Bosque Azul: Animación de Algoritmos para la enseñanza de estructuras arbóreas[1]

Fernando González - Norma Moroni – Perla Señas
Laboratorio De Investigación y Desarrollo en Informática y Educación (LIDInE)
Instituto de Investigación en Ciencias y Tecnología Informática (IICyTI)
Departamento de Ciencias de la Computación. Universidad Nacional del Sur
Bahía Blanca. Argentina
fggonzal@yahoo.com.ar - nem@cs.uns.edu.ar - ips@cs.uns.edu.ar
T. (0291) 4595101 int 2614 - fax (0291)4595136

Resumen

Bosque Azul es un software específicamente diseñado y desarrollado para la enseñanza y aprendizaje de los temas relacionados con Árboles Binarios de Búsqueda y con árboles AVL que se estudian en los cursos de Estructuras de Datos y Algoritmos
Está específicamente basado en la animación de algoritmos y en la visualización de programas. Permite una exploración vertical de los contenidos, asumiendo el aprendizaje de conceptos simples y elementales que se reflejan en las primeras animaciones para concluir con conceptos más complejos, y una exploración horizontal, realizando la comparación entre la representación de la misma estructura de manera abstracta y por medio de celdas y enlaces. Además de incorporar el texto del algoritmo va destacando, en cada momento, las acciones que se llevan a cabo paralelamente a la modificación de la estructura de dato respectiva. La animación continua, el código cromático y sonoro acompañan a la visualización del algoritmo y del programa.
Bosque Azul es una herramienta diseñada como material de aprendizaje, de consulta y de refuerzo. Tiene como finalidad ayudar a los estudiantes favoreciendo el aprendizaje individual y personalizado, y a los docentes aplicando una metodología de enseñanza presencial, semi-presencial o a distancia.

Palabras claves: visualización de software, Árbol Binario de Búsqueda, Árbol AVL

1. Introducción

El ser humano desarrolla su existencia transformando la realidad presente en otra deseada. Se observa, en tal sentido que las actividades del hombre consisten en percibir y actuar. Ambas tareas conforman un proceso cíclico, creciente y acumulativo; una acción manifiesta responde a una percepción previa y al cúmulo de acciones que han afectado anteriormente la realidad de la que el mismo hombre es integrante. Para el individuo, la evolución cíclica y espiralada de ambas actividades constituye el proceso de conocimiento del mundo que le permite crear su propia representación a manera de copia virtual en la que puede ensayar estrategias para lograr su cometido. A estas imágenes internas acerca del funcionamiento del mundo que limitan su modo de actuar se las llama *modelos mentales*. Toda persona tiene modelos mentales para cada fenómeno del mundo, más o menos claros en la medida en que ha conocido y cuestionado tales fenómenos, [1].
Por otra parte, las representaciones gráficas permiten la entrega rápida de vasta cantidad de información. La visualización en general tiene como misión importante crear un cuadro mental a semejanza del que la propia visualización se propone. Como la información contenida en las visualizaciones debe pasar a través del sistema perceptual humano, una cuidadosa atención a las características del mismo mejora su efectividad. La presentación de la visualización de una manera adecuada, provoca una gran actividad de razonamiento, de lenguaje, de memoria y de percepción,

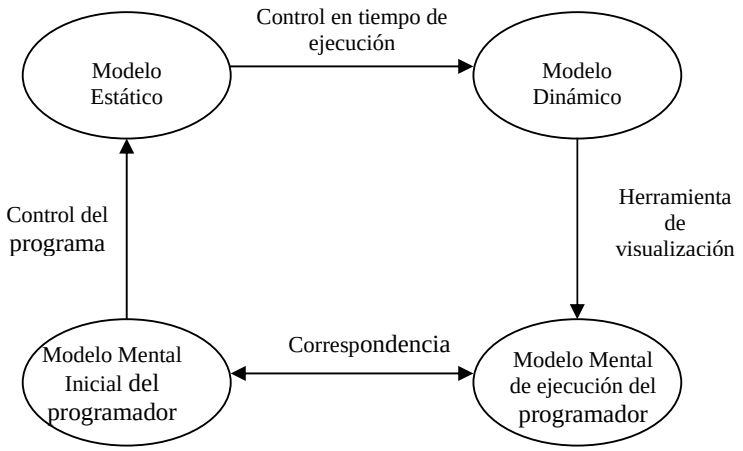
[1] El presente trabajo fue totalmente financiado por la Universidad Nacional del Sur, Bahía Blanca, Argentina.

con la finalidad de lograr sus objetivos. Frente a situaciones nuevas el ser humano trata de seleccionar desde la memoria algún esquema para ser adaptado a la realidad cambiando los detalles que sean necesarios. Para lograr esta influencia en el usuario se deben contemplar distintos principios que son requisitos importantes en una visualización efectiva.
Los principios de diseño gráfico forman una base para la evaluación y selección de las técnicas de visualización. Algunos de ellos son evidentes, mientras que otros surgen a través de la experiencia. Jeffery [2] cita las observaciones realizadas por Tufte con respecto a la excelencia gráfica: "la excelencia gráfica es la que ofrece al usuario el más grande número de ideas en el más corto tiempo con el menor gasto en el más pequeño espacio".
La habilidad humana para percibir más rápidamente representaciones pictóricas que textuales puede ser usada para ayudar a los programadores a comprender los programas que ellos leen, escriben y usan. La representación textual del código de un programa revela limitadamente la conducta del código en ejecución. Los gráficos son herramientas que logran inducir modelos mentales a partir de las ideas abstractas que se han mapeado en una representación visual [3]. Por tal motivo, la representación visual de los programas como la de su conducta, suministra un marco desde el cual los programadores pueden derivar sus propios modelos. Incluso, ayuda a acelerar los procesos de desarrollo de software, la comprensión del código en ejecución, la detección y corrección de errores. Usadas en esta forma las visualizaciones de programas agregan perspectivas adicionales que incrementan la comprensión del software.
La información se presenta con el fin de mejorar tanto la comprensión como la productividad del programador a través del uso eficiente de su percepción. El programador está capacitado para observar patrones de conducta dentro de la ejecución del código y para, rápidamente, detectar alguna desviación del comportamiento esperado. Sin embargo, dependiendo del paradigma de programación y de la plataforma de arquitectura usada, cambia la variedad y la manera en la cual la información debe ser exhibida.

