Tutorial gráfico interactivo de estructuras de datos

Mariano J. Cabrero Canosa, Javier Blanco Gutiérrez
Departamento de Computación, Universidade da Coruña
Campus de Elvira s/n
15071 A Coruña
cicanosa@udc.es

Resumen

En este artículo se presenta un sistema tutor interactivo de estructuras de datos. El objetivo es mostrar mediante animaciones gráficas el comportamiento de las operaciones básicas sobre las estructuras de datos clásicas, proporcionando al usuario un control total sobre la ejecución de los algoritmos, no sólo mostrándole paso a paso el efecto de las distintas operaciones sobre la estructura, sino permitiéndole cambiar las condiciones de ejecución. Por otro lado, proveerá de diversos controles sobre la reproducción de la animación, permitiendo la ejecución paso a paso, el avance, el retroceso o la parada. Para facilitar el acceso a la herramienta y ampliar el rango de usuarios potenciales se ha implementado de manera independiente a la plataforma hardware y software, siendo accesible vía web a través de un sencillo applet. En su desarrollo se ha utilizado un sistema de animación basado en Java, JAWAA, que proporciona un lenguaje simple de comandos para la creación de animaciones y su visualización a través de un navegador.

1. Introducción

La animación de algoritmos es una forma de visualización de software que consiste en utilizar elementos como el movimiento, el color, la forma y el sonido con el propósito de generar representaciones gráficas dinámicas asociadas a conceptos abstractos que definen el comportamiento de un algoritmo dado [1].

Supongamos la implementación de un conjunto prefijado de operaciones para manipular una estructura de datos. La animación permite que, a través de una sencilla representación gráfica de la estructura, sea posible mostrar de una forma didáctica y dinámica los cambios ocurridos como resultado de la ejecución de alguna de dichas operaciones. Permite ver, por tanto, la ejecución paso a paso de la operación, explorando los detalles subyacentes del algoritmo involucrado para obtener una mejor comprensión de su lógica.

Desde el punto de vista docente, la animación de algoritmos supone un apoyo importante en el aprendizaje de los conceptos básicos de la programación, puesto que permite al estudiante interactuar con el código y verificar de manera gráfica el resultado de la ejecución. Este aspecto es muy importante si tratamos de reorientar el estilo clásico de enseñanza centrado en el profesor a un estilo centrado en el alumno.

La utilización de la animación en la enseñanza de algoritmos no es nueva; de hecho comienza hace dos décadas, cuando el profesor Ronald Baecker de la Universidad de Toronto creó el video *Sorting Out Sorting*. Los primeros sistemas de animación surgen entre 1980 y 1990: Estos sistemas fueron denominados BALSA-I [2] (Brow ALgorithm Simulator and Animator) y TANGO [3] (Transition-based ANimation GeneratiOn). El primero de ellos, soportaba múltiples vistas simultáneas de las estructuras de datos y permitía visualizar la ejecución simultánea de distintos algoritmos. Sirvió como acicate a otros investigadores para participar en el desarrollo de otros sistemas. El segundo de ellos, desarrollado por John Stasko en la Universidad de Brown, supuso la introducción del paradigma "path-transition" en el diseño de la animación: se abstraen las instrucciones de dibujo para permitir un modelado más cómodo de las animaciones de los algoritmos. Su aportación más importante fue el establecimiento de un marco conceptual para los sistemas de animación de algoritmos, desde entonces adoptado por muchos como arquitectura fundamental.

A BALSA-I le siguió, BALSA-II, un sistema que manipula imágenes con múltiples vistas y provee una facilidad de "scripting" para la creación de animaciones. TANGO, por otro lado, tiene numerosos sistemas descendientes, como POLKA y su "front-end", Samba [4]. POLKA es un sistema multipropósito especialmente diseñado para construir animaciones concurrentes de programas en paralelo. Inicialmente en 2D orientado a objetos, ha sido extendido a 3D, Samba es un intérprete interactivo que lee

comandos ASCII y lleva a cabo las correspondientes acciones de animación. Completan la nómina de programas otras contribuciones como Zeus [5], primer sistema de visualización interactivo que permite animar programas paralelos, y Leonardo [6], entorno integrado para el desarrollo y animación de programas en C, que proporciona un extenso control del usuario sobre la animación.

