Visualización del mercado de acciones en 3D

Juan Esteban Pemberthy

Juan Sebastian Munoz

28 de abril de 2009

Índice general

Introducción

> *"Elegir acciones individuales sin una idea de lo que estás buscando es como atravesar una fábrica de dinamita con una cerilla encendida. Puedes sobrevivir, pero sigues siendo un idiota"*
> –Joel Greenblat[1]

En la bolsa de valores de Colombia existen cuatro mercados principales de operación, el trabajo de ésta tesis se enfoca única y exclusivamente a uno de estos mercados; El mercado de Renta Variable, donde se negocian las acciones de compañías inscritas en el mercado publico de valores.

Al tratarse de un mercado público cualquier persona esta en la libertad de comprar y/o vender acciones según lo desee, generalmente estas personas llevan cierto control sobre sus inversiones, ya sea de un modo sistematizado ó escribiendo en una agenda de apuntes sus movimientos, con esta información realizan un análisis de datos que le permite determinar cuándo una acción le genera ganancia y/o perdida que apoyen una decisión, como comprar o vender.

El análisis de datos siempre ha estado ligado a una representación gráfica de los mismos, y el mercado de acciones mencionado anteriormente no es la excepción, generalmente para representar éstos datos, los que describen el estado y movimiento en el tiempo de una acción se han utilizado gráficos en segunda dimensión como barras, tortas, entre otras.

Esta tesis introduce un nuevo paradigma de visualización en este campo, al pasar de gráficos en dos dimensiones, a una nueva propuesta de contenidos para visualizar los datos en tres dimensiones, permitiéndole a una persona visualizar de una manera innovadora el estado y movimiento de las acciones sobre las que lleve algún tipo de registro.

Para lograr éste trabajo, se desarrollo una aplicación web, en la que se integró satisfactoriamente un modelo de datos con un conjunto de contenidos 3D, sobre los cuales se estableció un protocolo de comunicación que permite el apropiado despliegue de gráficos, buscando que sea más fácil para un usuario

[1] *"The little book that beats the market"* por Joel Greenblat.

identificar el movimiento de una ó varias acciones.

La propuesta acá descrita es presentada como proyecto de grado para optar por el titulo de Ingeniería de Sistemas de la Universidad EAFIT.

Definición del problema

Durante la evolución de la computación moderna se han presenciado avances significativos en los diferentes campos que componen la misma, uno de estos campos es la visualización de aplicaciones, es decir la forma en la que una aplicación es desplegada para que una persona interactue con una máquina con el fin de cumplir una determinada tarea, el surgimiento de la computación gráfica como área de estudio le ha significado a la computación una nueva vía para desarrollar aplicaciones en las que la visualización juega un papel fundamental y que antes apenas si eran soñadas.

Con éste proyecto de grado se busca aprovechar la capacidad tecnológica actual para desarrollar contenidos en tercera dimensión que puedan ser visualizados desde cualquier lugar, el tema sobre el que se trabajó fue el movimiento del mercado de acciones Colombiano, se eligió este sector para brindarle a los propietarios de acciones que no están sumergidos de lleno en este mundo una herramienta que les permita identificar fácil y gráficamente el movimiento del mercado, pues actualmente muchas de estas personas llevan control de sus inversiones en un papel, o en un archivo plano, sobre el que construyen una tabla, y/o gráficas limitadas a dos dimensiones.

Las gráficas en segunda dimensión, son simples y fáciles de entender cuándo se están comparando acciones en una unidad de tiempo ó se esta viendo el movimiento de una sola acción en un período determinado, pero a la hora de comparar varias acciones en un período, una gráfica en 2D podría ser difícil de entender para un usuario común, mientras que una gráfica en 3D facilita el entendimiento de la misma, con el fin de apoyar el problema definido anteriormente, en esta aplicación se diseñaron un conjunto de contenidos en tercera dimensión que recrean la actividad del mercado, dichos contenidos son alimentados desde una base de datos que se actualiza constantemente, tanto datos como contenidos 3D fueron integrados en una aplicación web permitiendo que la aplicación sea visualizada desde cualquier lugar en el que se tenga acceso a Internet.

