SVED: SISTEMA DE VISUALIZACIÓN DE ALGORITMOS

Norma Moroni – Perla Señas
Laboratorio De Investigación y Desarrollo en Informática y Educación (LIDInE)
Instituto de Investigación en Ciencias y Tecnología Informática (IICTI)
Departamento de Ciencias de la Computación
Universidad Nacional del Sur - Bahía Blanca
Argentina
nem@cs.uns.edu.ar - psenas@cs.uns.edu.ar

Resumen

El aprendizaje de la programación tiene importancia no sólo desde el punto de vista de la adquisición de conocimiento, sino también desde lo que constituye el desarrollo de las capacidades de pensamiento. La Visualización de Software es una de las áreas de las Ciencias de la Computación desde la cual se pueden hacer aportes sumamente interesantes relacionados con la enseñanza-aprendizaje de la programación, especialmente con respecto a la lectura comprensiva de algoritmos. Se presenta en este trabajo el marco conceptual y el diseño de SVED, Sistema de Visualización de algoritmos de búsqueda y actualización de Estructuras de Datos.

Palabras clave: programación – aprendizaje – visualización de software – animación de algoritmos

1. Introducción

La programación suscita interés psicopedagógico debido a sus efectos en el desarrollo de las capacidades cognitivas. Contribuye en la actividad intelectual que permite establecer planificaciones y estrategias, construir algoritmos, estructurar instrucciones, analizar y comprender los programas propios o escritos por otros, aprender la sintaxis de los distintos lenguajes de programación, comparar recursos usados por los programadores expertos y por los principiantes. La computadora es una herramienta útil para el aprendizaje de programación de la misma manera que lo es para la enseñanza de distintas asignaturas.
En la búsqueda de nuevas estrategias metodológicas para la enseñanza de la programación, es interesante el paralelo que se puede realizar con los estudios sobre el aprendizaje de la lecto-escritura del lenguaje natural. Es muy común que al enseñar programación se centre la atención en la creación de algoritmos, descuidando el trabajo de lectura y comprensión de algoritmos existentes. Trabajar de esa manera puede pensarse en cierta forma equivalente a tratar de enseñarle a un alumno a escribir dejando de lado los contenidos de lectura mecánica y sobre todo de lectura comprensiva [1] [2].
En relación con la enseñanza de la programación se destaca la importancia de las actividades comprendidas en:

- ✓ construcción de programas
- ✓ lectura comprensiva de programas

Existe consenso en considerar un enfoque que parta de la resolución de problemas, para el primer ítem, y que se apoye en técnicas de visualización de algoritmos y programas para el segundo caso. Se aborda en este trabajo el estudio realizado sobre visualización de software, especialmente pensado para la lectura comprensiva de algoritmos y programas. Generalmente, para lograr una interpretación acabada de un programa se realiza el estudio del código fuente, que es incómodo y de difícil aplicación y, por otro lado, la construcción de casos de prueba para explicar la conducta de un programa, lo que es una tarea penosa y especulativa. Estas dificultades motivan el desarrollo de programas especiales que son usados para ayudar a explicar la conducta de otros programas [3] [4].

Si bien los entornos de los lenguajes de programación ofrecen capacidades cada vez más útiles al programador, como por ejemplo, la posibilidad de indentación, el uso de colores, los depuradores, éstos no son aún lo suficientemente amigables con el usuario como es deseable. Se ha pensado en el desarrollo de entornos específicos que ayuden y conduzcan efectivamente a los estudiantes en el proceso de aprendizaje de la programación de manera más gráfica, diversa y comprensible. La visualización de la conducta dinámica de los programas y de sus modelos abstractos, los algoritmos, y consecuentemente, el beneficio psicopedagógico de su aplicación en la enseñanza, constituyen una importante área de investigación que aún no ha sido completamente explorada [5].
La motivación especial que condujo a realizar este trabajo es la necesidad de mejorar y adaptar las herramientas existentes para facilitar la comprensión de los aspectos de la conducta dinámica de los algoritmos en las distintas fases del ciclo del software, pero fundamentalmente, poniendo énfasis en la comprensión de un software ya desarrollado. Para ello se debe contar con entornos de aprendizaje que conduzcan e induzcan tal proceso educativo. La calidad de dichos entornos está ligada íntimamente con su capacidad de logro en un proceso efectivo de enseñanza y de aprendizaje. Un entorno debe contemplar el nivel cognitivo, la capacidad de percepción, de respuesta, de abstracción, de los usuarios a los que va dirigido, como así también, la capacidad de interacción hombre-máquina, la velocidad de presentación, la estandarización de colores y sonidos, etc. Para ello se debe disponer de sistemas específicos, amigables con el usuario, que no actúen como elementos de dispersión en la elaboración y en la interpretación de la conducta del software propiamente dicho.