Frente a los anteriores, con la llegada de las nuevas tecnologías, la popularidad de Internet y la evolución del lenguaje de programación Java, los desarrolladores han pasado a construir sistemas de animación de algoritmos online, independientes de la plataforma, y de código abierto, no limitados en su uso al aprendizaje en aulas o laboratorios sino extensivos al aprendizaje a distancia. Catai [7] es un sistema de animación que anima programas en C++ y permite a varios usuarios compartir la misma animación a través de la abstracción de un aula virtual. Mocha [8] es un modelo distribuido con una arquitectura cliente-servidor que particiona de un modo óptimo los componentes software de un típico sistema de animación de algoritmos. Sólo se exporta a la máquina del usuario el código para la interfaz gráfica, mientras que el algoritmo se ejecuta en el servidor. Más reciente y completo es EDApplets [9], un conjunto de Applets Java que realizan animaciones clásicas de estructuras de datos y algoritmos, efectuando trazas y ejecuciones paso a paso sobre el código, y permitiendo interactuar al alumno.

La herramienta que se presenta en este artículo integra bajo un mismo Applet distintas animaciones de estructuras de datos abstractas como las listas, pilas, colas y árboles binarios, todos ellos con implementación dinámica. El alumno tiene a su disposición un conjunto completo de operaciones para el manejo de la estructura: creación y destrucción e inserción, eliminación y búsqueda de datos. Puede interactuar totalmente con las operaciones, modificando su comportamiento a través de los datos de entrada. En caso de error, la aplicación informa del motivo, bien por introducir incorrectamente alguno de los parámetros, bien por aplicar incorrectamente la operación. El diseño de la herramienta permite, a través de un lenguaje simple, crear y animar fácilmente elementos gráficos y sincronizarlos con la sentencia de código que se ejecuta, permitiendo de esta forma su extensibilidad.

Los estudios empíricos llevados a cabo sobre el hipotético beneficio pedagógico de la utilización de la animación de algoritmos en la docencia han arrojado resultados favorables, si bien en todos se destaca la necesidad de combinarla con otros materiales de aprendizaje. A modo de resumen, éstas serían las conclusiones más sobresalientes:

- Brown y Sedgewick [2]: la animación como complemento a la clase tradicional reduce el tiempo necesario para la comprensión.
- Lawrence [10]: permitir a los estudiantes controlar e interactuar con la animación y disponer de ella fuera de las aulas, de manera que utilicen sus propios conjuntos de datos para el algoritmo en lugar de observar conjuntos de datos preparados, contribuye a un mayor entendimiento.
- Kehoe [11]: las animaciones de algoritmos adquieren mayor efectividad si se usan en situaciones de aprendizaje abiertas e interactivas (como los ejercicios de clase). La comprensión del algoritmo se hace más accesible y menos intimidante.

En resumen, la literatura parece apoyar la utilización de estas técnicas como apoyo docente en lo que respecta al aprendizaje de algoritmos y estructuras de datos.

2. Contexto

La materia Estructura de Datos y de la Información (EDI) es una asignatura troncal cuatrimestral del plan de estudios de las Ingenierías Técnicas en Informática de Gestión y Sistemas y de la Ingeniería Informática de la Facultad de Informática de A Coruña. La asignatura se imparte en el primer curso del segundo cuatrimestre y tiene asignados un total de 6 créditos, 4.5 de teoría y 1.5 de prácticas, lo que se traduce en 3 horas teóricas y 1 hora práctica semanal. Los descriptores propios son:

- Estructura de datos y algoritmo de manipulación.
- Tipos abstractos de datos

Aunque no existe ningún requisito previo normativo para poder cursar esta materia, se recomienda tener conocimientos básicos de la

asignatura Programación, cursada en el cuatrimestre previo.

