PixelDraw: A Declarative Language for Pixel Art Generation

Nicolás Alberto Rodríguez Delgado, Daniel Mateo Montoya González
Department of Systems Engineering Universidad Distrital Francisco José de Caldas Emails: niarodriguezd@udistrital.edu.co, dmmontoy

***Abstract*—PixelDraw is a declarative language designed for generating pixel art images using simple, human-readable commands. It leverages compiler design principles to translate high-level graphical instructions into a pixel matrix, serving as an intuitive tool for teaching programming and logic through visual feedback. The system architecture includes lexical, syntactic, and semantic analyzers, culminating in the visualization of pixel grids. This paper describes the language's grammar, compiler architecture, and implementation roadmap.**

***Index Terms*—Pixel art, declarative languages, compilers, regular expressions, Python**

I. Introduction

Programmatic graphic generation has become a valuable tool in education and creative digital design. Although environments such as Logo, Scratch, and p5.js offer solutions for beginners, they often require prior programming knowledge or rely on procedural constructs. PixelDraw proposes a lightweight declarative approach where users specify "what" to draw rather than "how" to draw it, making pixel art creation more accessible.

This project aims to design and implement a compiler capable of interpreting high-level commands and generating pixel arrays, providing an educational tool for exploring compiler components and pixel-based rendering.

II. Related Work

Visual programming environments such as Scratch and Logo have been widely adopted for teaching programming concepts. Scratch employs block-based coding to reduce syntax complexity, whereas Logo utilizes turtle graphics for procedural drawing. However, both tools present learning curves and require understanding of control flows and loops.

Declarative languages for image generation are less common. Tools like Processing (p5.js) focus on procedural graphics, requiring more extensive coding. PixelDraw differentiates itself by simplifying image creation through intuitive commands (e.g., `SIZE`, `COLOR`, `POINT`), making it ideal for beginners and educational settings.

III. Proposed System

PixelDraw's compiler architecture follows a traditional pipeline with adaptations for its domain-specific language:

1) **Lexical Analysis**: Tokenizes source code into meaningful units (e.g., `SIZE`, `COLOR`, numbers).
2) **Syntactic Analysis**: Validates the token stream against the language grammar using a recursive descent parser.
3) **Semantic Analysis**: Interprets tokens and generates a pixel matrix while managing state (canvas size, current color).
4) **Visualization**: Renders the pixel grid using Python's Tkinter library.

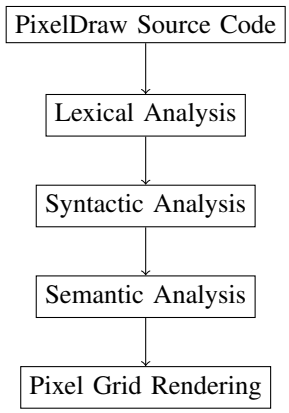
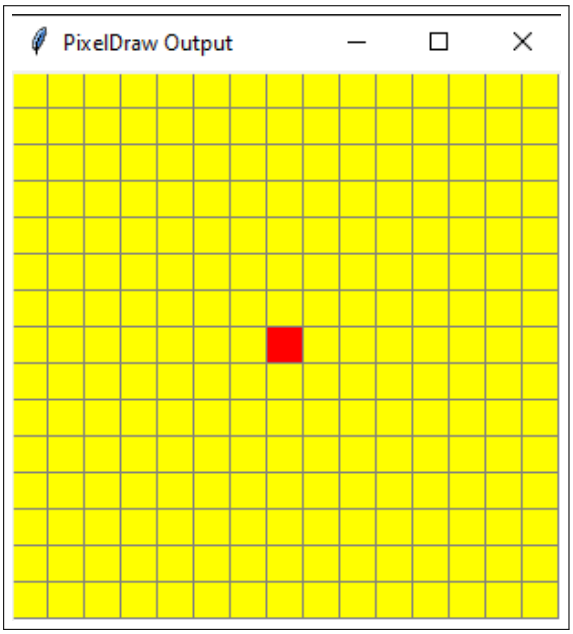


Fig. 1. System Architecture of the PixelDraw Compiler

A. Example Code

```
size 15x15
color #FFFF00
rectangle 0 0 15 15
color #FF0000
point 7 7
```

This input produces a 15x15 yellow canvas with a red point at the center.



IV. Methodology

The compiler was implemented in Python, structured into four modules:

- **LexicalPd.py**: Handles tokenization using regular expressions.
- **SintacticPD.py**: Parses tokens according to grammar rules.
- **SemanticPD.py**: Interprets commands and generates pixel data.
- **CompilerPD.py**: Integrates all components and handles visualization.

V. Expected Results

The compiler is expected to successfully process PixelDraw code, generate pixel arrays, and render images. Educational outcomes include enabling students to grasp compiler phases and engage creatively with pixel art.

VI. Conclusions

PixelDraw demonstrates potential as both an educational and creative tool. Its declarative language reduces barriers to digital art creation and provides an accessible way to learn the fundamentals of compilers. Future work includes expanding the instruction set, supporting animations, and exporting images in multiple formats.

References

[1] H. Abelson and A. diSessa, *Turtle Geometry*, MIT Press, 1986.
[2] Python Software Foundation, Python Language Reference, Available: https://www.python.org
[3] The Pillow Project, Available: https://python-pillow.org
[4] A. Aho et al., *Compilers: Principles, Techniques, and Tools*, Addison-Wesley, 2007.