PSEUDOCÓDIGO EJECUTABLE PARA LA ENSEÑANZA DE LA ALGORITMIZACIÓN

Ponente: José Albert Cruz Almaguer
Número C.I: _80092018572_____
Teléfono: _837 2572_____
Provincia: _Holguín_____
Categoría docente: __Asistente__
Titulo académico o grado científico: __Licenciado en Ciencia de la Computación _____
Medios necesarios para la exposición: ( ) Retroproyector, (X) Proyector de diapositivas, ( ) Video, ( ) Cañón, (X) Computadora, ( ) Otros (especifique)
Taller en el que participará: __Taller Nacional sobre Ciencias de la Matemática y la Computación.__
Línea temática: ___Programación e Ingeniería de Software_____
Modalidad:
Póster electrónico: __X_____
Póster cartulina: _____
Autor: _José Albert Cruz Almaguer_____
Institución a la que pertenece: _Universidad de las Ciencias Informáticas_____
E-mail: __jalbert@uci.cu_____

Resumen

Enseñar a programar es una tarea difícil, conceptos como estado, instrucción, estructura de control, iteración y flujo de ejecución toman tiempo en ser asimilados por los estudiantes y en muchos casos se pasa a niveles de abstracción superiores (subprogramas, clases) sin tener firmes los cimientos. El presente trabajo pretende ser un apoyo en este sentido, a partir de lograr un pseudocódigo ejecutable con construcciones sintácticas en español, entrada/salida a través de interfaces gráficas y posibilidad de actuar sobre un ambiente en ejecución que presente varios objetos “vivos” con los que se puede interactuar (mandarles mensajes).

Además se describen las herramientas utilizadas, la arquitectura de la aplicación y la manera en que la generación de código (metaprogramación), la interoperabilidad entre lenguajes de que se dispone en la plataforma Java y las herramientas actuales de construcción de compiladores, permiten desarrollar sistemas que hasta hace poco tiempo eran difíciles de lograr con pocos recursos y en breve tiempo.

Introducción

En la Universidad de las Ciencias Informáticas se usan como medio de iniciar a los estudiantes en la programación a los lenguajes C++, Java y C#, cualquiera de estos lenguajes exigen manejar símbolos de constituyen en sí mismos un problema: **static**, **void**, *main* son sólo algunos ejemplos; especialistas de renombre como [4] se han referido a la importancia del lenguaje que se use en los primeros momentos de la enseñanza y han coincidido en que pueden afectar el proceso.

Autores como los de [7] han considerado necesario el diseño de lenguajes específicos con el único objetivo de enseñar los conceptos básicos de programación, lenguajes con una sintaxis simple que permita centrarse en el problema a resolver y dejar a un lado detalles de otro tipo.

Otro factor a tener en cuanta es el idioma de las palabras reservadas, todos los lenguajes usados profesionalmente utilizan el Inglés como idioma base, si bien este es importante y no estaría bien pretender desconocerlo, en el caso del estudiantado que comienza las carreras de informática el dominio de este idioma no es el mejor y por lo tanto pensar soluciones usando estos términos añade complejidad a los problemas. Consideraciones de este tipo han sido tenidas en cuenta en la implementación del lenguaje Lexico[1] (**L**enguaje **EX**perimental **I**ntroductorio a la **C**omputación con **O**bjetos) usado con éxito en universidades colombianas [3].

La enseñanza de la algoritmia a personas sin conocimiento previo del tema requiere de la mayor simplicidad en el modo de expresión que se utilice, es por ello que en muchos contextos se utiliza alguna variante de pseudocódigo como medio de codificación de las soluciones. Por otra parte, dado el hecho de que un problema puede ser resuelto de muchas formas y que los alumnos son numerosos y heterogéneos en sus conocimientos, se hace imposible en la práctica el seguimiento de todas las soluciones planteadas por ellos. Esta es la principal motivación a la realización de este trabajo, una experiencia similar se encuentra en el PSeInt[2].

Algunas particularidades de la programación

Un curso de programación se enfrenta al reto de enseñar procesos, aprender a programar es despertar la habilidad intelectual de la previsión, es generar una actitud de anticipar permanentemente las consecuencias de las instrucciones dadas para que se desarrolle en el tiempo un proceso.