2. Visualización de software

La visualización de software es el uso de gráfica de computadoras y animación interactiva para ayudar a ilustrar y presentar programas, procesos y algoritmos. Se basa en el uso de diseño gráfico, de animación, de sonido, de video y tiene como característica sobresaliente la interacción entre el usuario y la computadora, apuntando a una mayor comprensión y a un uso efectivo del software. La tecnología de gráfica de computadoras es parte de este proceso y favorece notablemente la representación de los mismos. La información de la ejecución del programa se puede ofrecer en forma instantánea, o se puede presentar en un tiempo posterior, después que la ejecución se haya completado (análisis post mortem). Los analizadores en tiempo de corrida proveen refuerzo inmediato y permiten al usuario orientarlo respecto de la clase y nivel de detalle de la información monitoreada. El análisis post mortem puede realizar computaciones grandes para condensar la información de la ejecución y presentarla en una forma útil. Los métodos no son excluyentes.
Las herramientas en tiempo de corrida pueden ser pasivas o interactivas. En un sistema pasivo, la información es presentada al usuario, pero éste tiene poco control sobre la actividad del sistema; en un sistema interactivo, el usuario puede tener control externo sobre la información que se está exhibiendo, puede modificar el programa cuya conducta está siendo observada o bien el valor del dato que está siendo procesado [3].

2.1. Áreas de la visualización de software

La visualización de software comprende dos áreas fundamentales: la *visualización de programas* y la *visualización de algoritmos*.

2.1.1. Visualización de programas

La visualización de programas presenta vistas del código fuente del programa y de las estructuras de datos. La visualización estática de programa realiza un análisis del texto del mismo y provee la información que es válida para todas las ejecuciones independientemente de su entrada [6]. Los *Sistemas de visualización de programas* emplean editores sintácticos u optimizadores de código que producen un mejoramiento de la impresión, como indentado, diferencias de colores entre palabras reservadas y otros identificadores, la representación gráfica de la estructura del programa y de los datos; pero, no pueden explicar la conducta del programa ya que esto depende de los datos de entrada además del texto mismo. Una representación dinámica del programa provee información acerca de la ejecución de un programa específico sobre un conjunto de datos de entrada. Se puede realizar, por ejemplo, destacando las líneas de código fuente cuando éstas están siendo ejecutadas, estableciendo la pila de invocaciones de procesos y mostrando cómo los valores de los datos cambian mientras el programa corre [7].

2.1.2. Visualización de algoritmos

La visualización de algoritmos es el campo de la visualización de software que representa la semántica del programa de computadora. Constituye, por ese motivo, una representación más abstracta que la del programa en sí mismo y, al mismo tiempo, ofrece posibilidades mayores de interpretación de su conducta. La visualización de algoritmos comprende tanto la representación estática de los mismos como la dinámica. La visualización estática de algoritmos está representada generalmente por la descripción de la estructura de su especificación [8]. En ésta se muestran las relaciones entre las distintas componentes del software, que son especialmente útiles en la fase de diseño de los programas. Por lo general, esta representación se realiza por medio de gráficos, grafos y organigramas que permanecen invariables a lo largo del tiempo. Son importantes en cuanto a la visualización de la especificación del algoritmo, pero difícilmente suministran una visión adecuada de su funcionamiento.

La naturaleza dinámica de los algoritmos hace que estos sean difíciles de explicar con medios didácticos clásicos de pizarra, por eso el exhibir su evolución en el tiempo resulta un procedimiento más próximo al modo de operar del algoritmo y por ende más adecuado. Se habla en este caso de *animación de algoritmos*. Desde la aparición de los primeros sistemas de visualización [9], éste ha sido el principal uso de las visualizaciones de algoritmos. En este nivel de abstracción se representa la semántica de la ejecución del algoritmo, es decir, el comportamiento del algoritmo en ejecución, ocultando las operaciones y entidades que son internas a él, y que son irrelevantes en cuanto al modo de hacer del algoritmo que implementa. En un nivel de abstracción intermedio se presenta la comunicación entre los distintos módulos en forma estática y/o dinámica, y aún, en un menor nivel de abstracción, se presentan los valores de los datos que intervienen en los algoritmos como datos de entrada y como datos de salida, es decir los que son parte de la interface entre los distintos subalgoritmos durante la ejecución de los mismos.

