

Algunos Criterios para la Construcción de Animaciones de Algoritmos con Propósitos Pedagógicos

Arturo J. Sánchez Ruíz
Alda F. Pereira Ribeiro

Laboratorio de Construcción de Herramientas Automáticas (AuToolLab)
Centro de Ingeniería de Software Y Sistemas (ISYS)
Escuela de Computación, Facultad de Ciencias
Universidad Central de Venezuela
Apartado 47642, Caracas 1041-A, VENEZUELA

e-mail: {asanchez, apereira}@orquidea.ciens.ucv.ve
http://anubis.ciens.ucv.ve/~asanchez/autoolab.html

Resumen

En este trabajo mostramos algunos criterios que, desde el punto de vista pedagógico, hemos venido usando en la construcción de animaciones a través de la herramienta informática Xtango. Presentamos también dos ejemplos de animaciones, construidas usando estos criterios, en el área de Análisis de Algoritmos. Estas animaciones pueden ser obtenidas a través de la WWW.

1. Introducción

La *Animación de Algoritmos* es una forma de *Visualización de Software* que consiste en usar elementos tales como: movimiento, color, forma y sonido con el objeto de generar representaciones gráficas dinámicas asociadas a conceptos abstractos que definen el comportamiento de un algoritmo dado.

Este concepto tiene implicaciones de gran relevancia en áreas tales como depuración de programas, comprensión de programas/algoritmos y enseñanza de algoritmos, puesto que trata de satisfacer una de las necesidades más elementales del ser humano, vale decir: la necesidad de visualizar conceptos.

Nuestro trabajo está enmarcado en un proyecto cuyo objetivo es el de producir herramientas informáticas para dar soporte activo al proceso de enseñanza-aprendizaje de algoritmos, concepto fundamental en el estudio de las Ciencias de la Computación.

Con todo y lo avanzado de la tecnología informática en la actualidad, vemos que en nuestros países aún se emplean métodos clásicos para la enseñanza de algoritmos [9], es decir, herramientas pedagógicas estáticas tales como: pizarrón, transparencias, rotafolios, etc. Si bien se tiende a utilizar algún lenguaje gráfico *ad-hoc* para representar los conceptos abstractos presentes en los algoritmos (tales como: punteros, registros, variables, árboles, etc.), las exposiciones basadas en estos conceptos presentan una serie de desventajas, por todos conocidas. Podemos mencionar, entre otras, las siguientes:

- El concepto de ejecución es de naturaleza dinámica, concepto difícil de ilustrar usando herramientas estáticas como las mencionadas previamente.
- Las ilustraciones usadas pueden ser inexactas e inconsistentes.
- Dada la heterogeneidad inherente a todo grupo de estudiantes respecto al ritmo de comprensión, existe la posibilidad de que algunos encuentren las exposiciones aburridas, algunos complejas y otros confusas.
- Es posible que las ilustraciones escogidas para representar conceptos abstractos no capturen adecuadamente la naturaleza de los mismos.
- Existe también la posibilidad de que una exposición aborde sólo algunos de los aspectos asociados a la ejecución de los algoritmos.

Estos argumentos reflejan que las exposiciones basadas en esquemas tradicionales pueden no ser efectivas para transmitir los conceptos básicos asociados a los algoritmos. Tenemos entonces que la animación de algoritmos representa una alternativa atractiva y efectiva que puede ser utilizada en el proceso de enseñanza-aprendizaje de los mismos. Adicionalmente, el hecho de que el computador sea la plataforma sobre la cual se realizan las animaciones trae consigo una serie de ventajas que han sido señaladas por algunos autores, entre las cuales podemos citar [8]:

- Consistencia en la presentación.
- Neutralidad respecto a patrones de evaluación.
- Flexibilidad en cuanto al ritmo individual de cada estudiante.
- Concurrencia en el uso de las herramientas.
- Rango de disponibilidad verdaderamente amplio.

Existen varias herramientas informáticas para la creación de animaciones de algoritmos. Entre ellas podemos mencionar: Balsa [3], Zeus [4], StarLite [6] y Xtango [10, 11] (también puede verse <http://www.research.digital.com/SRC/zeus/home.html>). Para construir las animaciones que se presentan en este trabajo hemos escogido Xtango pues pensamos que: ofrece una plataforma que soporta una serie de conceptos primitivos útiles para la creación de animaciones, es fácilmente transportable, no exige mucho del sistema donde debe ser instalado, puede utilizarse en la *World Wide Web* (WWW) y, finalmente, el modelo conceptual que utiliza es bastante sencillo lo cual hace que la construcción de

animaciones pueda verse como la construcción de un programa en C que hace invocaciones a una librería de programas.