Diseñar procesos en detalle implica poner en concordancia conceptos, requisitos, consideraciones iniciales, etc. Esas características exigen gran coordinación para lograr plasmarlas en un artefacto que operará independientemente de su creador.

---

[1] http://riosur.net

[2] http://pseint.sourceforge.net/

Programar actuando sobre mundos virtuales

Dadas todas estas características de la programación de computadoras como actividad intelectual, se hace importante el uso de medios que sean estimulantes, motivadores… y que sobre todo, le den al estudiante la posibilidad de ver reflejadas en algún lugar las consecuencias de cada línea de código que escriba.

La inclusión de multimedios para estimular el aprendizaje se encuentra entre las variantes más utilizadas, es común en los juegos didácticos en computadoras la inclusión de vídeo y sonido para estimular varios sentidos a la vez y facilitar la adquisición del conocimiento. En el ámbito de la enseñanza de la programación a novicios se ha empleado la animación (el movimiento) como aliado para, al lograr realizar ejercicios vistosos, estimular el aprendizaje. Casos exitosos de esto lo encontramos en Traffic [3] (como desarrollo de las ideas de [5]) y Guido van Robot[4].

Desarrollo

A partir del anterior estudio y análisis se concluyó que era conveniente el desarrollo de un lenguaje de pseudocódigo que facilitara la E/S y pusiera al alcance del usuario una serie de objetos con los que se pudiera interactuar y cuyas respuestas tuvieran una connotación gráfica, que cada cambio en sus respectivos estados se viera reflejado en una interfaz.

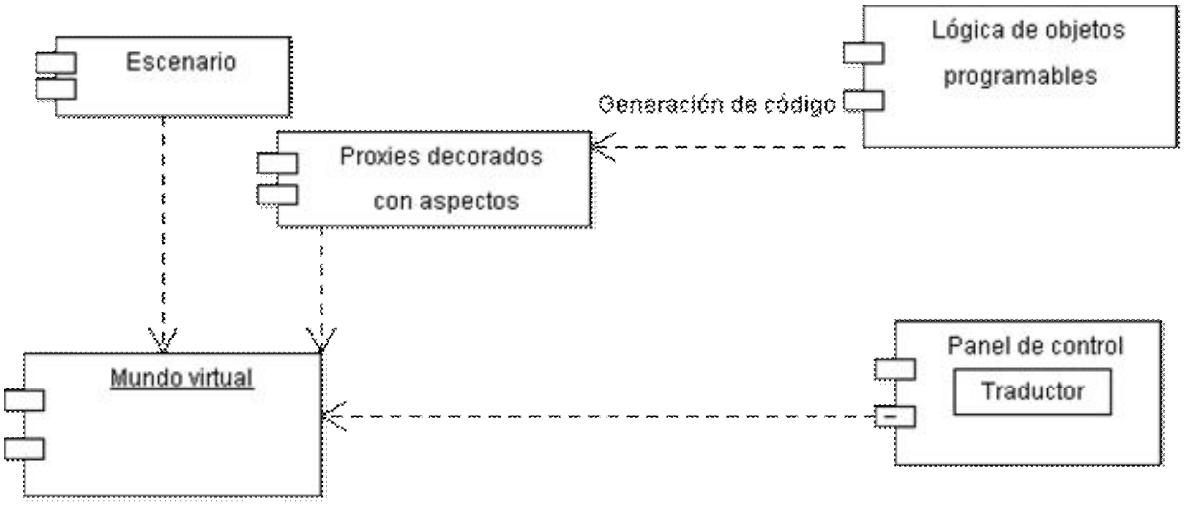

Figura 1. Arquitectura de la aplicación

[3] http://traffic.origo.ethz.ch/
[4] http://gvr.sourceforge.net/

Descripción de la arquitectura

Escenario

El escenario en que se desenvuelven los objetos está formado por componentes Swing encargados de visualizar el estado de cada objeto, consiste en un espacio coordenado en el que están los objetos.