Capítulo 1

Objetivos del proyecto

1.1. Objetivo General

Brindar un mecanismo que permita visualizar y comparar el movimiento de una o varias acciones de la bolsa de valores de Colombia en el tiempo, mediante contenidos en tercera dimensión, incursionando en un campo sobre el cuál la mayoría de aplicaciones están diseñadas para desplegar sus resultados en 2D.

1.2. Objetivos Específicos

- Permitir llevar el control e historial de acciones de manera personalizada.
- Desarrollar un protocolo de comunicación que pueda ser reutilizado en aplicaciones futuras, que exploten el poder del motor gráfico para desarrollar contenidos alimentados desde una base de datos.
- Integrar el modelo de datos con la aplicación 3D para desplegar apropiadamente el movimiento del mercado y/o una determinada acción.

Capítulo 2

Alcance y productos esperados

Se entrega una aplicación Web desarrollada en **Ruby on Rails** que integra un conjunto de contenidos en 3D desarrollados en **Unity 3D** un motor gráfico que permite exportar los contenidos para que sean acoplados con la aplicación Web y consecuentemente se puedan visualizar desde un navegador. Además los contenidos se estarán actualizando constantemente en un período de tiempo razonable, a esto se le suma la posibilidad de un usuario registrarse y configurar su perfil, sobre este perfil cada usuario podrá simular o evidenciar según sea el caso (sí posee ó no acciones en la vida real) ganancias o pérdidas sobre las acciones que tenga registradas y la cantidad asociada a las mismas.

Se aclara de antemano que la aplicación no sirve como medio para comprar o vender acciones, en el perfil también se puede desplegar de forma personalizada el movimiento de las acciones que el usuario tenga registradas, permitiendo diferentes esquemas de visualización en los que por ejemplo se tendrá en cuenta información guardada en la base de datos de la aplicación, por ejemplo los periodos de tiempo en los que un usuario registró una compra o venta serán utilizados para ayudar al usuario a que se de una mejor idea del estado de su inversión.

Adicionalmente los clásicos esquemas en segunda dimensión que actualmente son utilizados para la representación del movimiento de acciones como lo son las barras, también fueron implementados en la aplicación, pero de tal manera que tales barras no muestren sólo el estado de una acción en varios periodos de tiempo, o varias acciones en un periodo de tiempo, sino ofrecer la oportunidad de ver varias acciones como se comportan en varios periodos de tiempo.

Capítulo 3

Objetivos

3.1. Objetivo General

3.2. Objetivos Específicos

- Entender que son los operadores disfijos.
- Implementar un lenguaje de programación prueba de concepto que simule operadores disfijos mediante el lenguaje de programación Haskell.
- Utilizar los operadores disfijos del lenguaje como mecanismo de extensibilidad para implementar un lenguaje con características objetuales. Dicha construcción se realizará en etapas.

Capítulo 4

Importancia del problema.

Muchos de los avances en el desarrollo de software han tenido su origen en el area de lenguajes de programación. Los desarrollos en lenguajes de programación han permitido:

1. Escribir software en forma portable gracias a la posibilidad de abstraer los detalles arquitectónicos de la plataforma para la cual se desarrolla el software.

2. Escribir software mas seguro, ocultando y restringiendo ciertas construcciones inseguras del lenguaje objeto.[1]

3. Reutilización de código, mediante la incorporación de abstracciones para la generalización de soluciones. [2]

De la misma manera, desarrollos en lenguajes de programación han permitido el desarrollo de lenguajes mas expresivos. Entendiendo por lenguaje expresivo es aquel que brinda construcciones sintácticas y abstracciones que permiten al programador resolver problemas en forma mas "elegante" [3].

La importancia de la elegancia en los lenguajes de programación además de tener ventajas en mantenibilidad, y legibilidad tambien parece ser pieza importante en la reducción de errores. Existen indicios [7] para afirmar que la densidad de errores por número de líneas no se ve alterada por la elección del lenguaje de programación. De esta forma, lenguajes que requieren menos líneas de código para resolver diferentes problemas (lenguajes expresivos) pueden ayudar a disminuir la cantidad de errores.