Actualmente, las técnicas en el campo de las estructura de datos van en la línea de diferenciar las "especificaciones" de una estructura de datos, es decir, su comportamiento, de sus "implementaciones", en el ámbito de las disponibilidades de un lenguaje de programación específico. La clave, tanto en el diseño de estructuras como de algoritmos que las manipulen es el concepto de "abstracción". Utilizando el concepto de abstracción, podemos definir las diferentes estructuras "olvidando" los detalles de implementación de los objetos, tales como su representación en la memoria o los algoritmos de manejo, concentrándonos en especificar sus valores y sus operaciones permitidas, que es lo que se conoce como "Tipos Abstractos de Datos".

Dentro del plan de estudios mencionado, la asignatura se podría englobar en el bloque temático de asignaturas relacionadas con la Ingeniería del Software, dentro del cual esta asignatura constituye una disciplina totalmente básica. En este grupo, las relaciones más estrechas se establecen con Bases de Datos I, Algoritmos, Metodología de la Programación y Programación Orientada a Objetos.

2.1. Objetivos generales de EDI

- Conocer los mecanismos de abstracción y valorar el papel importante que juega la abstracción en general, y la de datos en particular, como herramienta fundamental en la metodología de construcción de programas.
- Comprender el concepto de Tipo Abstracto de Datos (TAD).
- Comprender la necesidad de separación entre los niveles de especificación, implementación y uso de un TAD.
- Conocer, desde el punto de vista de la abstracción de datos, las estructuras de datos básicas que se utilizan en programación, sus realizaciones más comunes, sus algoritmos de manipulación y su utilidad en aplicaciones básicas realistas.
- Familiarizarse con el manejo de estructuras de datos básicas y desarrollar la capacidad de crear otras más complejas usando las primeras como referencia.
- Analizar la estructura de datos apropiada para la resolución de un problema y valorar la solución en cuanto a eficiencia y coste.
- Desarrollar la habilidad para definir y analizar alternativas de representación.
- Desarrollar implementaciones de las estructuras estudiadas utilizando un lenguaje de programación imperativo de alto nivel.
- Valorar como buenas prácticas de programación el desarrollo de código estructurado, modular y bien documentado.

2.2. Contenidos teóricos

El temario de teoría que se imparte en la asignatura es el siguiente:

1. Gestión dinámica de la memoria
2. Recursividad
3. Introducción a los TAD
4. Estructuras de datos lineales: listas, pilas y colas
5. Estructuras de datos no lineales: árboles binarios y árboles de búsqueda

Para el desarrollo de las clases teóricas se sigue un patrón expositivo magistral de descripción de contenidos temáticos, implementando en un lenguaje de programación como PASCAL las operaciones básicas de manejo de las distintas estructuras de datos. En estas clases presenciales el profesor involucra en el desarrollo práctico a los alumnos, que colaboran por grupos en el proceso de solución. Además, se plantean distintos ejercicios de aplicación de estructuras de datos a problemas reales. En suma, la orientación de la asignatura es fundamentalmente práctica, lo cual también se refleja en el proceso de evaluación del alumno.

2.3. Contenidos prácticos

Para evaluar la capacidad del alumno en la aplicación práctica de tipos abstractos de datos a problemas reales se plantean dos supuestos prácticos obligatorios que los alumnos deben resolver por parejas. El profesor supervisará el trabajo durante las distintas sesiones, solucionando dudas y corrigiendo errores de interpretación, malos hábitos de programación, etc. Los objetivos docentes a alcanzar a través del programa de prácticas se resumen en:

1. Aumentar el dominio técnico en la tarea de programación.

2. Consolidación de los conceptos de abstracción a través de la resolución práctica de supuestos reales.
3. Mejorar la capacidad analítica para seleccionar la estructura de datos adecuada en la solución software adoptada, valorando y criticando las distintas aproximaciones.
4. Desarrollar y/o potenciar aptitudes de trabajo en equipo.