La visualización de algoritmos se ha desarrollado comenzando desde los primeros diagramas y notas escritas a mano usadas por los programadores y científicos de la computación para describir la estructura del programa, hasta el advenimiento de las herramientas visuales y auditivas automatizadas actuales.

La visualización de algoritmos en cualquiera de sus dos formas, dinámica o estática, puede ser muy difícil de relacionar con las construcciones del programa que la codifican y que constituyen las especificaciones de su comportamiento. Es decir, se presenta allí una brecha entre la representación

del algoritmo y la especificación del programa subyacente. Esta diferencia se acentúa cuando la visualización pretende describir la semántica del algoritmo, (qué es lo que el algoritmo hace), pero que al mismo tiempo puede resultar excesivamente distante de su código fuente (cómo lo lleva a cabo). Este es un problema importante si consideramos el uso de la visualización para la comprensión del código del programa [10].

2.2. La visualización de software como recurso didáctico para la enseñanza de la programación

La *visualización de software* y especialmente la *visualización de algoritmos*, contribuye favorable y efectivamente en la comprensión de los mismos. Ayuda tanto en la tarea de interpretación del programa subyacente como en la de diseño, depuración, prueba y mantenimiento del software. Para ello se debe disponer de sistemas amigables con el usuario (facilidad de interacción hombre-máquina, en cuyo diseño se debe poner especial interés en la percepción humana), que no actúen como distractores en la interpretación de la conducta del software propiamente dicho.

La visualización de software presenta un medio audio-visual interactivo multimedial cuyas bondades en el campo de la psicopedagogía han sido comprobadas. Los sistemas no sólo deben estar diseñados por especialistas en computación sino por psicopedagogos que permitan establecer un equilibrio entre el exhibicionismo espectacular que presentan algunas visualizaciones (que a veces no pasa más que por la exhibición de los potentes recursos informáticos) y lo que el ser humano puede percibir efectivamente de ellas.

Las Animaciones de Software presentan importantes beneficios educativos ya que estimulan y ayudan a los estudiantes en las distintas situaciones del aprendizaje:

- ✓ Logran un incremento de la motivación. Esto es, a través de una presentación atrayente, en la que un algoritmo desafiante se transforma en uno más accesible y menos intimidante, los estudiantes se sienten más motivados a interactuar con el material y a estudiar temas complejos.
- ✓ Facilitan el desarrollo de destrezas por la oportunidad de realizar prácticas adicionales. Los estudiantes tienen una nueva forma para experimentar los algoritmos. Además de resolver los ejercicios en papel y escribir los programas, ellos pueden percibir visualmente los algoritmos y estudiar sus características observando e interactuando con la animación. [11].
- ✓ Asisten en el desarrollo de habilidades analíticas y promocionan las predicciones ya que los estudiantes deben coleccionar sus propios datos para el análisis del algoritmo y los subsecuentes diseños de algoritmos mejorados. Las animaciones de software ofrecen distintas ventajas comparadas frente a la ayuda ofrecida por la enseñanza tradicional, tal como el libro de texto y el pizarrón.
- ✓ Ofrecen un buen soporte al docente y son de gran ayuda en la clase durante la explicación de la conducta dinámica de un algoritmo.
- ✓ Permiten la exploración, jugando interactivamente, de las peculiaridades de un software, mejorando la comprensión individual de los estudiantes. Permite, a los mismos, manipular el software y sus entradas, formular hipótesis de la conducta del algoritmo y luego estudiar las acciones resultantes, verificando o refutando sus ideas. Esto los capacita para formar un modelo conceptual del algoritmo además de aprender el código. La interactividad agrega un nuevo nivel de efectividad al ambiente de aprendizaje, y es una herramienta apropiada en concepto de enseñanza ya que ella fuerza a los aprendices a tomar parte de la lección, como opuesto a simplemente observar un movimiento [12]. La interactividad ayuda a los estudiantes a adquirir una experiencia invalorable en la resolución de problemas [13].
- ✓ Es importante el valor educativo de las técnicas de visualización de software. La animación de algoritmos y la visualización de programas ayudan a los estudiantes a comprender los conceptos de software y también a los docentes en su tarea de enseñar dichos conceptos. Distintas

experiencias con respecto a la aplicación de la Visualización de Software en la enseñanza de la programación se han realizado en diversas Universidades extranjeras. Actualmente se emplean como recurso didáctico tanto en los cursos elementales como en los avanzados [14] [15].

