

PLT LSystem Presentation

Ethan Hann
Jervis Muindi
Michael Eng
Timothy Sun

Lsystem: Introduction

- Short for Lindenmayer systems
- Grammar containing:
 - Alphabet of symbols
 - Initial string
 - Production rules
- Example- Koch Curve
 - Variables: F
 - Constants: + -
 - Initial string: F
 - Rules: $F \rightarrow F+F-F-F+F$
 - F = “draw forward”, + = “Turn left 90 degrees”, - = “Turn right 90 degrees”

L-system language goals

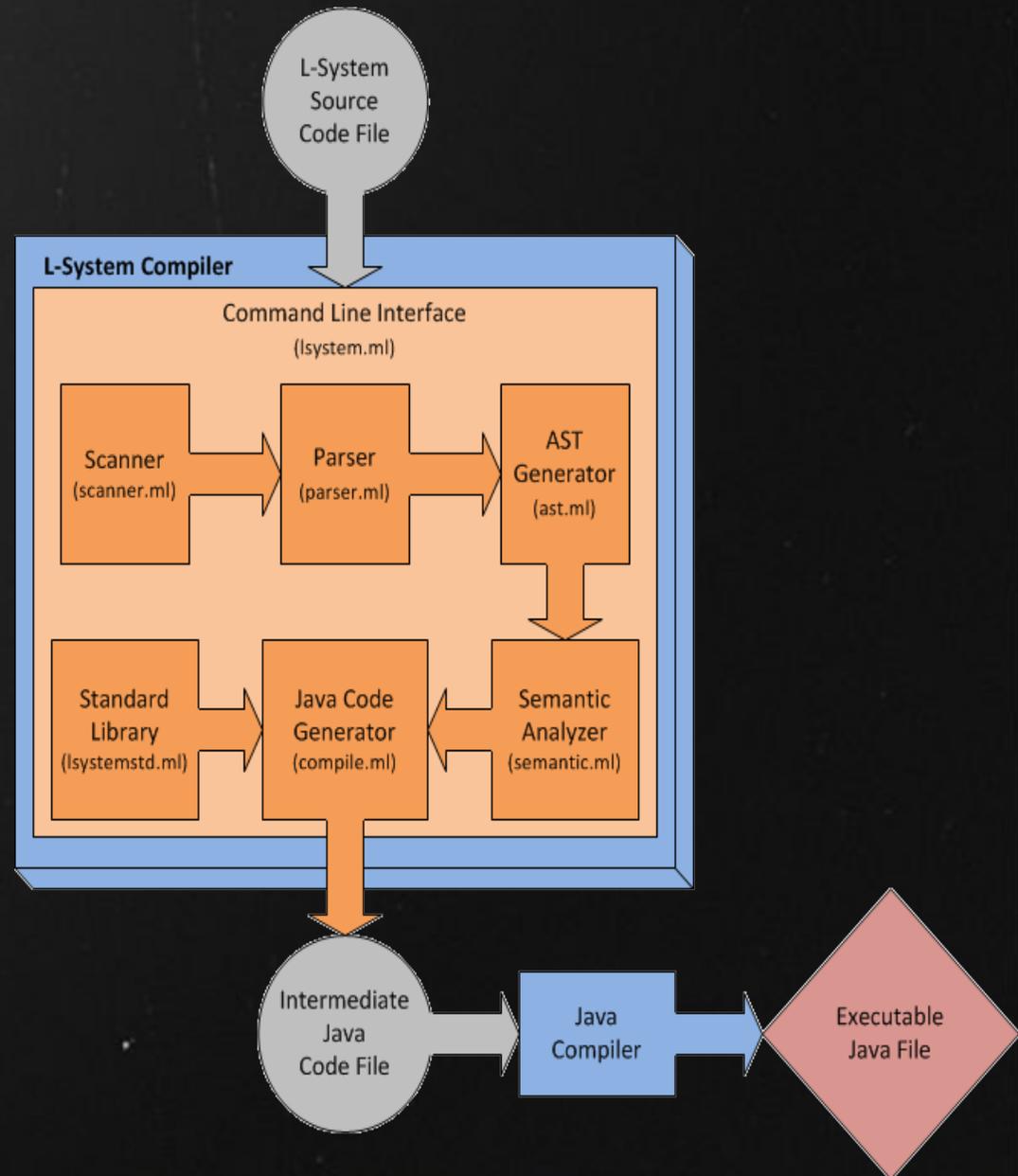
- Intuitive
 - Simple syntax, short programs to display L-systems
- Customizable
 - Can manually map terminals/variables to drawing commands
- Portable
 - Once fully constructed, compiler only needs JDK and JRE to compile intermediate Java files into class files and execute them