2.4. Evaluación

La evaluación de alumno tiene en cuenta la valoración de las prácticas obligatorias y la realización de un examen final escrito al término del cuatrimestre, de carácter eminentemente práctico para que pueda demostrar que ha adquirido los conocimientos necesarios de abstracción y diseño de TADs y se ha entrenado lo suficiente como para poseer las habilidades precisas para resolver supuestos. La nota final tiene en cuenta en un 90% la nota del examen escrito y en un 10% la nota de las prácticas obligatorias.

3. Motivación

Cualquier plan de estudios de informática contempla asignaturas relacionadas con estructuras de datos y algoritmos de manipulación. El estudiantado novel debe adquirir en poco tiempo capacidades conceptuales de algoritmos y estructuras de datos, capacidades procedimentales para especificar, diseñar y programar estructuras de datos y ser capaces de desarrollar el pensamiento creativo para resolver problemas reales. Así pues, el alumno se encuentra con un proceso abstracto y dinámico, como es un algoritmo, difícil de comprender aún cuando posean el seudocódigo que describa la lógica interna y consulten la bibliografía seleccionada.

El proceso instructivo tradicional ha sido siempre unidireccional: el docente utiliza distintos recursos (pizarra, transparencias,…) para presentar la materia y la única interacción con el estudiante se limita a discusiones verbales. En muchos casos, el profesor se limita a escribir fragmentos de código e ilustrar gráficamente el modo en que dichos fragmentos se comportan, pero no es posible reflejar el comportamiento de las distintas operaciones y los cambios que se producen en las estructuras de datos en virtud de su ejecución. Esto lo proporciona la animación de algoritmos, una combinación de texto y gráficos con la cual el estudiante puede experimentar la lógica interna del algoritmo en el laboratorio y por extensión fuera del aula, fijándose su propio tiempo de estudio. Con este tipo de recurso docente se pueden alcanzar valores pedagógicos óptimos, siempre y cuando el sistema de animación posea las siguientes características:

1. Usabilidad, referida a la capacidad del sistema para ser comprendido, aprendido, usado y ser atractivo para el usuario de modo que la curva de aprendizaje sea mínima.
2. Multipropósito, en el sentido de no limitarse exclusivamente a la animación de algoritmos y estructuras de datos.
3. Acceso abierto al sistema en el centro educativo, o a través de Internet.
4. Independencia de la plataforma.
5. Resaltado de sintaxis del código fuente, destacando de alguna forma las líneas del algoritmo que se ejecutan a la vez que la transcurre la animación.
6. Control de la animación, avanzando paso a paso, deshaciendo, retrocediendo o avanzando rápidamente hacia cualquier estado deseado.
7. Interactividad entre los usuarios y el sistema.
8. Historia para habilitar el acceso a estados previos de la animación.
9. Retroalimentación de los estudiantes para evaluar la efectividad del sistema con el fin de mejorarlo.

4. Requisitos funcionales del sistema

La siguiente lista muestra los requisitos funcionales que la herramienta debería de satisfacer:

- Mantener información acerca de la codificación de las estructuras y de las operaciones sobre las estructuras de datos (codificación)
- Detectar condiciones de error en operaciones
- Procesar las operaciones y generar la animación de la estructura correspondiente
- Controlar la velocidad de la animación
- Resaltar líneas de los algoritmos durante la animación
- Animar los nodos de las estructuras
- Mantener apuntadores gráficos entre los nodos

- Disponer variables auxiliares usadas en los algoritmos en la pantalla
- Mantener el estado interno de las estructuras de datos que se animan
- Detectar entradas de datos anómalos
- Soportar internacionalización