Aunque todas las animaciones de algoritmos tienen el mismo propósito, el uso de las mismas puede variar considerablemente. Algunas de las aplicaciones parecen ser las más comunes [16]:

- ✓ Para acompañar una lectura y ayudar a comprender los conceptos claves que explica el instructor durante la clase.
- ✓ En un laboratorio formal donde los estudiantes interactúan con las computadoras.
- ✓ Para uso informal por los estudiantes fuera de la clase, en su tiempo libre, para ayudar a comprender más acerca de un algoritmo.
- ✓ Para realizar un aprendizaje personalizado e individualizado

3. Un Sistema para la visualización de algoritmos

Se presenta a continuación el diseño de un sistema para la visualización de algoritmos de búsqueda y modificaciones de distintas estructuras de datos (SVED). El mismo ha sido pensado para trabajar sobre las siguientes estructuras: tablas de hash abierto, árboles binarios de búsqueda, AVL, 2-3 y parcialmente ordenados.

SVED cuenta con las siguientes características:

Usa un código cromático que permite distinguir los elementos existentes en la estructura, de aquellos que se están modificando.

Para el caso de cualquiera de los árboles que se visualizarán en el sistema, el conjunto de elementos que se representan debe admitir una relación de orden lineal. Por tal razón, en este caso el tamaño de la representación gráfica de un nodo guarda una relación de proporcionalidad directa con el elemento que tiene asociado. El sistema representa los nodos mediante círculos cuyos diámetros calcula automáticamente.

SVED permite ingresar los elementos de un conjunto u optar por ingresar la cardinalidad *c* del mismo. En este último caso se genera automáticamente el conjunto $S = \{x / x \in N \text{ y } 0<x<=c\}$.

Las comparaciones entre elementos asociados a diferentes nodos de un árbol se visualizan mediante la superposición de sus respectivas representaciones gráficas.

En el caso de la tabla hash, el sistema asigna un color diferente a cada una de las clases, que se representan por medio de barras horizontales con el contorno coloreado. Los elementos del conjunto son representados con rectángulos del mismo tamaño y coloreados de acuerdo a la clase a la cual pertenecen.

El recorrido que se realiza en la búsqueda de un determinado elemento queda explícitamente marcado.

El encuentro de un elemento que se está buscando se señala de manera especial (ver cuadro 4 de la figura 2).

El usuario puede solicitar que la opción Mostrar (para cualquiera de las estructuras) se resuelva en forma estática o animada

La opción que permite la búsqueda de un elemento en la estructura, se resuelve de dos maneras diferentes: con animación continua o discontinua. La elección de la modalidad queda a cargo del usuario.

La representación de un subárbol de un determinado árbol se grafica por medio de un triángulo que aparece por detrás del mencionado subárbol.

SVED cuenta con animación de código. Se acopla con la animación de los algoritmos incluyendo un marco donde aparece el texto del algoritmo. Se destaca la línea de código que está siendo ejecutada en el algoritmo simultáneamente con la animación del mismo.
A modo de ejemplo, se presentan a continuación secuencias de imágenes correspondientes a las visualizaciones de operaciones de búsqueda y actualización de árboles binarios de búsqueda:
Sea el caso del ingreso del conjunto S = {2,9,15,4,7,5,12} y de la selección de la opción Mostrar realizados por parte del usuario. SVED responde, por ejemplo, con la imagen de la figura 1. Además el sistema solicita al usuario que se defina por una de las dos opciones siguientes, cuyo efecto se mantendrá mientras se trabaje con el conjunto recientemente ingresado:
El conjunto vuelve a su situación original luego de cada operación de actualización.
El conjunto guarda y refleja el resultado de las actualizaciones que se van realizando.

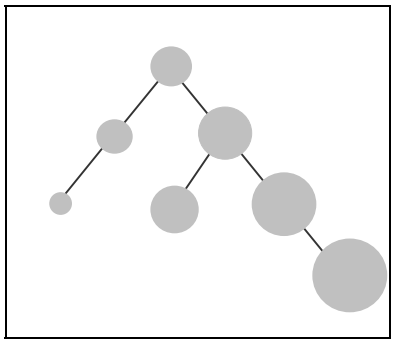

figura 1

Si el usuario selecciona la operación de búsqueda con el valor 7, se sucederán las imágenes de la figura 2.