Brindar expresividad enfrenta al diseñador del lenguaje de programación al dilema de Cardelli [3]: Decidir entre dar una amplia y expresiva notación o

[1] Entendiendo lenguaje objeto como el lenguaje producido por el lenguaje de programación. Por ejemplo, **x86** es uno de los lenguajes objeto de **C**

[2] Un ejemplo de ésto es Polimorfismo. El lenguaje diseñado en esta tésis tiene características polimórficas. Más sobre esto en la sección 4.

[3] Elegante puede tener connotaciones subjetivas, sin embargo el concepto es formalizado en [4]

tener un *"core"* pequeño. Ambas propiedades son deseables en un lenguaje de programación. Los *"cores"* pequeños son mas fáciles de mantener y permiten que el lenguaje sea asimilado más fácil por los programadores. Existe un enfoque híbrido basado en lenguajes extensibles. Estos lenguajes parten de un *"core"* pequeño, pero permiten que el usuario defina su propia sintáxis.

Esfuerzos para construir lenguajes que puedan "crecer" [6] se han realizado anteriormente. Dentro de las técnicas utilizadas en este enfoque hibrido cabe mencionar las siguientes: *"Syntax Macros"* [2], *"Extensible Syntax"* [3], *"Conctypes"* [1].

Este trabajo complementa dichos esfuerzos mostrando como un subconjunto de operadores disfijos puede ser utilizado como mecanismo de extensibilidad en los lenguajes de programación.

Capítulo 5

Alcance y productos esperados

Se entrega una aplicación Web desarrollada en **Ruby on Rails** que integra un conjunto de contenidos en 3D desarrollados en **Unity 3D** un motor gráfico que permite exportar los contenidos para que sean acoplados con la aplicación Web y consecuentemente se puedan visualizar desde un navegador. Además los contenidos se estarán actualizando constantemente en un período de tiempo razonable, a esto se le suma la posibilidad de un usuario registrarse y configurar su perfil, sobre este perfil cada usuario podrá simular o evidenciar según sea el caso (sí posee ó no acciones en la vida real) ganancias o pérdidas sobre las acciones que tenga registradas y la cantidad asociada a las mismas.

Se aclara de antemano que la aplicación no sirve como medio para comprar o vender acciones, en el perfil también se puede desplegar de forma personalizada el movimiento de las acciones que el usuario tenga registradas, permitiendo diferentes esquemas de visualización en los que por ejemplo se tendrá en cuenta información guardada en la base de datos de la aplicación, por ejemplo los periodos de tiempo en los que un usuario registró una compra o venta serán utilizados para ayudar al usuario a que se de una mejor idea del estado de su inversión.

Adicionalmente los clásicos esquemas en segunda dimensión que actualmente son utilizados para la representación del movimiento de acciones como lo son las barras, también fueron implementados en la aplicación, pero de tal manera que tales barras no muestren sólo el estado de una acción en varios periodos de tiempo, o varias acciones en un periodo de tiempo, sino ofrecer la oportunidad de ver varias acciones como se comportan en varios periodos de tiempo.

Capítulo 6

Parsing, Análisis y Evaluación.

Esta sección explica los diferentes pasos en la implementación de nuestro lenguaje para soportar operadores permisivos.

6.1. Una visión general

Inicialmente

$$Texto \rightarrow_1 Tokens \rightarrow_2 (Declaraciones, Main)$$

6.2. Lexing y Parsing

El primer paso de parsing detecta las declaraciones de funciones y la función principal main. Esta es la gramática del lenguaje siguiendo la notación del capítulo 3.