5. Arquitectura del sistema

En 1990, John Stasko publicó un informe en el que introdujo un nuevo modelo para desarrollar los sistemas de animación de algoritmos [3]. "La animación de algoritmos abarca la animación del programa y la estructura de datos que interviene, lo que implica típicamente una relación uno a uno entre los datos del programa y las imágenes de la animación…Con frecuencia, las imágenes de la animación no tienen ninguna correspondencia directa a los datos del programa o a las unidades de la ejecución, sino que por el contrario representan las abstracciones diseñadas para aclarar la semántica del programa…Los diseñadores de la animación no deberían tener que crear una rutina distinta de animación para cada configuración posible del programa. Por lo tanto, la animación de algoritmos requiere modelos que soporten la creación de rutinas de animación que puedan adaptarse a varios estados del programa." De este modo, su informe manifiesta la imperiosa necesidad de destruir el acoplamiento entre las rutinas que modelan la animación y el programa que se quiere animar.

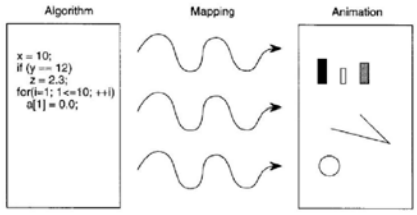

Figura 1. Marco conceptual definido por Stasko

El objetivo del marco conceptual es desarrollar un sistema limpio, de gran alcance y flexible. La arquitectura propuesta consta de tres componentes como se muestra en la Figura 1: algoritmo, animación y "mappings". El componente algoritmo sirve para conducir o activar la animación usando un enfoque conducido por eventos: se identifican las posiciones en el código que son importantes en la semántica del programa, y se modelan como llamadas a funciones con parámetros; a medida que se van alcanzando dichos puntos durante la ejecución del programa, su nombre y parámetros se pasan al componente de animación. El componente de animación contiene los objetos gráficos que cambian la posición, el tamaño, y el color en la animación, y las operaciones que controlan los cambios para simular una acción. El componente "mappings" contiene la traducción del programa y de su ejecución en la animación. Una porción de este componente implica la construcción de un mecanismo general de almacenamiento y otro de recuperación de objetos, que permite a los diseñadores conectar un objeto de datos que participa en la animación con un conjunto de parámetros recibidos del programa.

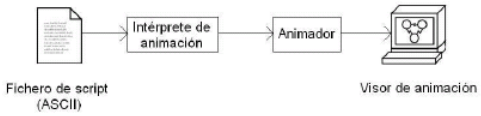

Figura 2. Arquitectura de los sistemas de animación de algoritmos

Desde la publicación del citado informe, muchos sistemas de animación de algoritmos han adoptado el marco conceptual propuesto por Stasko. La arquitectura genérica constaría de tres componentes; un animador, un intérprete de animación y un visor de la animación (Figura 2). El animador contiene librerías con funciones de representación gráfica y animación de los objetos, en los dispositivos gráficos. El visor de la animación provee un entorno gráfico para mostrar la animación a los usuarios. El intérprete de animación actúa como un canal de comunicación entre el sistema de animación y el usuario final. Las animaciones se modelan a través de comandos de script, insertados en los puntos clave del programa que se desea animar. El intérprete traduce estos comandos y pasa los parámetros de control del objeto que se quiere animar al animador. Finalmente el animador dibuja el objeto animado de acuerdo a los parámetros de control en el visor de animación. Estos parámetros son las coordenadas x e y donde aparecerá el objeto, su color, forma, etc.

Esta arquitectura fue adaptada y ampliada para construir el sistema de animación interactivo. A

partir de sistema de animación existente como JAWAA [12], se planteó un modelo como el de la Figura 3, compuesto fundamentalmente por un animador de objetos complejos, un generador de animaciones de algoritmos, una interfaz gráfica de usuario y el subsistema de animación JAWAA.

