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 IFAlpha, 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 IFAlpha. 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.

Ease of Use

The principle design goal of the language is ease of use. My intent is to make the language easy for non-programmers to understand and use. Game writers should not have to be programming experts spending long hours learning complex syntax to express simple relationships and objects for a game. The test of success for the IFAlpha project is not whether I create the most feature-ful system ever (I don't have the time or knowledge for that), but if the system I create is easy and intuitive to use.