⟨Program⟩→⟨Declarations⟩⟨Main⟩

⟨Declarations⟩→⟨Declarations⟩⟨Declaration⟩|ϵ

⟨Declaration⟩→⟨Infix⟩|⟨Prefix⟩|⟨Suffix⟩|⟨Closed⟩

⟨Infix⟩→⟨Init⟩⟨InfixKeyword⟩⟨Precedence⟩⟨Id⟩⟨Id⟩⟨Id⟩=⟨Definition⟩

⟨Precedence⟩→ϵ⟨Number⟩

⟨InfixKeyword⟩→**infixr**|**infixl**

⟨Init⟩→**let**|**let rec**

⟨Prefix⟩→⟨Init⟩⟨Id⟩⟨Ids⟩=⟨Definition⟩

⟨Suffix⟩→⟨Init⟩**suffix**⟨Ids⟩⟨Id⟩=⟨Definition⟩

⟨Closed⟩→⟨Init⟩**closed**IdIdsId=⟨Definition⟩

⟨Ids⟩→⟨Id⟩⟨Ids⟩|ϵ
⟨Definition⟩→⟨ExpresionTokens⟩
⟨ExprTokens⟩→⟨ExprToken⟩⟨ExprTokens⟩|ϵ
⟨ExprToken⟩→⟨Literal⟩|⟨Id⟩
⟨Literal⟩→⟨Number⟩|⟨String⟩
⟨Number⟩→⟨Digit⟩|⟨Digit⟩⟨Number⟩
⟨Digit⟩→**0**|**1**|**2**|**3**|**4**|**5**|**6**|**7**|**8**|**9**
⟨String⟩→**"**⟨Chars⟩**"**
⟨Chars⟩→⟨Char⟩⟨Chars⟩|ϵ
⟨Char⟩→Cualquier caracter
⟨Id⟩→⟨Char⟩|⟨Char⟩⟨Id⟩
⟨Main⟩→**main =** |⟨Definition⟩

Ahora bien. Para definir una expresión añadiremos notación adicional: Op_p^a Indica que el operador tiene precedencia p asociatividad a. Usaremos un entero para denotar p y $\{l, r\}$ son los posibles valores de a. l denota asociatividad por izquierda y r asociatividad por derecha.

$$Expr := Open\, Expr\, Closed$$

$$Expr := Expr_1 Infix_n^r\, Expr_2 | \forall_{Op_m^r \in Expr_1 or Expr_2} n \leq m$$

En español... Una expresión puede estar formada por un operador infijo operando 2 subexpresiones. siempre y cuando el operador tenga la menor precedencia. En la defición falta agregar el manejo de asociatividad (en el caso de $Infix_n^r$ se debe garantizar que no exista ningún operador con la misma precedencia y misma asociatividad en $Expr_2$).

$$Expr := Expr_1 Infix_n^l\, Expr_2 | \forall_{Op_m^j \in Expr_1 or Expr_2} n \leq m$$

$$Expr := PrefixOp\, Expr$$

$$Expr := Expr\, SuffixOp$$

$$Expr := Literal$$

$$Literal := string | natural$$

6.3. Resolución de operadores

Data: Lista de ExprTokens
Result: Arbol de evaluación
initialization;
if *understand* **then**
| go to next section;
end

Algoritmo 1: Resolución de precedencia

6.4. Sistema de tipos

El lenguaje de programación es fuertemente tipado y está construido a partir de la implementación encontrada en [5]. El sistema de tipos esta conformado por los siguientes constructores en Haskell.

```
data Type    =  TVar String
             |  TInt
             |  TBool
             |  TString
             |  TList Type
             |  TProd Type Type
             |  TFun Type Type
```

El sistema de tipos cuenta con variables de tipos, enteros, booleanos, cadenas de caracteres, listas genéricas, funciones y producto de tipos.

6.5. Construcción de notación aritmética

6.6.

Capítulo 7

Conclusiones

Bibliografía

[1] Anika Aasa. *User Defined Syntax*. PhD thesis, Chalmers University of Technology, 1992.

[2] Arthur Baars and Doaitse Swierstra. Syntax macros. Unfinished Draft.

[3] Luca Cardelli, Florian Matthes, and Martin Abadi. Extensible syntax with lexical scoping. Technical report, Systems Research Center, 1994.

[4] G. J. Chaitin. *People and Ideas in Theoretical Computer Science*, chapter Elegant LISP Programs. Springer-Verlag, 1999. Disponible en línea: `http://www.cs.auckland.ac.nz/CDMTCS//chaitin/lisp.html`.

[5] Martin Grabmüller. Algorithm w step by step. Draft paper, September 2007.

[6] Guy L. Steele, Jr. Growing a language. *Higher Order Symbol. Comput.*, 12(3):221–236, 1999.

[7] Ulf Wiger. Four-fold increase in productivity and quality. Technical report, Ericsson, 2001.