VPL. This solves both of the problems for studying novice programmers. Once created, it will allow a large number of people to go through the series of problems on multiple computers. This means the results from a large population sample can be averaged to reduce the effect of beginner variance. The use of a VPL will also mean that users will not be able to make syntax errors and the code will always be able to compile and run. This means the results will be affected by beginner errors the least.

### B. Blockly

Blockly is an open-source library from Google that adds a VPL editor to a website that allows the user to compile and run code on their client from a browser. As an open-source platform, it has the built-in ability to create any custom blocks. This will be the main method of creating the VPLs. It will also make it possible to passively measure productivity metrics such as how long it took the user to create a solution, how many attempts they went through, and how many extra lines of code they created and later deleted.

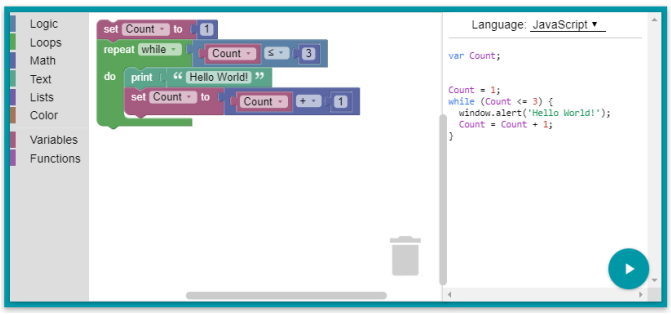


Fig. 2. Example of a Blockly editor

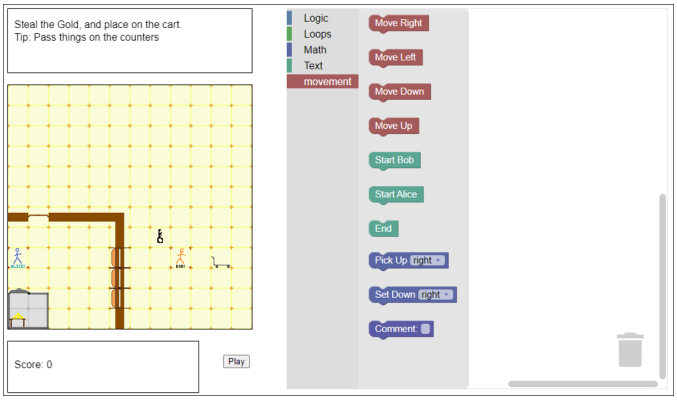


Fig. 3. HELPUS Game Layout

## III. Fall Semester Progress

### A. HELPUS Layout

HELPUS is broken up into two sides. The left side presents the game board with instructions for completing the level and the score. The right side is the VPL. The user is able to control everything on the right side in order to solve the puzzle on the left. The board itself is a 2D grid which represents a 2D array of memory addresses.

### B. Game Rounds

Users begin the first few rounds have only one character and limited blocks available. This allows the user to learn the mechanics of the game easier. As they progress, more blocks are added to teach them how characters interact with other features on the board. Eventually a second character is added to the game, which the user learns to control at the same time. Once users have been introduced to all of the VPL tools that they can use, rounds are presented that represent various parallel challenges. Figure 3 shows one of the intermediate rounds that represents a two-step pipeline problem. The two characters are on either side of a partition, but are able to pass things across a counter. The task that is presented requires the first character to complete a number of tasks which each must be handed off to the second character to complete.

### C. Available Blocks

HELPUS adds three types of blocks to the Blockly API. The first set are the Start/End blocks. These blocks delineate the

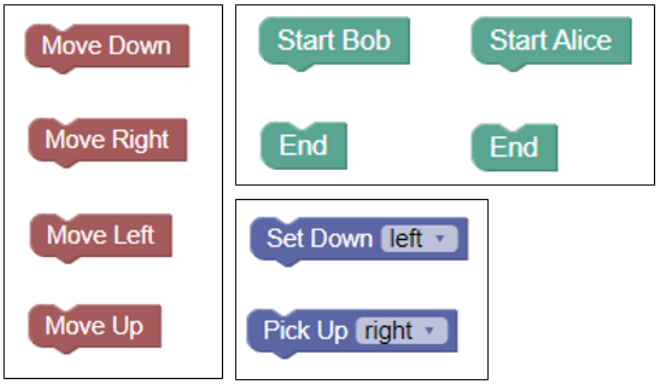


Fig. 4. Blocks available to the User

different threads that the user creates. A new set of Start/End blocks are given to the user for each character. When the game is run, each thread is initialized once. The second set of blocks are the movement blocks. These allow the characters to move around the 2D board to access different items. The character's movement is analogous to a memory pointer moving to various memory locations. The third set of blocks are the Pick Up/Set Down blocks. These allow the characters to interact with the board. Each character's inventory represents localized memory. The pick up/set down blocks are the way in which the character simultaneously locks a specific item and moves it to their localized memory.

## IV. Future Work

### A. Round Creation

The next step in completing the HELPUS VPL is to create more game rounds. Currently, the game only has three designed rounds to represent single-process, embarrassingly parallel, and two-step pipeline. Rounds will need to be created to represent further parallel patterns, such as replicable tasks, repository access, divide and conquer, recursive function access, event-based coordination, and heterogeneous computing.

### B. Fielding

Initially, the game will be fielded to other students and professors in the ECE department on an individual basis. This will allow a game developer to monitor their progress in-person to understand what aspects of the VPL need to be tweaked as well as real-time feedback from the players of the game. Following these initial tests and subsequent revisions, the game will be fielded to students who are currently in, or have recently completed, the SY110 and EC310 courses. This group will encompass students who have had at least an introduction to programming, but will still be novice programmers and will have not encountered parallel programming.

## References

[1] C. DeLozier, B. Lucia, A. Eizenberg, J. Devietti. *SOFRITAS: Serializable Ordering-Free REgions for Increasing Thread Atomicity Scalably*, University of Pennsylvania, March 2018

[2] E. Pasternak, R. Fenichel, A. Marshall. *Tips ofr Creating a Block Language with Blockly*, Google, 2017

[3] V. Pankratius, A. Adl-Tabatabai *A Study of Transactional Memory vs. Locks in Practice*

[4] L. Hochstein, V. Basili, M. Zelkowitz, J. Hollingsworth, J. Carver *Combining self-reported and automatic data to improve programming effort measurement*, University of Maryland, 2005

[5] L. Hochstein, J. Carver, F. Shull, S. Asgari. *Parallel Programmer Productivity: A Case Study of Novice Parallel Programmers*, University of Maryland, 2005