Programming Language: My experiences

Iván Aveiga

February 7, 2013

Notify Me: Observations, Conclusions and Experiences [1]

1 Observations

As observations we highlight the work of using GoogleMaps' API in mobile apps, and the usage of mobile development in Android. Como observaciones podemos destacar el trabajo de usar la API de GoogleMaps en aplicaciones móviles y el uso de desarrollo móvil en Android. También el uso de software de versionamiento para controlar las versiones del proyecto.

Versioning is so helpful in the moment to develop because offer advantages as history of commits, workingroups, reverse commits and aportations of every member in the team. We use Github as Versioning Software.

2 Conclusiones

Was beneficial work with GoogleMaps's API because the big data of google in maps allows a rich experience, GoogleMaps
offers better maps than Bing, Nokia, Apple.

We analize work with Google Place for use categories, this will be analize in future versions of NotifyMe.

We analyze work with Google Place to use categories, this will be analyzed in future versions of NotifyMe.

3 Experience

Notifyme's Team consider a positive experience and beneficial from academics point, because versioning will serve for futures works and lifestyles remains.

Work with mobile technologies was interesting, allowed us into this field today is very profitable.

[1] Aveiga Iván, Enríquez Wilson, Sornoza Andrés.

Haskell: Mastermind[2]

4 Introduction

Mastermind or Master Mind is a code-breaking game for two players. The modern game with pegs was invented in 1970 by Mordecai Meirowitz, an Israeli postmaster and telecommunications expert, but the game resembles an earlier pencil and paper game called Bulls and Cows that may date back a century or more.

"One player is the code maker, and selects a combination consisting of four pegs, each of which can be either red, yellow, blue, green, black or white. All four positions must contain a peg, no blanks are allowed and a colour can be used one or any number of times in a combination. The other player is called the code breaker and has to find the combination, taking as few guesses as possible. After each guess is made, the code maker scores it with small score pegs. Black pegs are given for each coloured peg which has the correct colour and correct position. White pegs are given for each coloured peg which has a correct colour but is out of position. If none of the coloured pegs in a guess matches any of the coloured pegs in the combination, then no score pegs are given. The game stops when the code breaker finds the combination, the goal being to find the secret combination in as few guesses as possible."

5 Resolving

To resolve we implement a code breaker based in **A heuristic hill climbing algorithm for mastermind**, a paper from University of Bristol, the principal algorithm is:

1. We submit to the Code maker a random guess constructed with 4 genes that we call the "Current Favourite Guess" (CFG).
2. From the CFG, we induce a new potential code with the method described in section paper.
3. If potential code is not consistent with all previous guesses, go to step 2 otherwise submit it as new guess.
4. If submitted guess scores [0,0] then suppress from the pool of colours all the colours present in the last submitted guess. Then find a new random combination (with the new pool of valid colours) consistent

[2]Aveiga Iván, Enríquez Wilson, Íñiguez Pedro

with all previous guesses' scores and set this new combination as new CFG and submit it to the code setter.

5. If submitted guess score is as good as or better than CFG score according to the heuristic described in section 3.2, then set this guess to be our new CFG and also set the new score as best score.

6. If submitted guess scores [4,0], stop otherwise go to step 2.

6 Why Climbing hill?

We use climbing hill because none of the members has taken Artificial Intelligence to understand the basis of genetics algorithms, then we decided use Climbing Hill because is less complex.

7 Experiences

The principal problem in the moment of develop in Haskell is the big change of thinking model, changing from imperative thinking to mathematical thinking. In this project we consider the change of paradigm was so hard, because from firsts years the university introduces in the concept of imperative and the time to make this change is so short.

This short period of time makes the people have a bad experience with functional programming languages and this cause that not take the bennefits of functional programming.