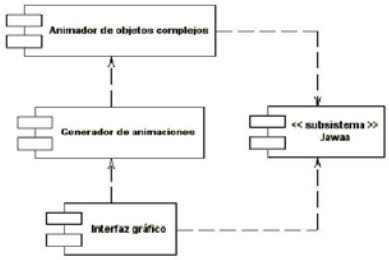

Figura 3. Arquitectura del sistema tutor

5.1. Animador de objetos complejos

El animador de objetos complejos contiene básicamente dos tipos de componentes: los objetos gráficos que intervienen en la animación y la coreografía de la misma.

Los objetos gráficos son aquellos que experimentan cambios en la localización, tamaño, color, etc. a través de los "frames" de la animación. Así pues, se han desarrollado en este componente los objetos que representan los nodos de las estructuras, los enlaces entre ellos y el algoritmo que se intenta animar (operación de manejo de la estructura de datos). Todos ellos están compuestos a su vez de otros objetos gráficos más simples que ofrece el subsistema JAWAA. Diversas operaciones estándar permiten cambiar el color, el tamaño, la posición, de los objetos gráficos, etc. A mayores se diseñaron operaciones para establecer apuntadores entre nodos, destacar la línea de código del algoritmo que se ejecuta, etc.

La coreografía guarda los cambios que se deben producir en los atributos de los objetos gráficos del subsistema JAWAA entre un "frame" y otro, información acerca de las pausas en la animación y de la agrupación de efectos animados sobre los objetos gráficos que se deben realizar en cada paso de animación.

5.2. Generador de animaciones

El generador de animaciones está compuesto por un conjunto de clases Coreógrafo que permiten modelar las animaciones correspondientes a la ejecución de los algoritmos sobre las estructuras de datos. Antes de modelar la animación se encarga de detectar condiciones de error debido a la introducción incorrecta de parámetros o a la violación de las precondiciones del algoritmo. Si la ejecución es exitosa añade los objetos gráficos a la primera secuencia de animación e invoca operaciones sobre ellos para conducir y transformar los objetos gráficos a través de las secuencias de animación posteriores. Además necesita mantener el estado de las estructuras de datos para lograr que las animaciones sean dinámicas y que los usuarios obtengan animaciones diferentes tras la ejecución de los algoritmos. También se encarga de proporcionar información acerca de la implementación de los algoritmos y las estructuras de datos.

5.3. Subsistema JAWAA

El subsistema JAWAA se encarga de proveer la visualización multimedia de la animación. Para lograr dicho objetivo, interpretará las secuencias de animación especificadas por el componente Animador de objetos complejos.

5.4. Interfaz gráfica

El componente interfaz gráfica es el responsable de gestionar, por un lado, la interacción entre el usuario y el componente generador de animaciones, y por otro, la interacción entre el usuario y el subsistema JAWAA. Así pues permite la construcción de la animación correspondiente a la ejecución del algoritmo seleccionado por el usuario, mostrando las condiciones de error que se producen, si procede, y activando la animación si la ejecución es satisfactoria. Además, se encarga de gestionar los eventos que producen llamadas desde/hacia el subsistema JAWAA para reproducir, parar, pausar, reanudar, aumentar o disminuir la velocidad y visualizar mediante pequeños pasos la reproducción de la animación.

6. Implementación

El sistema tutor interactivo está programado en Java porque: a) está especialmente diseñado para programar en Internet y permite transmitir, mediante el protocolo HTTP, programas autoejecutables e independientes de la plataforma;

b) verifica su código al mismo tiempo que se escribe, y una vez más antes de ejecutarse; c) hace posible la incorporación de aplicaciones interactivas y especializadas, y aporta la posibilidad de distribuir contenidos ejecutables al ordenador del usuario (Applet); d) no permite realizar llamadas a funciones globales ni acceder a recursos arbitrarios del sistema, lo que aumenta la seguridad.