2. Visualización de Software

Según Ousdoorn y otros en Aspects and Taxonomy of Program Visualization [4] el ciclo de desarrollo de programas se puede pensar como un proceso en el cual, en primer lugar, el programador crea una imagen mental de un problema. Este modelo mental es, posteriormente transformado en un modelo estático usando herramientas de desarrollo de programas tales como editores de texto, herramientas CASE, y compiladores. El modelo estático comprende el código fuente, el código objeto asociado, y otras formas de código intermedias. El modelo estático es transformado en el correspondiente modelo dinámico que se refiere a la conducta del programa en tiempo de corrida. La transformación se realiza usando el control del entorno del programa en tiempo de corrida, tal como el sistema operativo y las librerías respectivas. Esta sucesión de transformaciones corresponden a las que realiza un programador desde que se enfrenta a un problema hasta que ejecuta el programa desarrollado.
Los autores mencionados sostienen que la visualización de un programa es un proceso inverso al descrito anteriormente, en el cual el modelo dinámico puede ser mapeado en una representación visual de la ejecución del programa similar al modelo mental que el programador se plantea del mismo. Este mapeo es importante porque posibilita al programador superponer el modelo mental de ejecución con el modelo mental original el cual ha concebido durante la creación del programa.
La dificultad de las visualizaciones de software estriba en la carencia de un modelo físico previo ya que no existen entidades tangibles a las cuales asirse para diseñarla. Para ello se debe tener en cuenta, en primer lugar, lo que se quiere visualizar y la manera en que se va a obtener la información necesaria para hacerlo. En segundo lugar, se debe decidir cómo se va a llevar adelante la visualización, es decir, la especificación de la visualización.
En las visualizaciones de software, en las que se utilizan grandes niveles de abstracción, puede resultar difícil especificar la apariencia visual de los objetos lógicos presentes. En este caso el

sistema de visualización debe ofrecer un conjunto de transformaciones y entidades gráficas para realizar la especificación. Al momento de definirla, el visualizador hace uso de estos elementos.
El diagrama representa un modelo de ciclo de desarrollo de un programa relacionado con la visualización del mismo.



La *Visualización de Software* con el uso de representaciones interactivas y multisensoriales, facilita la comprensión humana de la conducta y la interpretación efectiva del programa [5]. La enseñanza de la programación y de la manipulación de estructuras de datos encuentran, con este recurso, un mecanismo de enseñanza y aprendizaje que ayuda también a la lectura comprensiva de algoritmos [6]. Considerando la inclusión de visualizaciones explicativas de procedimientos de actualización de estructuras de datos, se genera una poderosa herramienta que extiende el aprendizaje de la programación, con medios de resolución de problemas y pautas de estudio, para el dominio y la manipulación de nuevas estructuras de datos.

3. Objetivos de la herramienta

Se han establecido los siguientes objetivos para el diseño y desarrollo de la herramienta educativa que se presenta:

- Brindar ayuda al alumno de un curso de Estructuras de Datos y Algoritmo para poder utilizar la herramienta como material de aprendizaje básico, de consulta y de refuerzo tanto de los conceptos impartidos en clase como de los encontrados en los libros de texto sobre el tema.
- Solucionar problemas de comprensión al momento de pasar de una representación abstracta a una representación con celdas y enlaces, proveer un mecanismo claro que demuestre el paralelismo entre ambas representaciones, de manera que el uso indiscriminado de notación durante las clases y explicaciones, pueda ser comprendido o consultado ante confusiones.
- Brindar al docente un medio audiovisual que facilita la explicación de nociones netamente dinámicas, difícilmente explicables con los medios convencionales.
- Presentar un mecanismo de estudio de una estructura de datos. Iniciando por un análisis sencillo y básico de la manipulación de la estructura de datos, pasar a una representación formal, finalizando con la incorporación del algoritmo en pseudo-código o en algún lenguaje de programación. De esta manera se brindan los pasos y pautas a considerar y a emplear al momento de iniciar el estudio de nuevas estructuras de datos individualmente.
- Solucionar ciertos problemas básicos, conceptuales fundamentales para el desarrollo de un curso de Estructuras de Datos y Algoritmos que se aplican inmediatamente sobre las distintas

representaciones (pasaje de parámetros, realización de una traza, pila de activación de llamadas recursivas, etc.).