En este trabajo presentamos nuestra experiencia en el uso de Xtango para la creación de animaciones con propósitos pedagógicos, la cual está enmarcada en nuestro proyecto de creación de libros electrónicos [1] a ser usados en los cursos de nuestra *Licenciatura en Computación*, en el contexto de la WWW. En la Sección 2 presentamos los criterios que hemos venido utilizando para la creación de las animaciones. Luego, en la Sección 3, presentamos dos ejemplos de animaciones que aparecerán en el libro electrónico a ser usado en un curso de *Análisis de Algoritmos*. La Sección 4 presenta un resumen de la experiencia realizada hasta ahora. Concluimos, en la Sección 5, con nuestras sugerencias para la continuación de la investigación.

Todas las animaciones aquí presentadas, y otras que hemos construido, pueden obtenerse de <http://anubis.ciens.ucv.ve/~asanchez/courseware.html>. La herramienta Xtango puede encontrarse en [12].

2. Criterios Usados en las Animaciones

Tal como comentamos anteriormente, es importante utilizar tanto un lenguaje (textual/gráfico) como un protocolo consistente para así lograr un canal de transmisión de conocimientos con el menor ruido posible. Con esta motivación en mente, hemos seguido una serie de criterios para la construcción de las animaciones, los cuales son discutidos en esta sección.

2.1 El Protocolo Causa-Efecto

Todas nuestras animaciones están basadas en un protocolo que hemos denominado causa-efecto y que no es más que la implantación en el computador de la metáfora de la tiza y el pizarrón, vale decir: para explicar el comportamiento del algoritmo partimos del código que lo describe y vamos visualizando el efecto de ejecutar cada una de las instrucciones. La pantalla del computador se transforma en el pizarrón en el cual se va mostrando, de manera dinámica, qué instrucción se está ejecutando y al menos una representación gráfica del efecto asociado a la ejecución de tal instrucción. A esta representación gráfica usualmente se le llama *vista* [3, 4]. Tenemos entonces que el protocolo establece que, para cada instrucción, debe mostrarse al menos una vista asociada a su ejecución. Las preguntas importantes que debe responder e instrumentar el animador son: ¿Qué vistas deben ser mostradas?, ¿Cómo construir las vistas con las herramientas que la plataforma provee?

2.2 Atributos de los Objetos Usados

Sin pérdida de generalidad uno puede concebir una vista como un conjunto de objetos gráficos que guardan cierta relación entre sí. Los *atributos* de estos objetos definen su apariencia en la animación. Dos atributos que hemos usado en nuestras animaciones son el color y la forma. El uso del color se hace evidente cuando éste forma parte inherente del

algoritmo utilizado. Podemos citar aquí el caso de árboles rojo/negro [5]. Volveremos a este ejemplo en la Sección 3.2. Otro uso que le hemos dado al color en nuestras animaciones es el de mostrar una cierta propiedad cuando el cómputo pasa de un estado al otro. Un esquema que aparece frecuentemente en soluciones algorítmicas es el de efectuar iteraciones manteniendo una condición *invariante*. Podemos entonces colorear los diferentes objetos gráficos en una vista de forma tal que la evolución de la condición invariante (y por tanto del cómputo) se haga evidente. Otros usos del color pueden verse en [4].

En cuanto a la forma, hemos usado el grosor para indicar la rapidez relativa con la cual las variables de control (implícitas o explícitas) de un ciclo iterativo varían. Por ejemplo, en un par de ciclos anidados, el objeto asociado a la variable de control del ciclo más externo será más grueso que el del ciclo más interno, lo cual da la apariencia de que el último se “mueve” más rápido que el primero. También hemos usado líneas punteadas para representar un objeto “vacío” o inexistente (por ejemplo, un subárbol vacío). Finalmente, utilizamos el centelleo de un objeto para indicar que éste está a punto de ser manipulado en el algoritmo.

