Basic Text Survival

Comparison of Game Implementation

Ding, Yunyi yding13@ucsc.edu
Lin, Brian bjlin@ucsc.edu
March 6, 2015

The extensive popularity of computer and video game prompts game programming to be one of the fastest growing industries in the computer programming and technology field. "What language do I use for game programming?" has become a very popular question. There is no easy answer for this question because there are, of course, some computer languages that perform better than others on particular features and work better for certain applications.

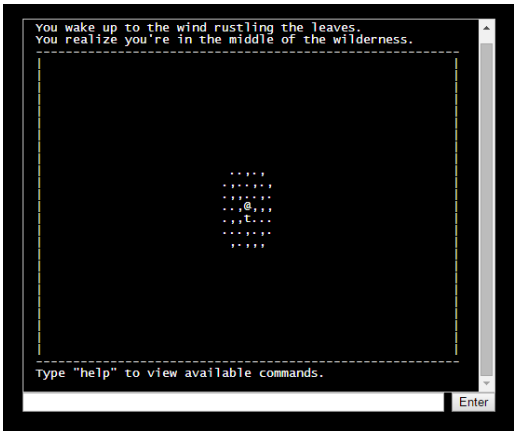
Overview

There are several programming languages are popular in game development, such as C++ and C#. They are preferred because they have certain powerful features that are beneficial for game making. For example, C++, an intermediate-level language, had object-oriented programming features that are good in high-performance server and client applications and video games. Although object-oriented programming languages are usually easier to use in game making, some other languages without that feature may have some potential benefits that can be explored.

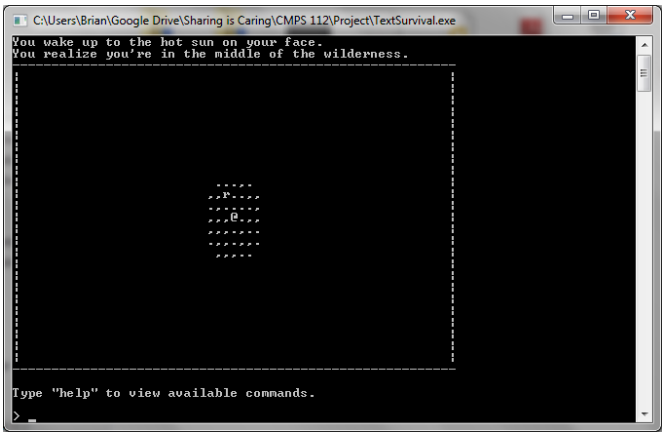
The goal of this project is to create a basic text based survival game in both JavaScript and Haskell and compare the implementation. By implementing the same game features in these two completely different languages, we compare how they perform in a simple game environment. Since the purpose of the project is to compare implementation of game features, the game itself will be less remarkable since it is not the focus.

The Game

The game is a very basic survival text game that implements a few features from typical survival games such as resource collection and an inventory system. The player plays the game by entering in commands. The game also displays a map of the environment through text.



JavaScript Implementation



Haskell Implementation

Game Features Implemented

Randomly Generated Environment and Character Movement:

The program generates the environment with randomly placed wilderness landmarks. As the player moves, more of the environment will be revealed. Only a certain range of space will be visible from a certain point and moving will reveal more of the environment.

Visual of Environment through text:

The program generates a text-based map of the environment so the player can see his/her movements, actions, and achievements by looking at screen directly instead of having to imagine the environment in his/her head. This visual feature makes the game more interactive and fun than traditional text based games.

Inventory System:

The inventory system manages items and objects the player collects from the environment during his/her exploration. The player can check the inventory by simply typing in the "inv" command, which will display the quantity and name of each object.

Item Functionality:

Most items and objects in the game are usable. The player can use anything in his/her inventory by typing the command "use [itemName]". For example, a tool can be used to cut down trees by typing in "use tool."

Health and Hunger System:

The character's health and hunger status can be viewed by typing in "status". The player has to manage his/her hunger by eating and watch that his/her health doesn't fall too low. This system shows how well the character is doing.

Programming Languages Used

JavaScript - Dynamic Language for Web:

JavaScript is one of the most simple, omnipotent, and effective programming languages used in web pages. In the past, JavaScript was only used to do simple screen visual effects on websites such as floating advertising box and form validation. Since AJAX grew popular, people found that utilizing JavaScript can bring users a better interactive experience. This advantage has been used to develop online web games. JavaScript also has Object-Oriented programming features, which is good for game programming because objects are an import thing when it comes to games.