4. Bosque Azul

Bosque Azul es un software de Visualización de Algoritmos y Programas desarrollado con la finalidad de ayudar en la actividad de aprendizaje en un curso de Estructuras de Datos. Luego de un análisis de distintas herramientas de desarrollo se procedió a la elección de Macromedia Flash para implementar las animaciones debido a su naturaleza Web, a la facilidad que presentó para plasmar los diseños y las ideas. Las animaciones resultantes son de un tamaño muy pequeño, lo cual es favorable si se considera que la organización global de la herramienta es un sitio Web. Se conjuga relativamente bien con una serie de aplicaciones complementarias como Macromedia Freehand. Además, Macromedia Flash cuenta con un entorno de desarrollo sencillo, poderoso y que permite gran expresividad. La incorporación de sonido a las animaciones se realiza fácilmente. No obstante la herramienta utilizada presenta el inconveniente de restringir la interactividad de la visualización con el usuario. Para salvar este inconveniente, los contenidos de cada operación fueron lo suficientemente elaborados para cubrir todas las posibilidades sin que sea necesario que el observador se involucre a nivel operativo. Se derivó la falencia de interactividad en ciertas ventajas que aportan beneficios considerando que el observador frente a distintas posibilidades de acción pueda distraerse jugando con la herramienta en lugar de percibir lo esencial. Los ejemplos creados permiten cubrir todas las alternativas y casos particulares introduciendo conceptos imprescindibles sin el riesgo que el usuario se distraiga probando distintas alternativas o repitiendo casos. Más aún, considerando que los árboles presentan configuraciones en las que encontrar casos particulares puede resultar un tanto difícil.

4.1. Controles para la reproducción de la animación: En cada una de las animaciones, se han colocado, en lugares estratégicos, detenciones de la misma por diversas circunstancias, finalización de una explicación, puntos intermedios del proceso de una operación o simplemente la oportunidad de ver detenidamente una instantánea y tener la posibilidad de repetir la reproducción de ciertos fragmentos de la animación.

La animación cuenta con distintos tipos de controles, como continuar con la reproducción de la animación, repetición del último fragmento visualizado, repetición de la operación visualizada, reinicio de la animación. La detención de la animación no se da como alternativa ya que permitiría interrumpir la dinámica de la misma en cualquier punto, vulnerando la eficacia de la transmisión de la información, habiéndose planificado la forma en que la transmisión de conceptos se realiza.

4.2. Contenidos temáticos

La herramienta educativa desarrollada se plantea como un conjunto de componentes pedagógicos de diversos tipos, texto, animaciones, gráficos y sonidos. Se previó una organización lo suficientemente flexible para poder incorporar, quitar y actualizar los contenidos. Es por ello que la herramienta se estructuró con el formato de un sitio Web, en el que se dispone de distintas secciones y en el que se puede realizar una exploración sencilla de los contenidos.

4.2.1. Presentación general de los contenidos.

Los dos grandes temas a tratar son Árboles Binarios de Búsqueda y Árboles AVL. Para cada uno de estos temas se analizaron minuciosamente sus operaciones principales y las formas de manipulación de cada estructura de dato [7]. El siguiente esquema muestra los contenidos temáticos desarrollados para cada representación

- Árbol Binario de Búsqueda(ABB)
    - Comparación entre árboles que cumplen y árboles que no cumplen la propiedad ABB.

- Generación.
- Búsqueda.
- Inserción.
- Eliminación (una hoja, un nodo interno con un hijo, nodo interno con dos hijos).

➢ Arbol AVL

- Rotaciones simples y dobles.
- Generación.
- Inserción.
- Eliminación.

4.2.2. Formas de Exploración de los contenidos

Como la finalidad de la herramienta es que sea tanto de apoyo al docente durante la introducción de los temas como de material de consulta para el estudiante, se dispuso una organización que permita diversas formas de acceso a los contenidos.

La herramienta se puede explorar en forma vertical, en forma horizontal o en forma directa.

– *Exploración vertical*: Realizando este tipo de exploración se recorren los contenidos de tal manera que se elige uno de los temas y se recorren sus distintas secciones de manera que se evoluciona con el planteo natural de los contenidos, desde lo simple a lo complejo. Una vez que el observador completa el recorrido de las secciones verticalmente, tiene un conocimiento completo y acabado del tema.
– *Exploración horizontal*: Mediante esta forma de recorrer los contenidos se apunta a obtener una comparación en la forma en que una operación actúa en distintas estructuras de datos. Se pueden ver las diferencias exactas entre realizar una operación sobre una estructura de datos con ciertas propiedades y aplicar la misma operación sobre otra estructura de datos con propiedades diferentes. Este enfoque permite obtener un conocimiento a nivel de detalle y las diferencias de los temas sobre una operación específica. El uso de este tipo de exploración es el que se realizará una vez que el tema ya se ha visto y se necesita una revisión o salvar una duda o confusión entre las estructuras de datos.
– *Exploración Directa*: Es el acceso directo a una operación específica sobre una estructura de datos particular. Permite repasar detalles y resolver cuestiones específicas. Es un tipo de acceso a los contenidos para el estudiante que ha tenido cierta experiencia con la herramienta.

5. Conceptos y técnicas consideradas para el desarrollo

Con el fin de satisfacer las necesidades de la persona que interactúa con la representación resultante de la *Visualización*, todo lo informado a través de la misma debe tener en cuenta aspectos de la percepción y de la adquisición del conocimiento humano [8] [9].

- *Percepción*

Las complejas y poderosas capacidades del sistema perceptual humano pueden ser aprovechadas para facilitar la comprensión de sistemas informáticos de mediana y alta complejidad. Como la información contenida en las visualizaciones debe pasar a través de dicho sistema, la atención cuidadosa a las características del mismo mejora asombrosamente la efectividad de las visualizaciones. Existen diversas técnicas que favorecen la percepción de acuerdo a la información a transmitir. A continuación se nombran algunas de las más utilizadas:

– *Abstracción*: Una de las metas principales de la visualización en general es la abstracción de la información con el fin de presentarla en forma más útil, en una representación de más alto nivel. Para ello se necesita estimar la información que se presenta y el nivel de detalle. En particular la visualización de software surge de una información de por sí abstracta cuya representación debe sugerir el comportamiento de un programa.