2.3 Uso de Vistas Diferentes

El protocolo causa-efecto que se describió antes hace que la vista del código deba estar presente. La forma de mostrarlo consiste en utilizar alguna notación algorítmica adecuada para el código el cual está presente (en general) en toda la animación. La instrucción que se está ejecutando es enmarcada por un recuadro y las posibles partes que ésta pueda tener centellean cada vez que está por mostrarse el efecto de su ejecución. Por ejemplo, si una instrucción condicional está por ejecutarse, primero se enmarca y luego, a medida que la condición lógica se va evaluando, sus términos centellean para así ubicar la evaluación de la misma en otras vista existentes. Si el código es relativamente complejo, es conveniente mostrarlo en diferentes niveles de detalle. Por ejemplo, un primer nivel contendrá un tratamiento por casos y sólo cuando cada caso está por ocurrir se muestra el código correspondiente. Si el código está estructurado en llamadas a procedimientos es conveniente mostrar el contexto de ejecución a través de una vista que muestre el contenido de los registros de activación y la pila (*stack*) de ejecución, lo cual puede ser particularmente útil al visualizar procedimientos/funciones recursivos(as).

En algunas aplicaciones puede ser importante llegar al nivel de detalle de visualizar las comparaciones que se efectúan. Esto lo hemos implantado haciendo centellear (en el código y/o en las vistas existente) los operandos indicados para luego conectarlos con el operador relacional en cuestión el cual muestra el resultado de la evaluación.

Una forma de comparar distintos algoritmos que resuelven un mismo problema es a través del estudio de una cierta propiedad. Es necesario entonces incluir en cada vista elementos que ayuden a visualizar si esta propiedad se cumple o no. Un ejemplo que mostraremos luego (en la sección 3.1) es el de visualizar si un algoritmo de ordenamiento posee o no la propiedad de ser estable.

Al visualizar las variables en el algoritmo es importante utilizar objetos que estén “naturalmente” asociados a éstas. Si bien este es un concepto muy subjetivo, existen representaciones que son parte del “folklore” de la algorítmica. Por ejemplo, existe una manera estándar (de facto) de representar objetos como arreglos y árboles. Nosotros hemos usado flechas erectas para visualizar cursores en un arreglo, flechas en forma de zig-zag para representar punteros, etc. Algunas veces es suficiente el simplemente rotular parte de una vista con el nombre de la variable cuyo valor se quiere visualizar. Por ejemplo, colocar el nombre como rótulo de un nodo en una vista que representa un árbol indica que el valor de la variable es el subárbol que tiene como raíz al nodo en cuestión. En algunos casos es importante visualizar no sólo variables que aparecen implícitamente definidas en el algoritmo, si no que además deben mostrarse valores que definen contextos o regiones. Por ejemplo, si en el algoritmo aparece una variable i , quizás es importante no sólo visualizar su valor si no también el del contexto dado por los valores de $i-1$ e $i+1$.

Existen algunos algoritmos que, por su complejidad, tratan el dominio de los objetos que manipulan por casos. Una forma, que consideramos efectiva, de capturar la esencia de cada caso es a través de la visualización de un patrón general que lo describa y la forma de determinar cómo la vista de la ejecución es una instancia del patrón. Esto lo ilustraremos en la sección 3.2.

Para concluir esta sección, consideramos pertinente indicar que se debe dar flexibilidad al usuario de la animación para que éste escoja las vistas que desea visualizar, entre aquellas que estén disponibles. De esta forma, si bien el protocolo causa-efecto está basado en el código como origen o punto de referencia de la animación, es perfectamente válido que el usuario pida no ver el código (o cualquier otra vista) pues, en usos posteriores, está interesado en una en particular. Esto lo hemos incorporado a nuestras animaciones a través del mecanismo simple de preguntar al usuario si desea o no ver una vista determinada. Esto se ilustra en la sección 3.1.

3. Ejemplos de Animaciones

Para realizar una animación en Xtango se debe comenzar por crear dos programas en C: aquel que contendrá el algoritmo que se va a animar, y el que contendrá el código de la animación usando las herramientas que Xtango provee.

La tarea de construir la animación de un algoritmo empieza colocando invocaciones a la función denominada `TANGOalgoOp`, en las posiciones apropiadas del programa correspondiente al algoritmo a animar. Esta función permitirá la activación de las rutinas de animación requeridas.