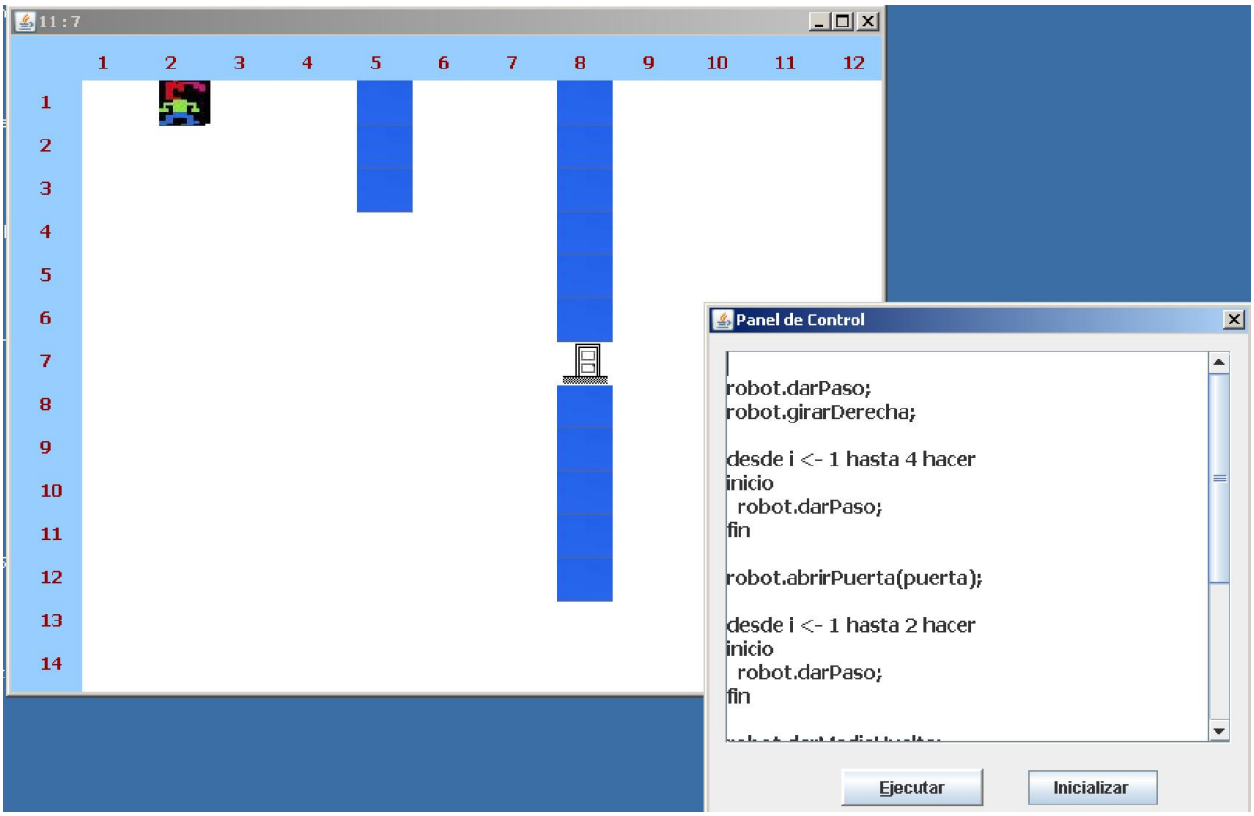

Figura 2. Componentes visibles: el entorno y la interfaz de control.

Formado por dos ventanas: una que refleja el mundo virtual en el que se desenvuelven los distintos objetos, la otra, encargada de la entrada de las nuevas acciones que se le programen a los mismos. Tras cada orden "*Ejecutar*" pasan a obedecerse las instrucciones editadas en la segunda ventana en el contexto de la primera.