- *Interacción limitada del usuario*. La interacción con el usuario es limitada, para evitar la inclusión de elementos que puedan alterar la correctitud de la exhibición conduciendo a errores de conceptos. La interacción permitida brinda una relación temporal individual, de manera que cada observador pueda establecer su propio ritmo de aprendizaje. La herramienta permite la repetición de fragmentos de la animación y el reinicio de pausas específicas, utilizadas para congelar contenido explicativo. La visualización provee al usuario control sobre las vistas, de manera tal que se pueda usar la herramienta en forma personalizada. Permite mecanismos de pausa, continuación y de vuelta atrás [10].
- *Máxima densidad de información*. La herramienta se comporta como un buen proveedor de información remedial. Esto significa que la visualización da información al usuario y no la racionaliza. Una alta densidad de información permite mayor cantidad de información en el espacio disponible. La densidad se incrementa eliminando los espacios vacíos ya sea adicionando más información o reduciendo el tamaño de la pantalla. La alta densidad de información tiene su precio: la percepción humana. La herramienta desarrollada incrementa la densidad de información manteniendo una gráfica organizada. La complejidad organizada permite una rápida asimilación y es estética en apariencia. Las visualizaciones deben presentar tanta información como sea posible sin abrumar al usuario. Se debe hacer un balance entre la alta información contenida y la baja complejidad visual.
- *Estética.* La visualización también incorpora suficientes elementos estéticos, teniendo en cuenta que el resultado gráfico y los atributos del display deben ser información útil al usuario y no ser un elemento meramente decorativo.
- *Énfasis sobre el interés*. Se ha tenido en cuenta la importancia de considerar que las características más llamativas de la imagen sean también las más importantes de la escena. Para ello se han dispuesto colores brillantes, movimientos, cambios, límites sobresalientes o altamente saturados. Los arcos entre nodos en la representación de celdas y enlaces se encuentran representados por medio de flechas que modifican sus formas, sus orientaciones y cuyos colores cambian a medida que se va desarrollando la animación. La herramienta ayuda a los usuarios a focalizar rápidamente un punto particular de interés.
- *Metáforas visuales*: Las metáforas introducen conceptos familiares para el usuario de las visualizaciones y proveen un buen punto de comienzo para ganar en la comprensión de la visualización. Una metáfora visual familiar o inferida fácilmente por la conducta a ser presentada, puede disminuir la carga cognitiva impuesta sobre el usuario e incrementar la velocidad de comprensión. Aunque algunas metáforas son tomadas naturalmente de un dominio de aplicación específico o una notación de uso común por los programadores, otras son tomadas de la naturaleza o de símbolos no técnicos encontrados en la vida diaria. Las visualizaciones son más efectivas cuando su esquema y coreografía contienen elementos visuales y auditivos que están asociados con la experiencia del usuario. En el sistema desarrollado, las burbujas diseñadas, presentan iluminación superior, levemente desplazada a la izquierda, contemplando el hecho que los humanos están acostumbrados a fuentes de iluminación que proceden desde arriba de los objetos en una escena. Además, por medio de sombras, se logra una sensación de tridimensionalidad en las celdas que representan los elementos del árbol. De la misma manera, las sombras son buenos recursos para exhibir objetos y, además, la proyección de losmismos sobre una superficie es un método potente de comunicar su ubicación en el espacio. La información debe estar presentada de manera familiar para el usuario [11]. Esta característica se ha tenido en cuenta en el desarrollo de la visualización.
- *Bajo nivel de asombro:* Se ha contemplado este principio ya que es importante en visualizaciones animadas que la pantalla mantenga sus características básicas para evitar que los usuarios se vean forzados a estudiar continuamente cada imagen.
- *Interconexión:* La comprensión de una pieza de software implica la comprensión de una variedad de conductas distintas y de relaciones entre ellas. En general no se puede anticipar

cuáles subconjuntos de información disponible son necesarios para el usuario pero en Bosque Azul se enfatiza sobre posibilidades de visualización múltiple, con ejecución simultánea, mostrando diferentes aspectos de la conducta del programa.

- *Escala dinámica*: Las escalas impuestas en la descripción de información densa deben ser altamente dinámicas. Si la escala no cambia dinámicamente, muchas visualizaciones agotan el espacio y pierden detalles de la ejecución que está siendo observada. Pero, se debe tener en cuenta que los cambios de escala demasiado frecuentes son caros computacionalmente y desorientadores para el usuario.
- *Telón de fondo estático:* las herramientas de análisis dinámicas son siempre mejor interpretadas cuando se superponen a un contexto que consiste de información adquirida por análisis estático. La información estática puede proveer información familiar a los programadores sobre la cual se maneja la dinámica. Un ejemplo de esto en el software es el código fuente.
- *Representación*: Este principio se refiere a la forma en que las distintas componentes del sistema de software son exhibidas y de qué manera la información respecto de esas componentes puede ser codificada dentro de esa representación gráfica. En el software descrito se tiene en cuenta el criterio de individualidad que indica que las representaciones de componentes diferentes deben presentarse tan contrastantes como sea posible. De esta manera las representaciones son fácilmente reconocibles tanto como si son idénticas o como si son distintas, aún dentro de una visualización grande. La representación de los nodos de los árboles en forma abstracta es completamente distinta a la los mismos por medio de celdas.
- *Recorrido*. Las visualizaciones deben ser diseñadas con el usuario en mente. Es necesario estructurar la visualización e introducir mecanismos para ayudar al usuario en el recorrido a través de la misma.
- *Eliminación de información inútil.* Muchas de las tareas de visualización tienen más información disponible que lugares donde exhibirlas. La información no esencial distrae y va en detrimento de la información más importante. La información tiene un rango de utilidad, especialmente en situaciones de exhibición en las cuales los usuarios no están seguros de lo que están mirando. Una idea que se ha seguido es que el énfasis y espacio dado a una información sea proporcional a su utilidad, es decir, la selección depende de la importancia que el usuario le da a un objeto particular en un momento particular.
- *Variación de niveles de detalles*: se ha tenido en cuenta el nivel de detalle de la información contenida. Cuando la información contenida ya ha sido visualizada, en algunas situaciones se ha simplificado con el fin de enfatizar sobre la nueva información disponible, es decir, se contempla el hecho que a medida que la complejidad de las animaciones crece, se disminuye el nivel de detalle con el fin de priorizar los conceptos importantes a transmitir. Con ese propósito, en las animaciones donde se representa en paralelo el código y el árbol con celdas y enlaces se elimina la representación de los subárboles, incluida en animaciones de etapas anteriores. La complejidad visual de las representaciones es beneficiosa de acuerdo al contexto en el que es usada.
- *Integración de recursos gráficos*: El uso de recursos tal como color, sombra, y particularmente tamaño para codificar la información en una representación puede reducir el alcance que provee la información dentro de la visualización. Por ejemplo, si una representación usa el tamaño para codificar alguna información mientras que otra usa la profundidad posicional para codificar otra información se puede prestar a confusión cuando se diferencie entre una componente que está cerca o es grande o bien está lejos o es pequeña. Se ha tenido especial cuidado en este sentido.

