Motivation

* My background: Want to study interdisciplinary approach between neuroscience and computer science.
* Approached MBB professor, he suggested we do a project together.
* He studies fear, but clinical methods of testing it are lame (give example of plane crash experiment from 60’s); he wants “video game-like” approach
* Intelligent agents needed.
* Knew I wouldn’t have time to do something outside of class, so I tried to think of a way to incorporate it into my PLT project.
* Why not a language for modeling the behavior of intelligent agents?
* Prof. has since moved on – he couldn’t wait a whole semester for me to finish this project – but I may revisit it with him. Also, the need to model intelligent behavior has plenty of other applications.

Overall Geppetto Language Structure

* “Rule-driven analysis engine”
* YACC-like structure: Declarative section that defines objects called entities, followed by rules that act on them, followed by section for supplementary code (functions)
* No entry point; Geppetto program is inherently event-driven
* Imperative part of language is very much like C or Java, with the same semicolon-delimited statements, the same basic primitive data types, the same syntax for expressions and function calls, etc. But the imperative part isn’t what Geppetto is about, so I didn’t worry too much about being innovative there. That said, I think this was actually the slickest, most solid part of the project implementation. I hope you get a chance to look at the code and comment on it.

Disclaimer

* Basic idea is sound, implementation of design is sound, but the part in between – the design of the language itself – is a little shaky. It doesn’t do much and what it does do is probably done more easily in other ways.
* Hamstrung by simplicity of rule evaluation. Right now every entity must be explicitly referenced. Wanted to have a variable for that. Also wanted to add first-order logic. Also wanted to apply some of the principles & algorithms from AI. But we weren’t able to do any of that.
* I hope the flaws in the language design and the things we weren’t able to add don’t hurt our grade too much.
* I actually got in most of the basic language elements we wanted. Pretty pleased with that given limited resources.
* This (or any) language composed of: variables, functions, expressions, and statements. Given that all these elements are now working, we could probably repurpose Geppetto to do something else pretty quickly.

Project Management

* “Nothing is particularly hard if you divide it into small jobs” – Henry Ford
* “'Begin at the beginning,' the King said gravely, 'and go on till you come to the end: then stop.'” – Lewis Carroll, *Alice’s Adventures in Wonderland*
* No specializing.

Translator architecture

* Architectural block diagram from project report
* Lexer -> parser -> GeppettoProgram (AST) -> Interpreter

Runtime environment

* Geppetto is a Java-based interpreted language, so to run it you also need Java.
* java –jar gepptto.jar programfile.gep

Tools

* Git
* Eclipse
* JFlex
* BYACCJ
* JUnit?

Overall Geppetto Project Structure

* Package layout
* Most functionality encapsulated in domain classes (e.g., expressions know how to evaluate themselves, statements know how to execute themselves, etc)
* Parser is single biggest piece of code
* The main program and interpreter themselves are quite small (show screen shot)

Testing

* toString()
* simple.gep, predator-prey.gep
* Nascent JUnit tests, but honestly that’s just for show – they weren’t used in the creation of the project

Demo

Wrap-up

* I was worried more about getting the thing working and completing the functionality in the LRM than in creating a work of art.
* Due to the parts of the language we didn’t implement and pressing time concerns, we never really got the chance to dream up an impressive demo for the language to show what it can do.
* I intend to keep tinkering with it, and get it into a state. Hopefully that psychology professor is still around so I can approach him with this!