El pseudocódigo

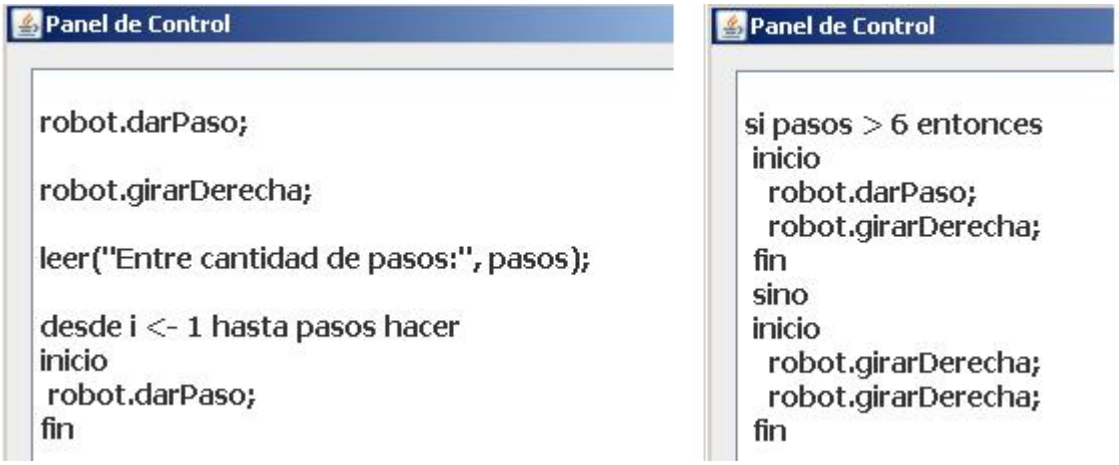

Figura 3. Fragmentos del lenguaje desarrollado.

El lenguaje de pseudocódigo incluye estructuras de control iterativas y de selección, el uso del símbolo <- para realizar la asignación así como operadores aritméticos y lógicos para realizar las operaciones comunes. Además posee rutinas de entrada/salida que despliegan ventanas para que se puedan realizar dichas operaciones en un ambiente con una vistosidad mayor a cuando se hace por línea de comandos, todo esto con el objetivo de motivar a los estudiantes durante el proceso de ejecución de sus programas.

Figura 4. Ventana de entrada de datos consecuencia de la ejecución de la función *leer*

Objetos disponibles en el ambiente

Los objetos a usar por el usuario mediante el lenguaje declaran sus contratos fundamentales mediante interfaces.

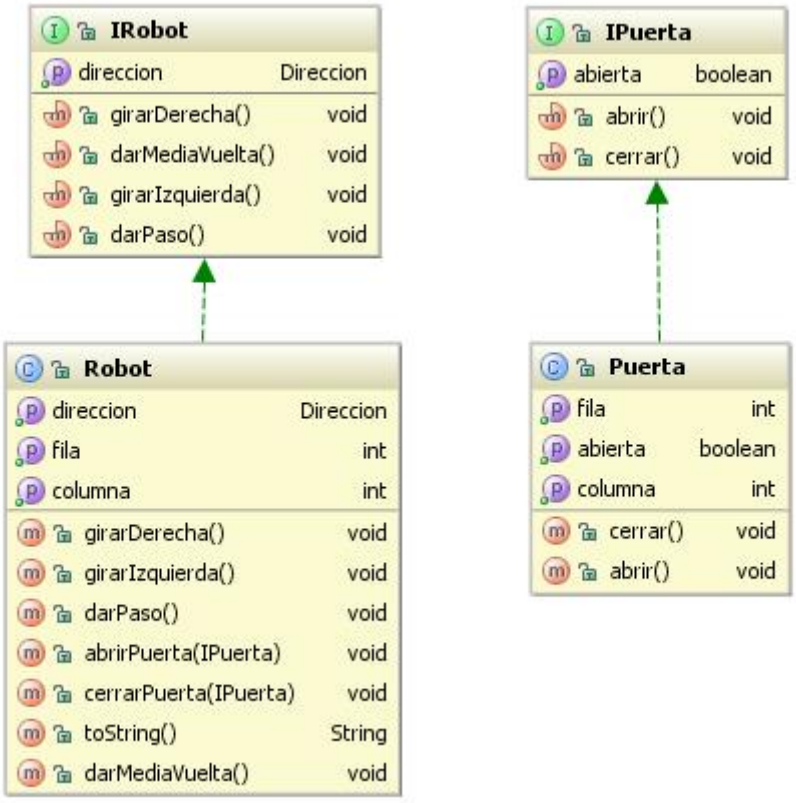

Figura 5. Objetos usados en el ejemplo anterior

Constituyen los entes a los que tendrá acceso el usuario para mandarles mensajes y ver cómo reaccionan a los mismos, cada uno tiene un objeto gráfico asociado en el escenario y la lógica que les permite reaccionar adecuadamente en el mundo virtual en el que se desenvuelven.