Para que `TANGOalgoOp` pueda desempeñar cabalmente su papel requiere de la creación de una estructura de datos que contendrá las escenas de animación que son invocadas por tal función a través de un nombre de operación dado como parámetro. El formato que toma esta función es el siguiente:

```
TANGOalgoOp &(Nombre Estructura, &Nombre Operación,  
             &Parámetros Operación)
```

La estructura mencionada anteriormente se le denomina `NAME_FUNCT`, y es creada por el animador. La forma más fácil de definirla es inicializándola estáticamente al comienzo del programa que contiene el algoritmo a animar. Su formato se indica a continuación:

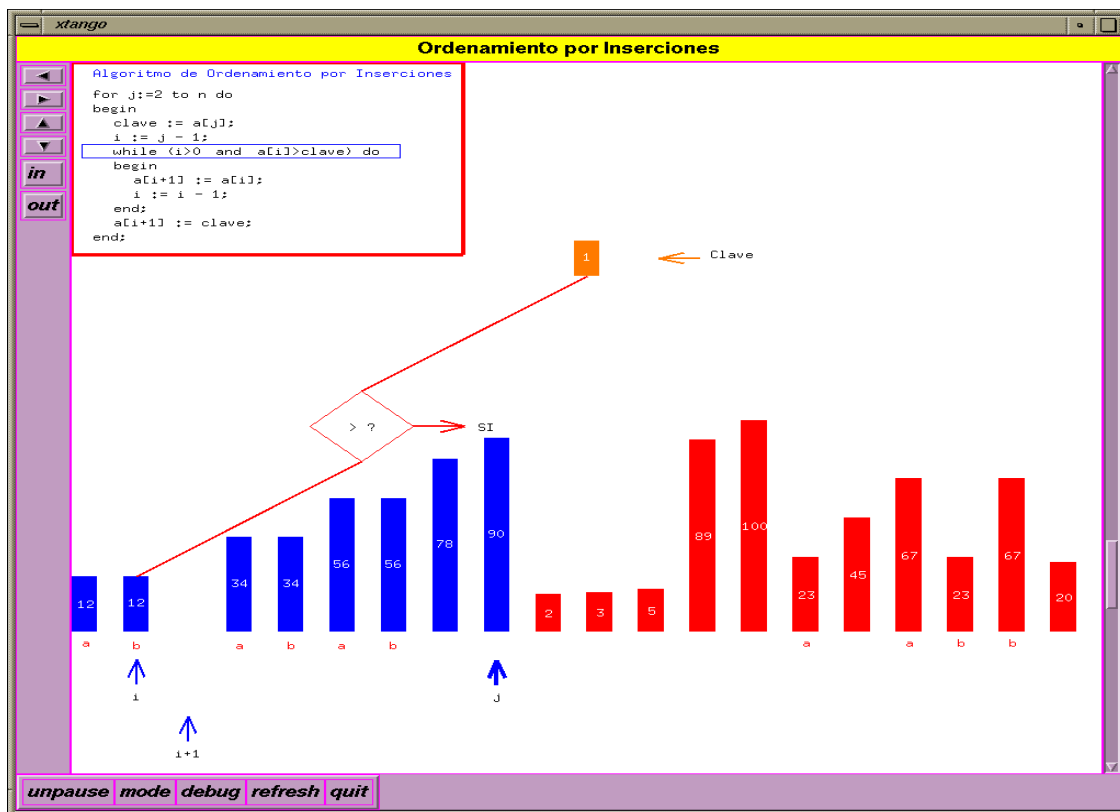
```
static NAME_FUNCT nombre de estructura [] = {  
{ nombre de operación, número de escenas de animación, { lista de escenas de animación } }  
  ..  
{ NULL, 0, NULL, NULL } }
```

Antes de hacer cualquier invocación a alguna operación, se debe invocar a “Begin”, la cual producirá la inicialización de Xtango. Al final debe invocarse a “End” como la última función en el algoritmo a animar, esta se asegurará que todas las escenas de animación hayan sido mostradas y espera hasta que el botón “quit” sea seleccionado para finalizar la animación. Una vez ubicadas las llamadas que permitirán el despliegue de la animación, se procede escribir el código de las rutinas en C (Escenas de Animación) correspondientes a tales llamadas. Para desarrollar estas rutinas se hace uso de los distintos tipos de datos que provee Xtango: `TANGO_LOC`, `TANGO_IMAGE`, `TANGO_PATH` y `TANGO_TRANS`; a través de las funciones o procedimientos que permiten su manipulación y así crear las distintas animaciones deseadas.

Finalmente, es importante mencionar las rutinas denominadas `ASSOCIATIONS`, las cuales constituyen el intermediario entre los datos de un programa y los objetos de animación asociados con esos datos. Estas rutinas son útiles para el manejo de los distintos objetos involucrados en la animación y se crean tantas como sean necesarias.

3.1 Ordenamiento por Inserciones

Tal como se indicó anteriormente, la animación para este algoritmo está basada en el protocolo causa-efecto. De esta forma, primero se destaca la instrucción del algoritmo a ejecutar para luego visualizar las correspondientes transformaciones que se llevan a cabo cuando ella se ejecuta. Puede verse una imagen congelada de esta animación en la figura que se muestra a continuación.



