Final Project Proposal

Evan Silberman

11/2/06

I am designing a domain-specific language (DSL) for the creation of interactive fiction (IF) games.

1 Purpose

The purpose of my project is to explore programming language design and metaprogramming techniques in Ruby through the creation of a DSL for creating IF games. My vision is to create an intuitive interface for the game writer, my challenge is to employ the host language effectively to achieve this vision. The value in the project is to be found in both the technical implementation and the design of the interface. Human-computer interaction is often misunderstood as "creating pretty GUIs." My goal is to create a programming language interface that is easy for a non-programmer end-user to use for writing IF games that hides implementation details while allowing expressiveness.

2 Scope of Study

I intend to implement a reasonably large subset of the basic features of IF games. Namely, I will minimally allow the programmer to create rooms and link them together, create characters and objects, allow one-level (not tree-based) conversations with characters, allow the hero to pick up and put down objects, allow events to be triggered that change the game state, and allow scoring and victory conditions. Though there are other possible features of IF that will not be implemented in my language, my system should allow for games of reasonable complexity. If I finish all of this, my "extra hard" project may be to allow the extension of the grammar system by the programmer.

3 Background

Interactive fiction has been around for a long time. For about a decade, Inform 6 has been the standard language for enthusiasts writing IF games. It allows very complex games, but its syntax is not necessarily intuitive and has lots of brackets and whatnot. Earlier this year, Inform 7 was released, encompassing an IDE, a language specification, and a compiler. Inform 7 has a syntax that is very close to natural language, operating within the paradigm of writing a book. Creating a system like this is beyond my capabilities, but hopefully my system will be more intuitive and easy to use than Inform 6.

4 Procedure and Methodology

During the first quarter, I created a functioning first version of the backend for the IF language (as yet unnamed). The frontend uses some top-level methods in Ruby that accept code blocks as arguments; the backend uses metaprogramming to pass the evaluation of these code blocks to instances of various classes that are hidden from the programmer.

Metaprogramming is my primary tool for implementing the language. In recent years, Ruby's metaprogramming features have been used to implement a variety of hosted DSLs, including some of the features of the Ruby on Rails web framework. Using metaprogramming, methods and instance variables can be created and set on the fly, and explicit receivers of method calls can be left undeclared, only to be provided later. My plan is to simultaneously expand my understanding of metaprogramming techniques and add functionality to the language. There is lots of prior art in the realm of Ruby DSLs, so I will be able to look at other examples to gain some inspiration for my code.

Since my goal is to create an easy and intuitive interface for writing IF games, the true test of my project is not that the language works in combination with the backend, but that the language works in combination with the user. I intend to distribute the software to my classmates, allowing them to experiment with the language and provide me with feedback as to its ease of use and functionality. My users will also hopefully provide me with suggestions for features to implement, which I will evaluate in terms of ease of integration with the existing system and possibly add on. My features will grow organically to the point where the set of functionality that I'm aiming for is implemented, and not by a timeline.

5 Expected Results

If I achieve my vision, the end result will be a reasonably powerful system for creating interactive fiction. If I succeed, I will know because my peers and other users will be able to create games in the language in an intuitive fashion, and without encountering too much frustration (unless they want a feature I didn't include, in which case they can add it themselves, due to the dynamic and interpreted nature of Ruby). The results will be the games people have created, and hopefully future researchers interested in language interfaces will find something valuable within my research.