We chose JavaScript for a few reasons. First, JavaScript is a relatively easy language for us to learn. This advantage is very important because we need to navigate the other "alien" language in a very short amount of time. Second, the experimental process and analysis will be easier and clearer if everything including the inputs, outputs, and environment can be displayed in a visual way on the screen. JavaScript's dynamic nature perfectly satisfies this requirement. Third, object-based features make it easier to track the objects such as the main character in the game. Finally, JavaScript is functional like Haskell, the purely functional programming language.

Haskell - Purely Functional Programming Language:

Haskell is a purely Functional Programming Language designed by a group of smart scientists and named after a logician named Haskell Brooks Curry. The program was developed in the 1985 when there was an interest in lazy functional languages.

Although it is unconventional and difficult to make a game in Haskell, we decided to use Haskell as the second language because we wanted to experiment with a language that is not typically used for games. Using Haskell in game development is very rare, but Haskell does have incredible potential that allows us to explore more about how things work. This is a good opportunity for us to improve our Haskell knowledge by implementing something that we are interested in and also to familiarize ourselves with what makes up the necessary set of elements to build a game. Some possible advantages of using Haskell are its purity, the static type system, pattern matching, and interesting abstractions.

Process

Throughout the project, we implemented every feature in both languages simultaneously so that the implementation is as similar as possible between the languages. In some cases, it was a simple task, but in others, it seemed like certain features just couldn't be done in one of the languages. In some cases, doing the implementation in one language helped the implementation of the other language. As a result, both versions of the game run almost identical.

Compare and Contrast

The main difference between the two is the use of variables. In JavaScript, the use of global variables made everything relatively simple, but in Haskell, we had to pass a state around. The JavaScript implementation ended up being really simple and easy mostly because the language is very conventional for games because of its object-oriented nature. Haskell's implementation was really quite the challenge. However, the restrictiveness meant that it was harder to make mistakes and easier to control exactly what was going to happen in the program.

```haskell
printMap :: GameState -> Int -> Int -> IO ()
printMap gs w h = printMap' gs (-1) (-1) w h >> putStr "\n"
printMap' :: GameState -> Int -> Int -> Int -> Int -> IO ()
printMap' gs@(GameState (x, y) _ _ _ _ map) i j w h
    | j >= h && i > w  = putStr "\n" >> return ()
    | j == y && i == x = putStr "@"  >> printMap' gs (i+1) j w h
    | i > w            = putStr "\n" >> printMap' gs (-1) (j+1) w h
    | j < 0 || j >= h  = putStr "-"  >> printMap' gs (i+1) j w h
    | i < 0 || i == w  = putStr "|"  >> printMap' gs (i+1) j w h
    | v == True        = putStr [c]  >> printMap' gs (i+1) j w h
    | otherwise        = putStr " "  >> printMap' gs (i+1) j w h
    where (Cell c v) = (map!!j)!!i
```

Haskell

```javascript
function printMap(){
    var str = "";
    for(var x = 0; x < MAP_WIDTH + 2;  x++) str += '-';
    str += '\n';
    for(var y = 0; y < MAP_HEIGHT; y++){
        str += " |";
        for(var x = 0; x < MAP_WIDTH; x++){
            if(isPlayerOn(x,y)) str += "@";
            else if(Environment[y][x].visibility) str += Environment[y][x].type;
            else str += " ";
        }
        str += "|\n";
    }
    str += ' ';
    for(var x = 0; x < MAP_WIDTH + 2;  x++) str += '-';
    return str;
}
```

JavaScript

In terms of the code, JavaScript and Haskell ended up being fairly similar. The implementation of certain functions may have been slightly different, but the implementation of the program as a whole worked through the same pipeline.

Challenges

JavaScript:

Though JavaScript was relatively easy, there were still some difficulty in the implementation. Since JavaScript implicitly defines types, it was sometimes confusing as to what certain variables represented since everything was labeled "var" including functions. Also, JavaScript caught less errors, so the program would behave unpredictably sometimes and it would be really hard to trace the problem.

Haskell:

We faced much more challenges when we implemented the game in Haskell. First of all, a lot of time and effort is needed to learn about libraries. Though sometimes it helped, Haskell's restrictiveness was more of a curse than a blessing. Implementing randomness was a nightmare since pure functions cannot support randomness. Also, the use of a state was sometimes a burden. If a variable that was needed down a line of functions, it had to be passed down the whole line of functions just to be used in one place. Other than that, the lack of for-loops of somewhat of a problem even though it was solved with the less conventional recursive method.

Conclusion

Object-oriented programming is a very big deal when it comes to game programming, and thus JavaScript is a very ideal language for game programming. Haskell has the potential to do much more, but it's not tailored to making survival games because of the lack of objects. The functional features allowed us to accomplish interesting features, but JavaScript's functional features did the same just as well.

Future

The result of this project produced a pretty well made template for a survival text game since it has the bare basic features of a survival game. We do wish to continue the development of the game on the JavaScript end.