Language Tutorial

- <http://ethanhann.github.com/Lsystem-Compiler/>

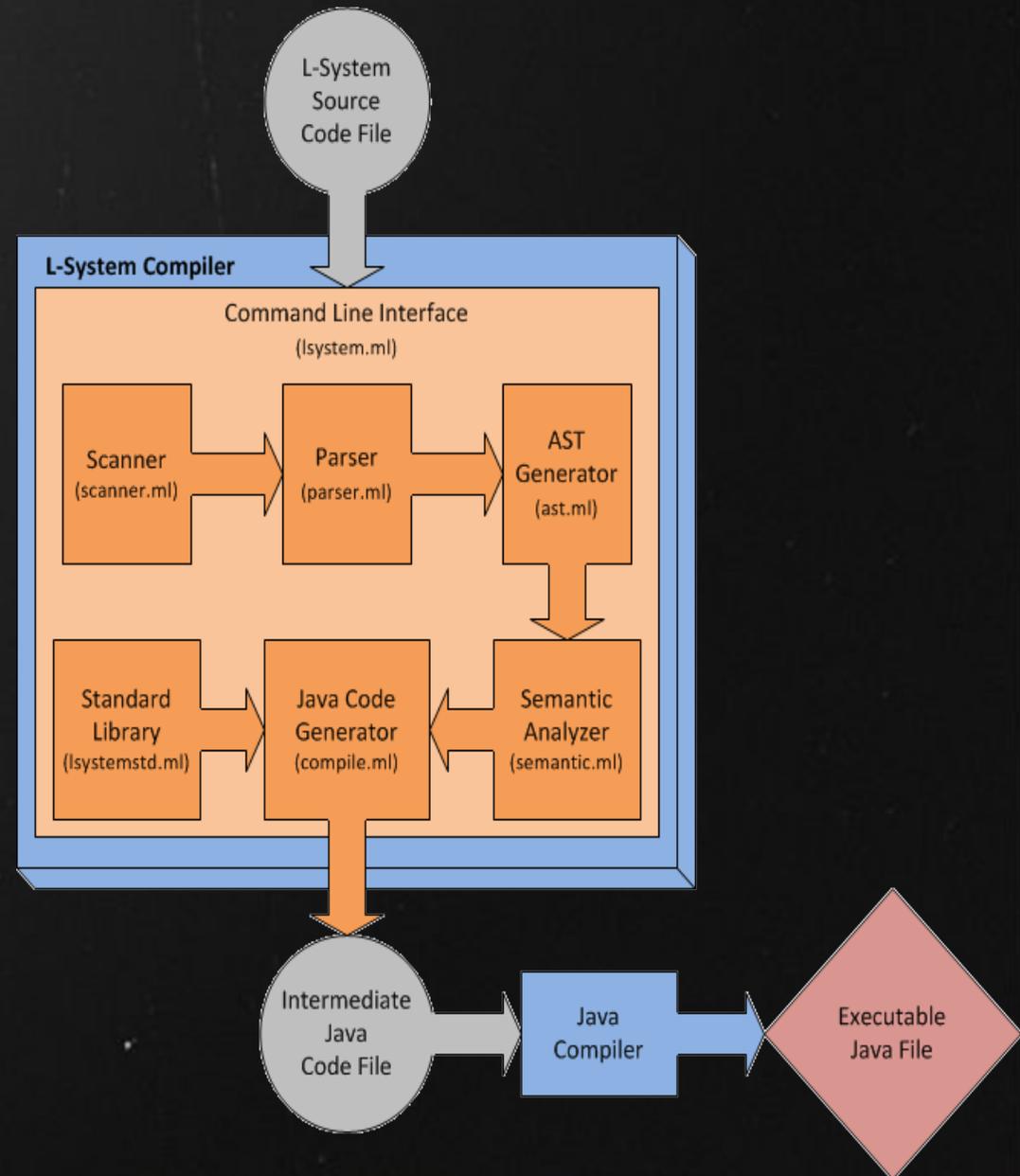
Language Implementation

- Scanner recognizes language tokens
- Parser consumes tokens and validates program in syntactically correct.
- AST is generated in conjunction with parsing
- Semantic Analysis done on AST



Code Generation

- A program consists of compute functions and draw functions.
- Translate Compute Functions
- Translate Draw Functions
- Output java source code
- Compile to Java



Lessons Learned

- functional programming is slick
- SVN (google-code)
 - > git
- Testing with the compiler, not after
- Keep things simple
 - more restrictive syntax -> more semantic analysis
- not something to be done overnight
 - had to do it little by little

Advice for future teams

- start early, designate tasks
 - even the final report
 - get used to O'Caml
- microC is your go-to reference
- process should be enjoyable
 - like the language you make
 - doesn't have to be the final product you envisioned
- coding standard
 - (silly whitespace)
- Nibble at it
 - one feature at a time
 - try to look at the code regularly

Thank You !