COMP6991 Rust Logo Interpreter Assignment

Logo Language Overview

- Logo is a programming language derived from Lisp and others.
- Older programmers often had their first programming experience with Logo.
- Key feature is a "turtle" for drawing by picking up and putting down a pen and moving around.

Assignment Goals

- Practice designing and structuring a larger Rust program.
- Focus on modern programming skills and design patterns.
- Have fun creating an aesthetic and interesting application while connecting with programming history.

Assignment Requirements

- Build a Logo interpreter that writes to an.svg or.png image using the `unsvg` crate.
- Handle various Logo commands for turtle control, variables, queries, conditionals, and math operations.
- Implement design excellence features for full marks.

1. Introduction to the Logo Language

Tokens

- A token can be a procedure (like a function), a variable (prefixed by `:`), or a value (prefixed by `"`).
- Procedures always take a fixed number of arguments.
- Values in Logo are always strings, but some like `"TRUE"` and `"FALSE"` are interpreted as booleans and some can be parsed as numbers.

Program Structure

- A logo program consists of lines of text split into tokens by whitespace.
- Lines starting with `//` or empty lines are ignored as comments.

2. Introduction to Unsvg

- The assignment uses the `unsvg` crate to generate SVG or PNG images.
- `unsvg::Image` represents an image and has methods like `draw_simple_line`.
- `unsvg::get_end_coordinates` returns where a line drawn from a given point would end.

3. How Your Program Will Work

- Produce a program called `rslogo` that takes four arguments: a logo program file (.lg), the output SVG/PNG file path (.svg or.png), image height, and image width.
- Read the logo program, parse and execute it line by line.
- Exit with a non-zero return code and print an error message if there's an issue.
- Write an SVG or PNG using the `unsvg` crate if there are no issues.

4. Design Excellence

- Options for design excellence include making beautiful errors, achieving 80% test coverage, using a parser combinator library, creating a facility for language extensions, building a zero-copy program, contributing to the `unsvg` library, or building a transpiler.
- Markers will consider a reasonable attempt at one of these tasks as sufficient.

5. The Tasks To Complete

Part 1: Turtle Control (20%)

- Control the "turtle" which is like an invisible pen that can draw on the image.
- Turtle starts "up" (not drawing) in the center of the screen facing straight up.
- Commands include `PENUP`, `PENDOWN`, `FORWARD`, `BACK`, `LEFT`, `RIGHT`, `SETPENCOLOR`, `TURN`, `SETHEADING`, `SETX`, `SETY`.
- Turtle can go off the image without causing an error.

Part 2: Variables and Queries (20%)

- Implement the `MAKE` command to create and assign variables.
- Implement the `ADDASSIGN` command for variable increment.
- Support "queries" like `XCOR`, `YCOR`, `HEADING`, `COLOR`.

Part 3: IFs, WHILE, [ ] (20%)

- Implement `IF EQ` and `WHILE EQ` commands for conditional execution and looping.

Part 4: Implementing Maths and Comparisons using a Stack (20%)

- Implement operations in Polish Notation like `EQ`, `NE`, `GT`, `LT`, `AND`, `OR`, `+`, `-`, `*`, `/`.
- Implement stack operations for `IF` and `WHILE`.

Part 5: Logo Defined Procedures (20%)

- Implement procedures analogous to functions in other languages.
- Procedures are defined with a line starting with `TO`, followed by the procedure name and arguments, and ending with `END`.

6. Common Questions

Design Approaches

- Line-by-line approach: Execute each line in turn while storing more data and state.
- Parse then execute approach: Convert the text into an Abstract Syntax Tree and read from it.

Planning

- A happy middle is suggested: think about the approach and read through the assignment without spending more than 30 minutes planning.

Using AI

- Permitted uses of AI include seeking help with concepts, pattern matching, generating skeletons, and writing tests.

7. Other Information

Submission

- See instructions at the bottom of the page.