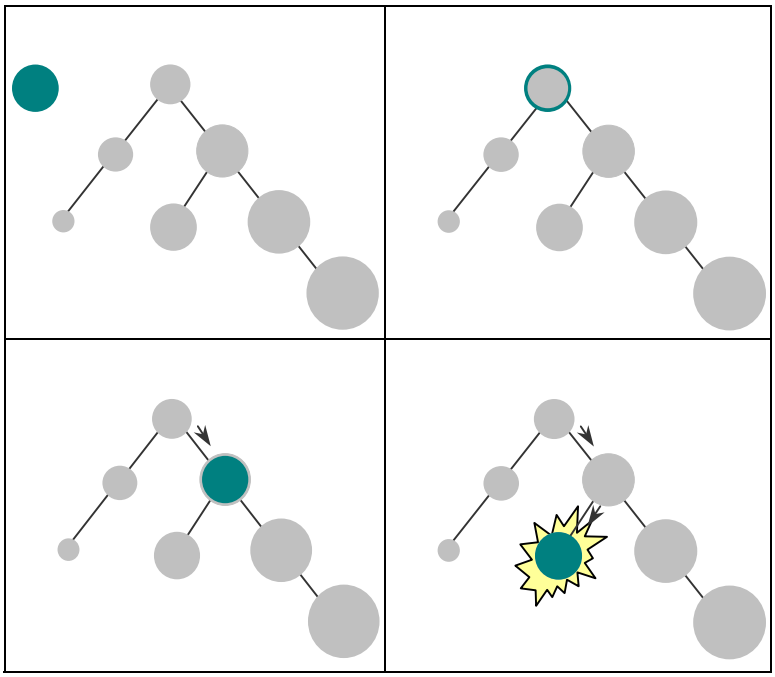

figura 2

La figura 3 muestra la respuesta del sistema a una solicitud de búsqueda con el valor 3 sobre el conjunto representado en la figura 1.

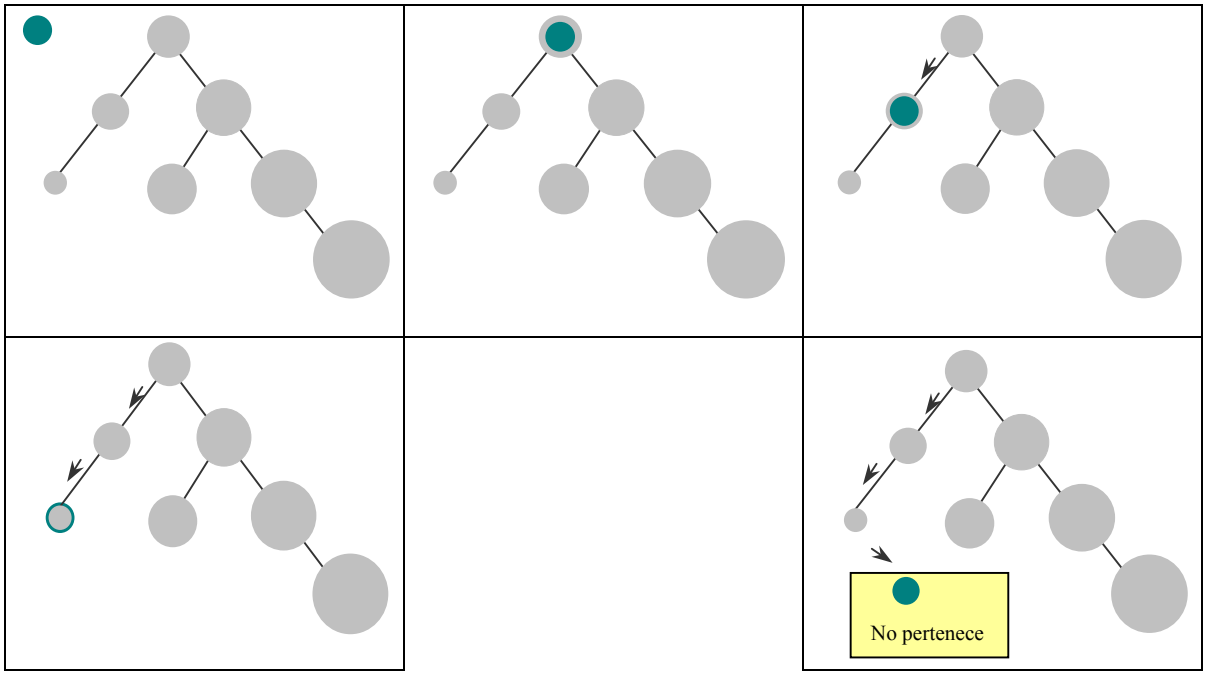

figura 3

En el caso de haber ingresado S = {x / x∈N y 0<x<10} y de obtener la visualización que muestra la figura 4, al ingresarse la opción Eliminar 1, SVED resolverá según se muestra en la figura 5. Puede observarse cómo se realiza primero la búsqueda del elemento, se señala luego, en el cuarto cuadro, el momento del encuentro, mostrando en el último la imagen final de la secuencia que permite visualizar la operación.

