A Domain-Specific Language for Interactive Fiction

Evan Silberman, TJHSST Computer Systems Lab, 2006-07

Abstract

A domain-specific language (DSL) is a programming language designed to be used for a specific and limited set of tasks. Using metaprogramming techniques, I designed a DSL hosted within Ruby for creating interactive fiction games. My goal was to create an intuitive and expressive language for creating IF games while hiding the details of implementation from the programmer.

```
#example 1
room :living do
   name “Living room”
   desc “This is not tom's favorite room”
   exit :north, :fantasy
end
```

Introduction

Metaprogramming is simply the technique of writing code that writes code. The Ruby programming language contains extensive built-in facilities for metaprogramming. Using these features, a programmer can pass responsibility for evaluating blocks of code to different receivers than the ones implicitly being used by the code and generate code for methods on the fly. I used these techniques in the creation of my DSL for IF game writing. By passing details of implementation to a backend, I was able to keep the syntax of the frontend code (which, being hosted, is all valid Ruby code) simple, intuitive, and terse. (Example 1)

Methods

Several Ruby metaprogramming features simplified the implementation of the DSL. Instance_eval is a method which every Ruby Object has which takes a code block, then evaluates that code block as if the methods in it were instance methods of that object. This allows the syntax wherein details of a room are declared without the programmer knowing anything about the Room class. The method_missing method, which is called on an object when no instance method exists, allows the programmer to define arbitrary properties to be dealt with later for the rooms he creates.

Conclusion

Well, I obviously can't really conclude everything after only one quarter of work. So this'll be a status report instead. Room declarations and creature declarations are functional, and a rudimentary, though moderately enjoyable, proof-of-concept game has been written in the language and successfully played. One-layer conversations with other characters also work, and extending this functionality to more complex conversations is probably going to be a challenge.