Con respecto a los atributos de los objetos usados en esta animación, el color resulta ser de gran utilidad pues permite capturar una invariante fundamental del algoritmo, es decir: en la j -ésima iteración la región $[1, j-1]$ del arreglo se encuentra ordenada ($j \geq 2$). De esta forma:

- Los elementos ya ordenados son coloreados de azul.
- Los elementos por ordenar son coloreados de rojo.
- El elemento cuya posición en la región $[1, j]$ debe calcularse, es coloreado de naranja.

En cuanto a la forma de los objetos, ésta se emplea con la finalidad de asociar el grosor de los cursores (representados como flechas) con un ciclo en particular del algoritmo, de esta forma se pretende captar la velocidad con la cual se mueven las variables asociadas a cada uno de ellos. En este caso, la flecha gruesa (asociada con el índice j) se corresponde con el ciclo externo y las otras flechas (menos gruesas) con el ciclo interno. Otro de los atributos utilizados es el “parpadeo” de los objetos implicados en el chequeo de las condiciones del algoritmo.

En esta animación se diseñaron varias vistas, siendo dos de ellas las principales: una que contiene el código del algoritmo y la otra que despliega las imágenes asociadas al arreglo de entrada del algoritmo. En ambas vistas se producen diferentes acciones. En la vista que contiene el código, se va enmarcando la instrucción que se está ejecutando en un momento dado y en la otra vista se van produciendo los diferentes cambios que conlleva la acción asociada a tal instrucción. Eventualmente aparecerán otras vistas que ilustran diversas

operaciones del algoritmo: por ejemplo, una pequeña vista mostrará el elemento clave en un instante dado y el cual es motivo de análisis.

En otras vistas se ilustran las comparaciones entre los elementos, derivadas del chequeo de las condiciones en el ciclo interno del algoritmo. En otra se puede apreciar una propiedad importante del algoritmo: su estabilidad. Esto se logró rotulando los elementos iguales con letras, de forma que una vez que el algoritmo termina de ejecutarse es posible ver claramente si el algoritmo no es estable (o conjeturar que es estable para luego proceder a demostrarlo).

En lo relativo a la representación utilizada para las variables principales del algoritmo, éstas fueron las siguientes: el arreglo de elementos fue representado por un grupo de rectángulos, donde el tamaño de cada uno está relacionado con la magnitud del elemento asociado. Los índices de los ciclos (i, j) fueron caracterizados en la animación de forma textual y gráfica. De la misma forma es identificado el elemento clave.

Cabe mencionar, que tanto la observación del código, como las comparaciones y la estabilidad son vistas opcionales, dependiendo de la elección que haga el usuario al comienzo de la animación.

3.2 Inserción en árboles Rojo/Negro

La animación del algoritmo de inserción en árboles rojo/negro fue construida a partir de una versión sencilla implantada por Aaron Candib del *Georgia Institute of Technology*, la cual viene con la distribución de Xtango [12]. Entre los aspectos que encontramos apropiados para ser agregados a esta animación están los siguientes: obtener varias vistas en la ventana de animación, desplegar el código del algoritmo, darle forma circular a los nodos del árbol (con el color correspondiente) y añadir otras pequeñas vistas que ayuden a una mejor comprensión del algoritmo, dado que este es un tanto complejo. Una imagen congelada de esta animación se muestra a continuación.

Al igual que en la animación anterior, los cambios que se realizaron están basados en el protocolo causa-efecto. De manera que siempre se muestra el código.

Los colores utilizados en la animación están sugeridos de forma directa por la naturaleza de los árboles que manipula el algoritmo. En cuanto a la forma de los objetos, las líneas punteadas han sido utilizadas en este caso con la intención de relacionar un nodo inexistente con su ancestro inmediato. Al igual que en la animación anterior, se hace uso del “parpadeo” de los objetos para denotar las diferentes condiciones existentes en el algoritmo.

En esta animación se diseñaron varias vistas, una de ellas comprende el código del algoritmo a animar, que a su vez se divide en otras pequeñas vistas que contienen pedazos de código correspondientes a los diferentes casos que pueden ocurrir durante la ejecución del algoritmo. Así mismo, por cada caso se originan otras subvistas que despliegan el esquema general del caso y una descripción textual del mismo.