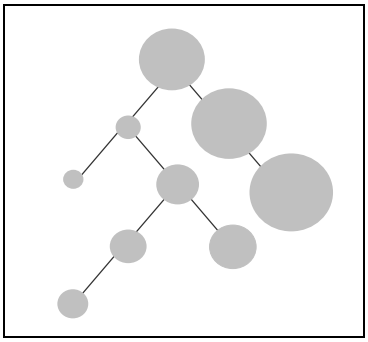

figura 4

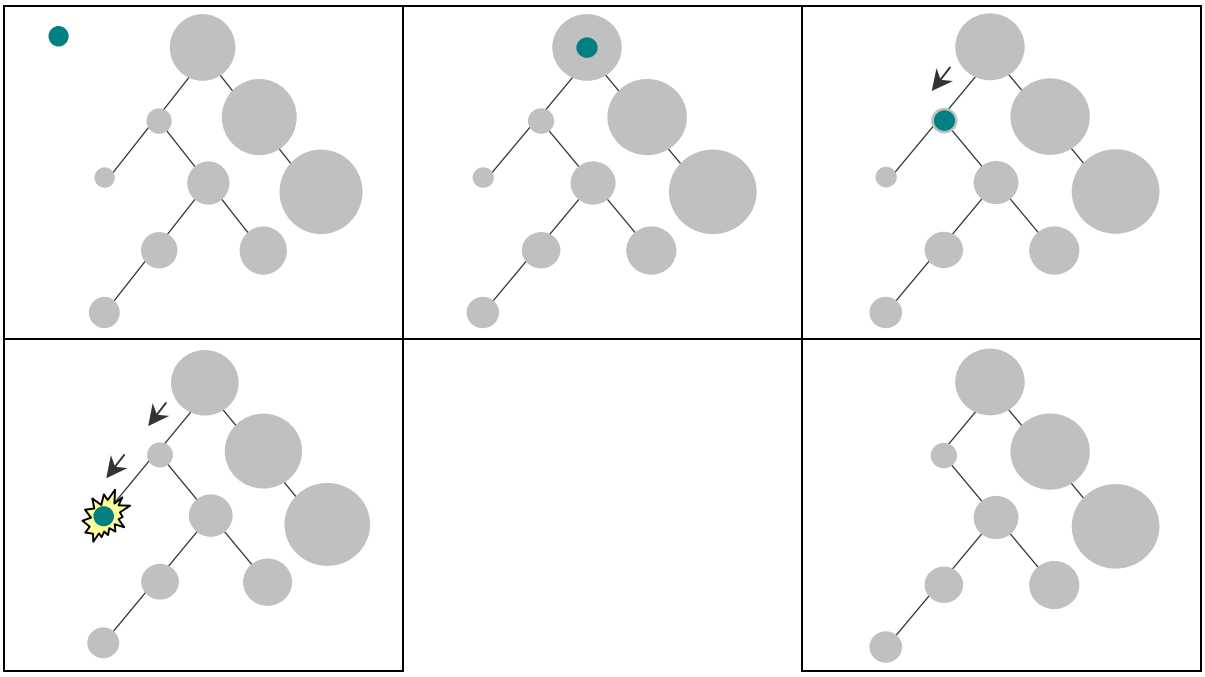

figura 5

Para el ingreso de S = {x / x∈N y 0<x<9} y su visualización (figura 6), la selección de la opción Eliminar 5, será resuelta por el sistema según se muestra en la figura 7.

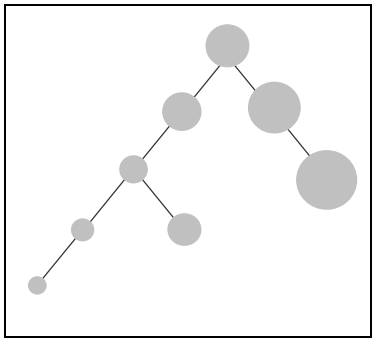

figura 6

Los tres primeros cuadros muestran la búsqueda del elemento; en el tercero, se destaca el elemento encontrado y la situación especial de tener un solo hijo. En el cuarto cuadro aparece una representación más abstracta como fondo del subárbol que se debe desplazar para terminar con la visualización de la operación solicitada.