Mediante anotaciones se especifica: la precondición (@Require) necesaria para la ejecución de cada método, la categoría del método (@Command o @Query que significan: métodos que transforman el estado del objeto o que sólo lo consultan) utilizada para la sincronización del estado gráfico de cada objeto respecto al estado lógico; el siguiente fragmento de código, tomado de la implementación de la clase Robot muestra esto:

```
public class Robot implements IRobot {

    @Require("@context['tablero'].canAccess(self, direccion)")
    @Command
    public void darPaso() {...}

    @Query
    public int getFila() {...}
```

Figura 6. Fragmento de la clase Robot.

Herramientas utilizadas

ANTLR

El lenguaje de pseudocódigo fue desarrollado con la herramienta ANTLR, la misma es capaz de producir los analizadores léxico y sintáctico de una manera sencilla y unificada en varios lenguajes de programación (C, Java, C#, Python entre otros); para el diseño, implementación y prueba de la gramática se contó con el asistente ANTLRWorks, el cual posee avanzadas características para el desarrollo de esta actividad. El código generado consistió en un traductor que mapea las instrucciones del pseudocódigo en instrucciones Ruby válidas, listas a ser ejecutadas por el intérprete JRuby.

JRuby

En los últimos tiempos el lenguaje Ruby ha ganado muchos adeptos debido a la claridad de su sintáxis y, sobre todo, al enorme éxito alcanzado por el framework Ruby On Rails (para desarrollo Web), paralela a la implementación estándar en C del mismo se han hecho otras, JRuby es la versión hecha en Java que tiene completa interoperabilidad con la plataforma, en el caso de este proyecto fue utilizada para generar en Ruby los proxies encargados de unir la capa lógica con la de presentación de los objetos y, sobre todo, para utilizando su naturaleza dinámica poder ejecutar código de programas en el contexto de objetos existentes en ejecución, elemento clave para alcanzar nuestro objetivo.

Java/Swing

El escenario y la lógica de los objetos fueron desarrollados en Java usando los componentes Swing estándares, en la integración de las diferentes tecnologías usadas en el proyecto se hizo intensa utilización de las características del JDK 6.0 relativas al uso de lenguajes script en Java, de las posibilidades de reflection y de las anotaciones.

Conclusiones

En este trabajo se ha presentado una aplicación que posibilita niveles de interacción entre los estudiantes y las máquinas que estimula el aprendizaje, el lenguaje de pseudocódigo desarrollado puede servir de base al diseño de cursos de algoritmización que le brinden a los estudiantes las habilidades básicas que se necesitan para construir software. Además se describió su estructura y la manera en que puede usarse para dar solución a diferentes ejercicios.

En los últimos años el país ha dirigido ingentes esfuerzos hacia convertir a la industria del software en un baluarte de la economía, lograr altos niveles de asimilación de los contenidos de programación en nuestras universidades juega un importante papel para alcanzar este objetivo. Sirva este trabajo al noble empeño de formar a los nuevos profesionales del área.

Recomendaciones

Esta aplicación puede ser extendida con nuevos escenarios y el lenguaje proveído con facilidades para la edición de los programas y para la corrección de los errores que los estudiantes cometan.

Bibliografía

[1] M.E. Caspersen and M. Kölling. *A Novice's Process of Object-Oriented Programming. Companion to the 21st ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications,* páginas 892-900, 2006.

[2] Fabián Ríos Castrillón. *Lexico en programación orientada a objetos*, marzo 2009.

[3] Sandra Maria Morales M. *EVALUACIÓN DE LAS HERRAMIENTAS DE SOFTWARE LEXICO Y CSHARP COMO ALTERNATIVAS EN LA ENSEÑANZA Y EL APRENDIZAJE DE LA PROGRAMACIÓN ORIENTADA A OBJETOS (POO).* 2006

[4] Bertrand Meyer. *Object-Oriented Software Construction*, capítulo 29, páginas 938-940. Prentice Hall PTR, 2nd edition, 1997.

[5] Bertrand Meyer. *TOUCH OF CLASS*. Springer-Verlag, 2009.

[6] Terence Parr. *The Definitive ANTLR Reference. Building Domain-Specific Languages*. 2007.

[7] *Rosana Satorre et al. Un modo de entender la programación. X Jornadas de Enseñanza Universaria de la Informática, Jenui 2004*