- *Técnicas de Animación de Algoritmos*

- *Instantáneas y signos de estado*. La animación es una técnica que se adecua perfectamente para representar las relaciones temporales de la conducta dinámica de un programa en ejecución. El software desarrollado presenta animación continua en la modelización de la conducta del

algoritmo. La animación muestra cambios en el estado de la estructura de datos modificando su representación gráfica sobre la pantalla. Las señales del estado reflejan el comportamiento dinámico del algoritmo. Además, los signos de estado vinculan distintas películas representando objetos específicos de la misma manera en distintas animaciones. Por ejemplo, el uso de la pila de activación y de cada indicador de llamada recursiva es un grupo de objetos específicos dentro del conjunto de objetos que conforman la animación y que describen el comportamiento recursivo de los diversos algoritmos.

- *Historia Estática*. Una vista estática de la historia y de las estructuras de datos de un algoritmo ayuda a explicar una pseudo-ejecución del algoritmo como lo hace un ejemplo en un libro de texto. Tales vistas estáticas le permiten al usuario familiarizarse con el comportamiento dinámico del algoritmo a su propia velocidad, y teniendo la oportunidad de repetir ciertos fragmentos hasta esa vista estática, permitiéndole descubrir el comportamiento de la operación que deriva en dicha vista.
- *Modificaciones continuas o discretas*. La representación gráfica de un cambio en una estructura de datos puede darse en forma continua o discreta. Los cambios continuos son más útiles para conjuntos de datos pequeños, donde la complejidad de las modificaciones sobre la estructura requiere la comprensión y el análisis del cambio producido. Las modificaciones continuas suponen una dinámica un tanto más trivial o simple, de manera que el usuario puede anticipar cierto comportamiento. Las modificaciones deben ser suaves a medida que los eventos ocurren para ayudar a describir las operaciones individuales de un proceso y permitir al usuario percibir y comprender fácilmente los cambios. La modalidad discreta ofrece instantáneas de posiciones previas a la operación y posteriores a la misma sin reflejar el camino recorrido. Se adecuan a sistemas con gran cantidad de elementos. El software desarrollado presenta animación continua. Por ejemplo, un arco entre nodos es representado como una flecha que es animada lentamente sin parar desde su referente anterior y eventualmente apuntando a su futuro referente. Además, se introduce una serie de cuadros intermedios para actuar como una forma de interpolación entre los estados iniciales y finales. Dado que los algoritmos a animar tienen un grado de complejidad importante, y que se intenta mostrar un gran conjunto de información, las modificaciones continuas permiten introducir los cambios de manera más atinada permitiendo al observador entender completamente la secuencia realizada, con la posibilidad de rever el último.
- *Complejidad de los algoritmos*. La visualización de un algoritmo pretende la comprensión general del propósito del mismo. Para algoritmos que contienen un alto grado de detalle, o que se desglosan en varias partes, se reemplazan grupos de acciones en una acción general que encapsula todo el nivel de detalle. En la animación del algoritmo, cuando se reproduce la ejecución, al alcanzar uno de dichos bloques, se busca distinguir su ejecución, desplegando un nuevo grupo de acciones correspondiente a ese bloque mostrando todos los detalles ocultos. Esta técnica también fue usada para evitar la técnica de desplegado en la vista del código, que a veces no resulta adecuada.

- *Técnicas de Color*

El color tiene el potencial de comunicar una enorme cantidad de información eficientemente. Se usa el color en varias formas distintivas: para codificar el estado de las estructuras de datos, para resaltar la actividad, para agrupar vistas, para enfatizar patrones, para realizar historia de la animación.

- *Codificar el estado de las estructuras de datos*. El color, como indicador del estado del algoritmo, mejora y complementa las técnicas gráficas descriptas anteriormente, facilita la distinción de nueva información incorporada a la vista, resalta objetos importantes en la evolución de la animación.
- *Resaltar actividad. M*uchas animaciones pintan temporalmente ciertas regiones de la animación intentando captar en momentos específicos la atención del observador. Como la actividad de

resaltar es temporaria, se debe contar con la capacidad de quitarlo y rápidamente restaurar las partes de la escena que había ocultado.