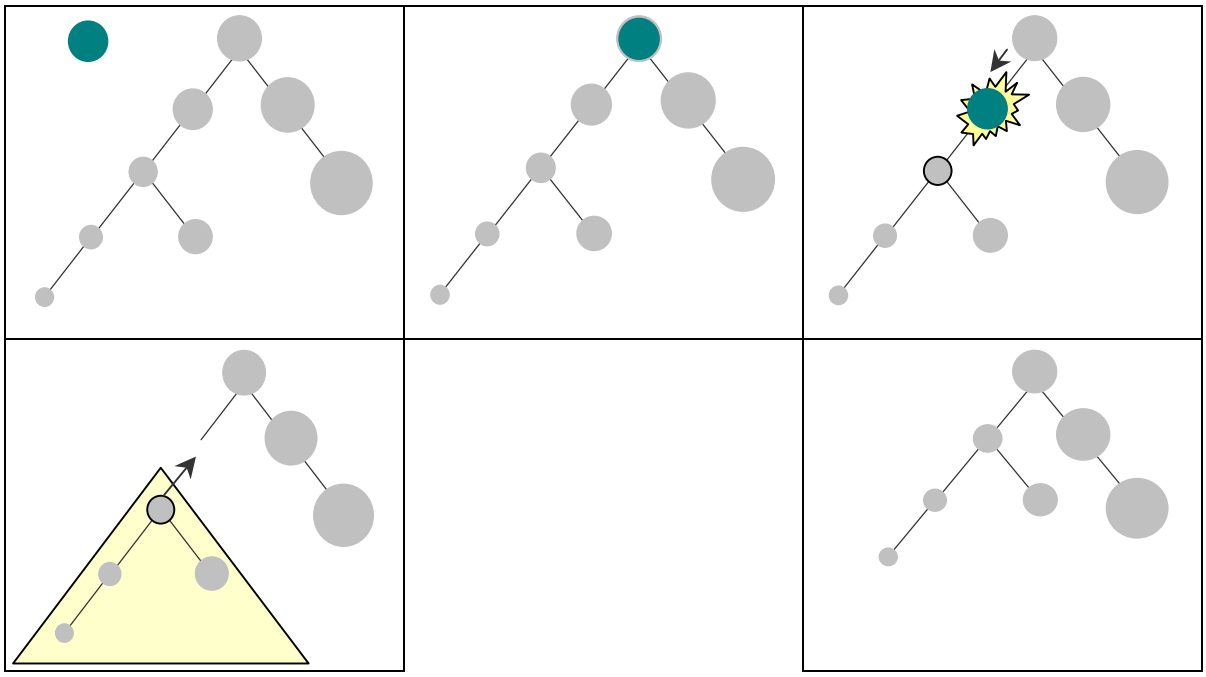

figura 7

Por último, la figura 8 muestra imágenes de la visualización obtenida a partir de la solicitud de Eliminar 2 al conjunto S = {x / x∈N y 0<x<10} de la figura 4