Como sistema básico de animación se utilizó la herramienta de animación JAWAA, que posee la misma arquitectura que la mostrada en la Figura 2. Recordemos que los comandos que modelan las animaciones se almacenan en un fichero de script, de modo que cuando se desea visualizar una animación JAWAA interpretaría el contenido de dicho fichero. Para hacer uso de JAWAA, necesitaríamos generar un fichero de script cada vez que el usuario quiere probar el funcionamiento de un algoritmo sobre una estructura de datos. Sin embargo, una de las restricciones de seguridad que impone el uso de Applets atañe a las funciones de entrada/salida a disco, puesto que no es posible utilizar ficheros en la máquina del usuario a menos que éste lo permita. De esta forma era necesario implementar un mecanismo más flexible para generar el script que modela cada animación. Para ello se añadió un método, que se encarga de leer los comandos de script desde un vector. Este nuevo planteamiento de colaboración entre componentes nos permitirá construir animaciones sin tener que escribir ficheros a disco para su posterior lectura en el intérprete de JAWAA (Figura 4).

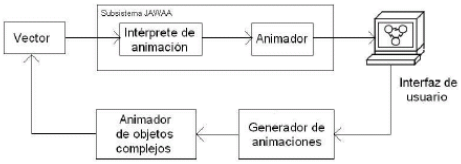

Figura 4. Modificación de la herramienta JAWAA

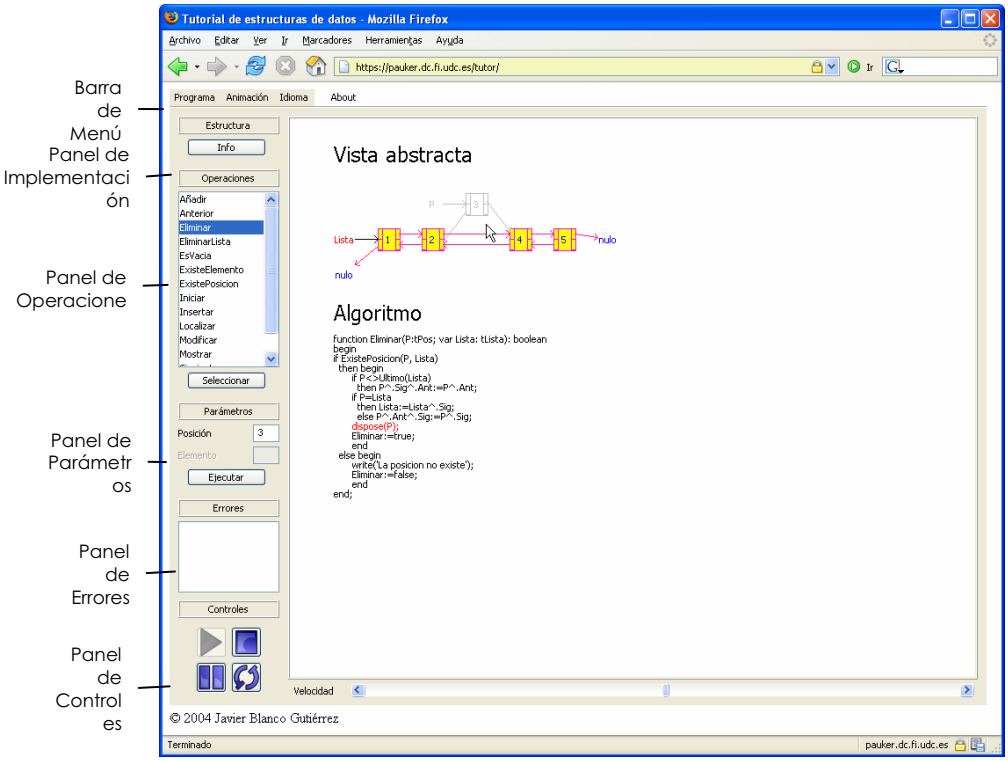

Figura 5. Interfaz de usuario del sistema tutor