– *Agrupar vistas múltiples*. Las vistas múltiples, frecuentemente muestran aspectos diferentes de la misma estructura de datos o diferentes representaciones de objetos relacionados lógicamente. En estos casos, un grupo de animaciones genera un entorno o una gráfica integrada y armónica ilustrando ciertas características con los mimos colores en todas las vistas.

- *Técnicas de Sonido*

La inclusión de sonido en las visualizaciones de algoritmos está ligada al alto grado de capacidad del ser humano para detectar y recordar patrones de sonidos. En las visualizaciones de algoritmos el sonido se incluye para refuerzo visual, para indicar patrones de comunicación, para detectar reemplazos visuales y para señalar condiciones excepcionales, aumentando el nivel de expresividad. Por otra parte el sonido, ayuda en la interpretación de la animación que se lleva a cabo y permite una conducta más pasiva por parte del usuario ya que se pueden detectar eventos excepcionales, sin necesidad de estar abocado a la observación de los mismos. El software en desarrollo establece un código auditivo donde no hay sobrecarga en los sonidos. Por ejemplo, el sonido para indicar que un objeto es hermano izquierdo de otro, es distinto al que indica que es hermano derecho. Como el sonido es intrínsecamente dependiente del tiempo, no es sorprendente que sea muy efectivo para mostrar fenómenos dinámicos tal como ejecuciones de algoritmos. Pero, el sonido es un medio más difícil para dominar, con muchos parámetros, frecuencia, volumen, velocidad de ataque y decadencia, duración, timbre, reverberación, brillantez, estéreo [12].

El resultado de investigaciones sobre la incorporación de sonido a la visualización de algoritmos ha arrojado varias razones que sustentan que esta clase de proyectos incluyan audio.

– La visualización es altamente subjetiva y lo que es claramente comprensible para una persona puede no ser significativo para otra. El uso de sonido provee otra “vista” de una animación.
– La audición se realiza en forma pasiva. Esto es, el usuario no tiene que estar prestando estricta atención auditiva del normal comportamiento de un programa para notar que algún evento excepcional ha ocurrido.
– La audición se puede realizar en paralelo con la visualización, en cuyo caso la refuerza, o en forma independiente.

6. Ejemplo

A continuación se presenta una sucesión discreta de las instantáneas de la animación que se observan en la visualización de la operación de Inserción de un elemento en un árbol AVL [Weiss]. En este caso se desea insertar el elemento de valor 10 en un árbol AVL ya creado, Fig.1, donde se indican en una pila dispuesta en la parte inferior de la misma y por medio de su número correspondiente en el texto del algoritmo, las invocaciones recursivas realizadas hasta ese momento. En la Fig.2 se destaca un comentario y un desplegado de las acciones de creación de la celda y de inserción del elemento que se deben realizar. La Fig.4 y la Fig.5 muestran la ubicación dentro del árbol del nuevo nodo destacando su factor de balance. El punto rojo a la derecha de la acción indica que el efecto de la misma es el que se está observando en ese momento. A la derecha de las vistas se observa el cambio de valor booleano cuando el elemento se inserta en el árbol, Fig.6. En las Fig.7 y Fig.8 se puede observar el análisis que se realiza del factor de balance en la rama donde se ubicó el elemento. En la Fig.9 se presenta un mensaje al respecto. A continuación en las siguientes figuras de presenta el cambio en el factor de balance y las rotaciones que se deben realizar para conservar la propiedad de árbol AVL. Se destacan en ellas los nuevos enlaces, representados por las flechas amarillas, que se establecen entre los nodos del árbol. En particular, la Fig.11 muestra una instantánea de la animación que mueve el nodo de valor 10 a su ubicación final, Fig.12.