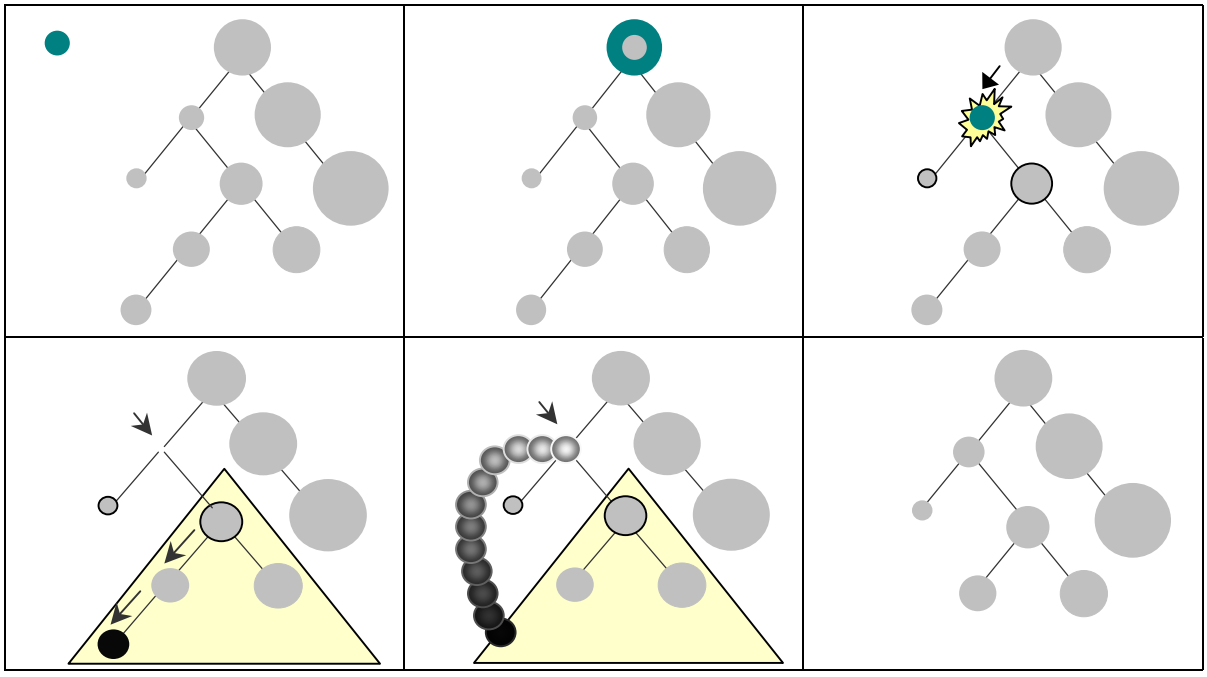

figura 8

Las secuencias mostradas en las distintas figuras son las instantáneas más relevantes de la visualización que se presenta en cada caso.

4. Conclusiones

Las visualizaciones de software, en general, tienen valor pedagógico en sí mismas ya que producen entusiasmo y son motivadoras para los estudiantes. Los docentes saben que atraen su interés, lo que resulta de ayuda invalorable en la instrucción. Las visualizaciones de software usadas como videos pasivos tienen menor impacto sobre el aprendizaje que las interactivas. Son de gran ayuda en los procesos de enseñanza y de aprendizaje cuando son usadas en forma apropiada. Es importante que los estudiantes interactúen activamente con las visualizaciones. Por ejemplo ellos pueden entrar datos para un algoritmo y observar luego una animación. Alternativamente, pueden observar la visualización y luego predecir qué va a ocurrir cuando se ingresen otros datos. También es beneficioso el hecho de permitir a los estudiantes diseñar y desarrollar visualizaciones de software [17].
Para la construcción de Sistemas de Visualización de Software se deben estudiar las capacidades humanas de captación, asimilación e internalización de la información para luego adaptar los recursos computacionales en un mayor beneficio de la comprensión de eventos. Los sistemas de animación bien diseñados, contemplando los aspectos perceptuales humanos, redundan en una mayor efectividad en su aplicación [18].

Bibliografía

[1] Álvarez Méndez, J. *Didáctica de la lengua materna*. Madrid. Akal. 1987.
[2] Contreras, D. *Enseñanza, profesorado y curriculum*. Madrid. Akal. 1994.
[3] Jeffery, Clinton L. *Program Monitoring and Visualization*. Springer-Verlag. 1999.
[4] Moroni, N. *Entornos para el Aprendizaje de la Programación.* WICC. 2001.
[5] Moroni N - Señas P. *La Visualización de Algoritmos como Recurso para la Enseñanza de la Programación*. WICC. 2002.
[6] Conroy K.- Smith R., *NEATER2: A AP/I Source Statement Reformatter*". Communitations of the ACM, 1972.
[7] Stasko, J.- Domingue, J.- Brown, M.- Price, B. *Software Visualization: Programming as a Multimedia Experience*. MIT Press, 1998.
[8] García, J.- Señas, P.- Moroni, N. *Cubik: Una Herramienta de Apoyo a la Enseñanza de la Programación.* IV Ateneo de Profesores Universitarios de Computación. San Luis. 1996.
[9] Brown - Sedgewick. *A system for Algorithm Animation*. ACM Computer Graphics 1985
[10] Gomez Henriquez. *Sistematización y técnicas de Visualización de Programas Concurrentes*. 1993.
[11] Lawrence, A- Brade, A.- Stasko, J.*, Empirically Evaluating the Use of Animations to Teach Algoritms*. Georgia Institute of Technology.
[12] PriceB.- Beacker R. - Small I. *An Introduction to Software Visualisation.*.MIT Press. 1998.
[13] Merril, P., at all . *Computers in education*. Allyn & Bacon. 1996.
[14] Naps, T. *Algoritm Visualization in Computer Sciencie Laboratiories.* SIGCSE. 1990.
[15] Basik, J- Tamassia,R- Reiss, S van Dam A. *SV in teaching at Brown University.*
[16] Stasko, J- Lawrence, A. *Empirically Assessing Algorithm Animations as Learning Aids*.1997.
[17] Stasko, J. *Tango: A framework and systen for algorithm animation*. IEEE Computer. 1990.
[18] Bailey, R. *Human Performance Engineering*. Prentice Hall. 1996.