FLAG LAB

Dissertation

Code as Art, Art as Code

Nicolás Londoño Cuellar

May 12, 2025

This thesis is submitted in partial fulfillment of the requirements for a degree of Undergraduate in Systems and Computing Engineering.

Thesis Committee:

Prof. Name Last name (Promotor)	Universidad de los Andes, Colombia
Prof. Reviewer 1 Reviewer 1	Institution, Country
Prof. Reviewer 2 Reviewer 2	Institution, Country

Code as Art, Art as Code

Systems and Computing Engineering Department

FLAG lab

Faculty of Engineering

Universidad de los Andes

Bogotá, Colombia

This manuscript has been set with the help of TeXShop and PDFLaTeX (with BibTeX support). Bugs have been tracked down in the text and squashed thanks to *Bugs in Writing* by **?**] and *Elements of Style* by **?**].

Thanks to everyone who made this possible and believed in the power of computation.

Abstract

Everyone should have the opportunity to interact with code just as the average programmer does no matter the condition or background they may have. The Simple Cubic Language for Programming Tasks (SCulPT) is a programming language and artistic ecosystem designed to challenge the status quo of programming languages and computational thinking. SCuLPT disrupts the standard typewriter as the default interface for coding and programming languages with an intuitive, 3D interface and its specification. This is done by creating programming and artistic constructs that become tangible objects such that, when composed, generate computable programs and art pieces. Along the language, we provide a set of tools aimed to make art a tool for coding and programming an art form. The constructed language is a fully capable, Turing complete language that serves as a creative outlet as well. Using said tools, we perform an empirical validation of the experience with novices and expert programmers alike. The results show programming proficiency in solving small computational programs across the two populations, and some really neat art!

Acknowledgements

Having so many to thank in such a small space is jarring, nonetheless, I'll be susinct. First and foremost, I would like to thank my advisor, Dr. Nicolas Cardozo, for listening to a weird idea from an even wierder student. His guidance has been incredible not only in this work but also what follows from it.

I would also like to thank the members of the Crypto and PIL groups at the FLAGLab. They gave me a space to ramble about printed plastic pieces and wierd programming languages.

I'll probably never be as good as a designer as my sister, Sara, but I'll always be grateful for her help in making the SCulPT visual language, posters and documentation look good and work as intended.

I also owe a lot to my mother, who has always supported me in my endeavors, even when they seem a bit out there, rushing to help with the intense manual labour that went into this endeavour.

At times I've been told that I should be more serious, but I've always been quite aloof. I'd like to thank my friends and my parter for putting up with me and my weirdness, and for always being there to help me out when I needed it. Always at the ready to make me crack a smile or two when times got rough and my hands numb from hand-sanding every printed piece of SCulPT.

Lastly, I want to thank you, the reader, wheather obligated or not to read, for taking the effort to push through several pages of me rambling about a quirky idea that turned into so much more. I hope you find it as interesting as I do, and that you can see the potential in it.

Contents

List of Figures

Chapter 1

Introduction

The typewriter interface, through the keyboard, has been the reigning interface to materialize programs into code since the beginning of programming.

Interface designs lack what is needed to support every user. Naturally, users are not created equally. Prior work in the user experience and human-interface integration show reports in which technology and its designs can discriminate against users due to several factors, such as different physical or cognitive capacities, race and gender [**ko23**]. Moreover, programming has a steep learning curve and a high entry level [**bosse17**]. Different approaches to computing are needed to lower the entry threshold and make computing more approachable to everyone.

Current programming languages are based on the same principles that we have used since the 50s. The same principles that were designed to be used with a typewriter and a flat surface in which to write. To lower the entry threshold of programming and computing, more approachable interfaces and methods are required. New fields in computing are emerging with different intents but still the same interfaces [**hongji16**]. In order to explore new methods of computing, we must look further ahead than the tools we, into how we approach code in general.

To do so, we must shift attention to other mediums that can be used for computation, yet serve a complete different purpose. Such path could lie into the realm of art. Using coding tools to create art is not a new concept. Many artists today embrace code in their practice, creating tools and environments to create interesting pieces of work. For example, Sonic Pi is an entire coding suite built over SuperCollider sound synthesis server that allows users to create music using code with a superset of Ruby tailored for live coding performances and initially conceived as a programming teaching tool for children in England [**aaron16**].

Artlang is an artistic sandbox with rules that ensure correct computation, shifting the focus away from computation, yet relying heavily on it to create physical sculptures that, by design, compile to valid programs.

1.1 Objectives

In the creation of Artlang we posit two objectives. First, we require to build a fully capable programming language addressing the concerns described previously about the state of programming, computation and programming languages. Artlang is designed to be a visual and physical oriented programming language, with a heavy focus on the ease of use, computational correctness and aesthetics.

Second, we aim to evaluate the language in two main aspects. First, we intend to evaluate the ease of the language to foster computational thinking without the intrinsic difficulties of current programming languages. Second, we evaluate the properties of Artlang as a paradigm that breaks existing preconceptions about programming languages and moves beyond the typewriter interface by leveraging artistic creativity as an alternative.