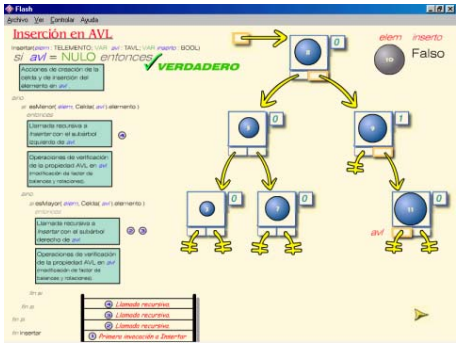


Figura 1

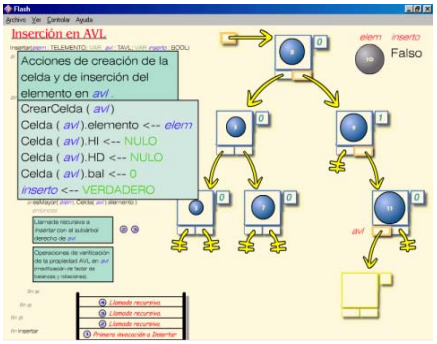


Figura 2

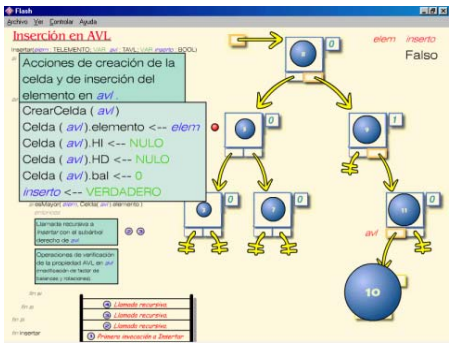


Figura 3

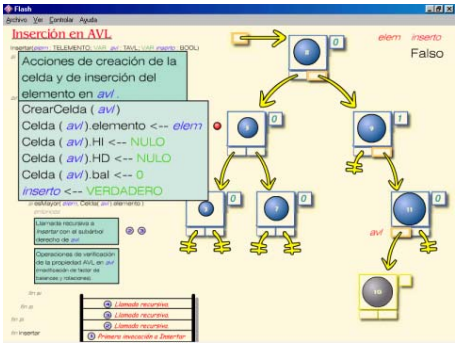


Figura 4

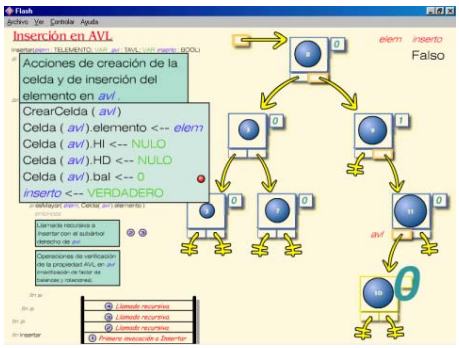


Figura 5

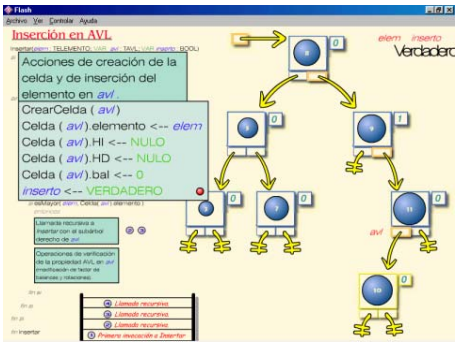


Figura 6

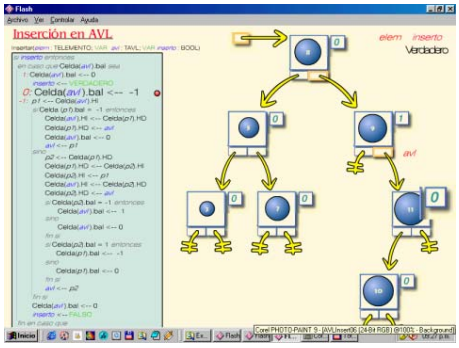


Figura 7

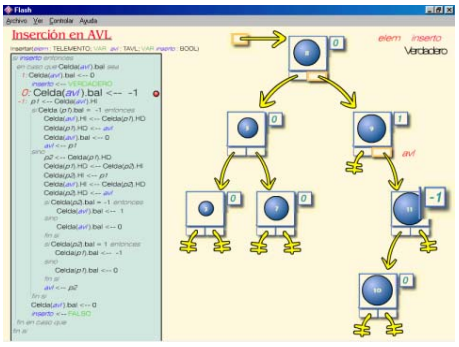


Figura 8

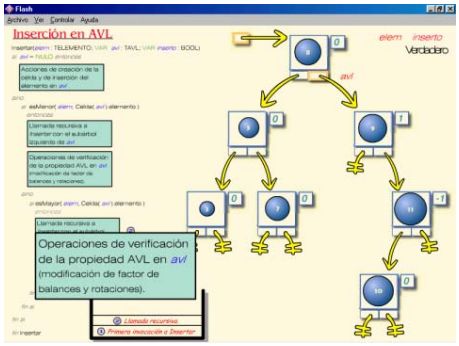


Figura 9

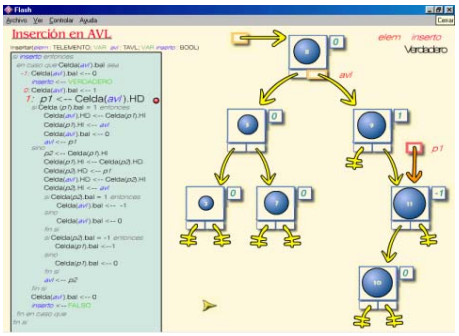


Figura 10

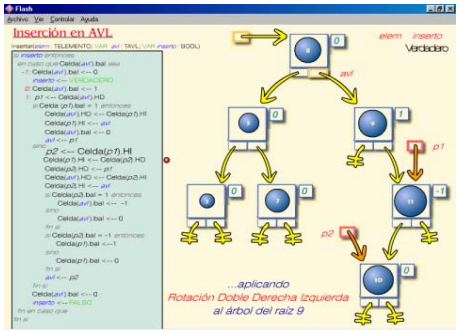


Figura 11

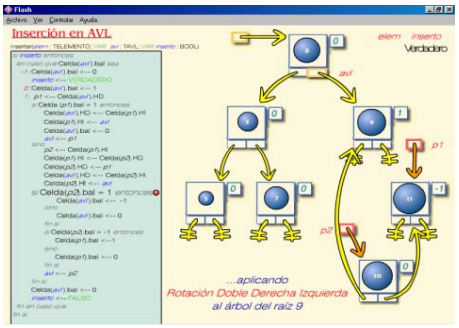


Figura 12

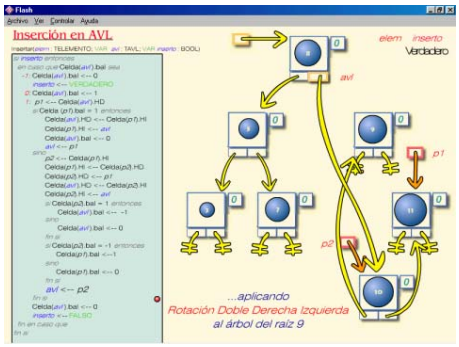


Figura 13

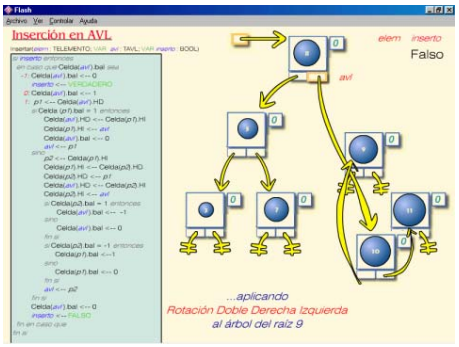


Figura 14

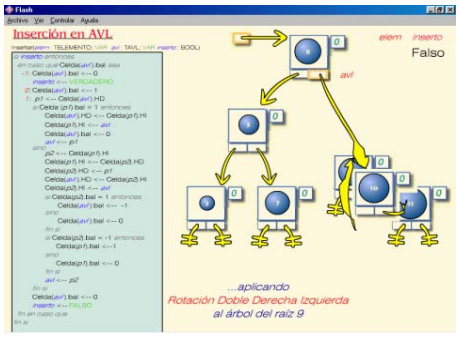


Figura 15

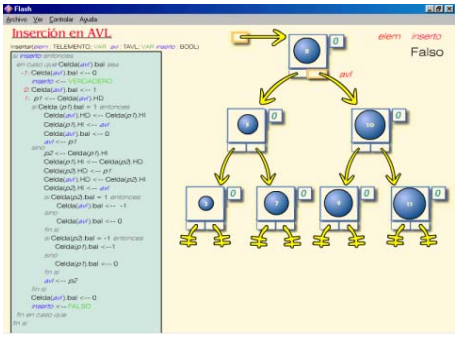


Figura 16

7. Conclusión

Bosque Azul es un software para la Visualización de Árboles Binarios de Búsqueda y de árboles AVL. Está basado fundamentalmente en la *Animación de Algoritmos* y en la *Visualización de Programas* presentando características de una exhibición tanto estática como dinámica. El software

está desarrollado en Macromedia Flash e incorporado a la Web por medio de DreamWeaver, ambos software con gran compatibilidad con Microsoft.
Es una herramienta con posibilidad de ayudar al alumno de un curso de Estructuras de Datos y Algoritmos a aprender los conceptos básicos del tema, a consultar frente a dudas surgidas durante el estudio y a reforzar los conocimientos adquiridos. Este beneficio lo ofrece tanto en la etapa de comprensión de la conducta del algoritmo como en las etapas posteriores de representación del árbol con celdas y enlaces y en la descripción del algoritmo. Además, la animación incremental permite al usuario predecir acciones futuras una vez que ha comprendido el proceso subyacente [10*]. Por otra parte, la exhibición del código, destacando las acciones que se llevan a cabo en cada momento paralelamente a la visualización del efecto que producen sobre la estructura de dato alienta la lectura comprensiva del algoritmo y por ende al desarrollo de algoritmos similares.
El software desarrollado favorece la formación de un cuadro mental del proceso que se lleva a cabo para los estudiantes que se enfrentan por primera vez con el tema y para la ratificación o rectificación del mismo para aquellos que han tenido cierto contacto con los contenidos. Con tal propósito, identifica estructuras, características, anomalías y relaciones entre los datos objeto de la visualización, como así también, presenta un pantallazo cualitativo cuando se trata de conjuntos grandes y complejos de información, y detecta las zonas de interés que merecen un análisis cualitativo focalizado.
La *Visualización de Software* que permita el contacto con representaciones gráficas de algoritmos o estructuras de datos que cambian dinámicamente constituye, sin duda, una posibilidad de exploración de gran valor en la enseñanza de la programación. Bosque Azul es una herramienta diseñada como material de aprendizaje, de consulta y de refuerzo. Tiene como finalidad ayudar a los estudiantes favoreciendo el aprendizaje individual y personalizado, y a los docentes aplicando una metodología de enseñanza presencial, semi-presencial o a distancia [13].

8. Bibliografía

[1] Minsky M. *The Society of Mind.* Ed. Simon and Schuster. 1986.
[2] Clinton L., Jeffrery. *Program Monitoring and Visualization. An Exploratory Approach*. Sprinter-Verlag. 1999.
[3] Price B., Beacker R., Small I., *An Introduction to Software Visualization*. Mit Press. 1998. Sprinter-Verlag. 1999.
[4] Oudshoorn M. j., Widjaja H., Ellershaw S. K., Aspects and Taxonomy of program visualization. Ed. S.K.Chang. 1996.
[5] Stasko, J.- Domingue, J.- Brown, M.- Price, B. *Software Visualization: Programming as a Multimedia Experience.* MIT Press, 1998.
[6] Lawrence, A- Brade, A.- Stasko, J. *Empirically Evaluating the Use of Animations to Teach Algoritms*. Georgia Institute of Technology. 1999.
[7] Weiss M.A., *Estructuras de Datos en Java,* Addison-Wesley, 2000.
[8] Stasko J., Domingue J., Brown M.H. y Price B.A. *Software Visualization. Programming as a Virtual Experience*. The MIT Press. Cambridge. Massachusetts. 1998.
[9] Landreth C. y Rheingans P. *Preceptual Principles for Effective Visualizations*. USEPA. Visualization Center. 1995.
[10] Young P., Munro M., Visualising software in virtual reality. Proceedings of IEEE (IWPC). 1998.
[11] Rheingans P., Landreth C., *Perceptual Principles for Effective Visualizations*. Sprinter-Verlag. 1995.
[12] Brown M., Hershberger J. *Program Auralization*. MIT Press. 1998.
[13] Gal-Ezer J- Lupo D. Integrating internet tools into traditional CS distance education: students' attitudes. Computers & Education. May 2002.
[14] Lupo, D- Erlich, Z. Computer literacy and applications via distance e-learning. Computer & Education, May 2001.
[15] Merril, P. at all. *Computers in education*. Allyn & Bacon. 1996.
[16] Moroni, N- Señas P. *SVED: sistema de visualización de algoritmos*. CACIC 2002. Octubre 2002.