1.2 Methodology

Artlang as a whole is in fact two interconnected languages, one being the visual language and specification that defines the components from a strictly physical standpoint and the other being the programming language that is used to define the behavior of each component. For the development of Artlang, we have followed a user-centered design approach, creating a visual language that is both usable and accessible to a wide range of users. Artlang's focus on usability and accessibility is displayed by using simple but distinctive shapes and forms for each component on the language. This first design choice allows for users to easily identify and differentiate between different components of the language, making it trivial to convey meaning behind each shape while allowing people with any background to recognize each component foregoing knowledge, language or capacity barriers that arise from conventional programming languages.

Regarding the programming language, Artlang is designed to be a Turing complete language. By building structures and following the intuitive rules of the language, users can create any program they desire. The language is designed to be simple and easy to understand, with a focus on the visual representation of the program rather than the code itself. Code block are physically assembled together to create a program, with each block representing a different function or operation. Block are designed such that their connections are intuitive and easy to understand, with each block having a set of features that must be connected to other blocks in order to function correctly. This allows users to create programs without the need to understand complex syntax or semantics, making it accessible to users with little to no programming experience.

Regarding the language's aesthetics, Artlang is designed to be an artistic sandbox therefore the rules and specification for how shapes should look and feel are very loose. This allows users to tinker with the components, suiting them for their needs and desires. The only rules that must be followed only consider are component connections and a set of features each block requires. Final shape, size, material, color or texture are completely open for anyone to modify without any impact on the program. This provides a sense of freedom and creativity that is not present in other programming languages, allowing users to express themselves through their code and create unique and personal pieces of art. This also implies that the language can be fitted to almost anyone's needs, allowing accessibility and customization for users with different physical or cognitive capacities.

Artlang current implementation features small 3D printed components with varying colors. The printed pieces follow the standard Artlang implementation with no modifications to the shapes. The components are connected using magnets and screw elbow joints, allowing for easy assembly and disassembly of the components, making it easy to create and modify programs on the fly.

1.3 Results

Artlang as a programming language is a functioning language capable of writing any program. To validate such a claim, we have written a program in Artlang, an implementation of the Fibonacci sequence. The language uses FILO (First In Last Out) stacks to store and manipulate data. Each stack is represented by a different shape other than the reserved shapes used by the instructions, and they are initialized on first call. Stacks can hold an arbitrarily large amount of rational numbers. Stacks allow the pop, push, move and duplicate operations. Numbers are defined by their literal counterparts, and the language supports basic arithmetic. Arithmetic operations are performed by popping the first two elements of the stack, performing the operation, and pushing the result back into the stack. Artlang allows for flow control through the use of loops, conditional blocks and jump blocks. Loops are physically represented by a literal loop of connected blocks. Conditional statements are represented by a question block using a value as a parameter. Positive values are considered true and evaluate the following block, and negative values are considered false and do not evaluate the affected block.

Artlang is based on the physical blocks. Blocks as three-dimensional objects are unwieldy to be described in text. Therefore, we will use a simplified notation to represent the blocks in planar space. Every block is represented as a square with the shape of each block inside the square. The connections between blocks are represented by arrows between the squares.

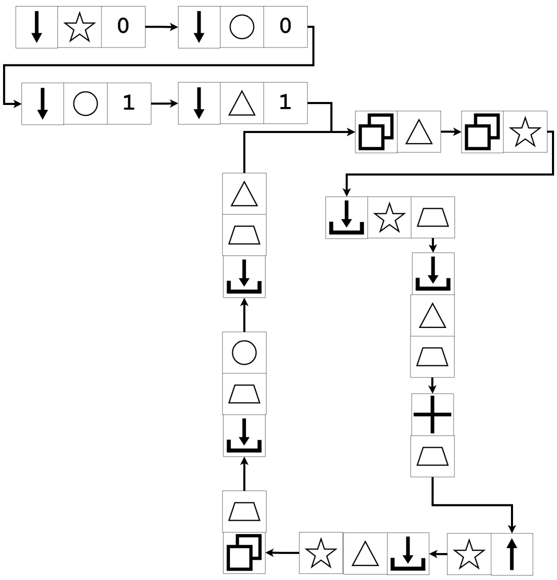

Figure 1.1: Artlang Fibonacci program

The program in Figure **??** fills the cirle stack with the numbers of the Fibonacci sequence. It can be rewritten as a more traditional algorithm as follows.

Algorithm 1 Fibonacci sequence in Artlang

Push 0 *into* **Star**
Push 0 *into* **Circle**
Push 1 *into* **Circle**
Push 1 *into* **Triangle**
while True **do**
    Duplicate **Triangle**
    Duplicate **Star**
    Move **Star** *to* **Trapezoid**
    Move **Triangle** *to* **Trapezoid**
    Add **Trapezoid**
    Pop **Star**
    Move **Triangle** *to* **Star**
    Duplicate **Trapezoid**
    Move **Trapezoid** *to* **Circle**
    Move **Trapezoid** *to* **Triangle**
end while

1.4 Conclusion and Future Work

Artlang is a visual and physical oriented programming language that aims to lower the entry threshold to programming by leveraging artistic creativity as an alternative to the traditional typewriter interface. We have presented the design and implementation of Artlang, a language that allows users to create physical sculptures that compile to valid programs. Artlang's design choices make it possible for anyone to create programs without the need for prior programming knowledge, making it accessible and modifiable to a wide range of users.

While the validations done so far are only preliminary, they show promising results in terms of usability and accessibility, while resulting in interesting pieces of art.