Por tratarse este algoritmo sobre la inserción en un árbol, éste es visualizado usando una representación gráfica que es un estándar de facto. Los valores importantes: x , $p[x]$, y y $p[p[x]]$ son visualizados en forma textual rotulando con sus nombres los nodos correspondientes.

4. Resumen de la Experiencia

Hemos presentado una serie de criterios que hemos aplicado en la construcción de animaciones con propósitos pedagógicos. Estos criterios han sido el resultado de nuestra experiencia en el área de enseñanza de la algorítmica en los niveles de pre y post-grado. Están basados en la metáfora sencilla de la tiza y el borrador usando el protocolo causa-efecto.

Comparando las animaciones que hemos discutido con otras que han aparecido en la comunidad, pensamos que la aplicación de nuestros criterios nos ha llevado a construir animaciones que capturan las ideas centrales subyacentes a los algoritmos en cuestión. En particular, pensamos que nuestra animación de los árboles rojo/negro (ver sección 3.2) posee vistas más útiles en el proceso de enseñanza-aprendizaje que la provista con la distribución de Xtango [12]. Entendemos, por supuesto, que la idea original de estas animaciones, incluidas con la distribución de Xtango, es la de ilustrar cómo usar la herramienta, más que el de mostrar buenos ejemplos desde el punto de vista pedagógico. Reconocemos, además, la gran utilidad de los objetos que provee Xtango para manipular árboles binarios, lo cual hace fácil dibujarlos, rotarlos, etc.

En nuestra opinión, la animación asociada al algoritmo de ordenamiento por inserciones presentado en la sección 3.1, supera en cuanto a sus características pedagógicas a la presentada en [2]. Esta comparación muestra la ventaja de usar herramientas informáticas sobre plataformas con ambientes gráficos robustos versus el uso de herramientas sobre plataformas tipo PC.

Hemos producido otras animaciones, tales como: algoritmo de ordenamiento por mezclas sucesivas (*merge sort*), *radix sort*, y tablas de *hashing*. Todas estas disponibles desde <http://anubis.ciens.ucv.ve/~asanchez/courseware.html>.

5. Continuación de la Investigación

Pensamos que es perfectamente factible el utilizar las herramientas informáticas disponibles en la comunidad internacional para producir herramientas pedagógicas de alta calidad y utilidad a un costo prácticamente nulo.

Desde el punto de vista del usuario de estas herramientas, nuestras actividades están enfocadas hacia la realización de libros electrónicos que sirvan de apoyo a los textos clásicos en ciertas áreas. Estos libros pueden abarcar fases que van desde la explicación de conceptos hasta su evaluación. Las áreas en las que estamos interesados son: Análisis de Algoritmos, Introducción a la Construcción de Algoritmos, Teoría de Lenguajes Formales, Construcción de Compiladores, Algoritmos Paralelos/Distribuidos y Matemáticas Discretas. Es importante poder enmarcar estos libros electrónicos en ambientes como la WWW. Un proyecto interesante, en el entorno iberoamericano, sería la realización conjunta de este tipo de textos, disponibles a través de la WWW. Pensamos además, seguir afinando nuestros criterios, incorporar el sonido como elemento en las animaciones y, muy

importante, evaluar la repercusión de este tipo de herramientas en el proceso enseñanza-aprendizaje [7].

Desde el punto de vista del creador de nuevas herramientas, existen problemas muy interesantes que, a nuestro entender, no han sido abordados por otros científicos del área. Entre ellos podemos mencionar los siguientes: creación de una herramienta que permita al animador definir los objetos que representan una abstracción dada y su conexión con eventos importantes en el algoritmo. El que el usuario/animador pueda cambiar dinámicamente las representaciones gráficas asociadas a los conceptos. Finalmente, nos interesa experimentar con el uso de la metáfora asociada a la *realidad virtual*.

Referencias

- [1] Ph. Baker. Electronic Books and Libraries of the Future. *The Electronic Library*, 10(3):139-149. 1992.
- [2] P. R. Borges. Una Herramienta para el Aprendizaje de Algoritmos de Ordenamiento. *En Actas de Panel ´94*, pp. 413-419. 1994.
- [3] Brown. Exploring Algorithms Using BALSAM. *IEEE Computer*, 21(5):14-36. 1988.
- [4] M. Brown, J. Hersberger. Color and Sound in Algorithm Animation. *IEEE Computer*, 25(12):52-63. 1992.
- [5] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest. *Introduction to Algorithms*. Cambridge, Mass. MIT Press/McGraw-Hill. 1991.
- [6] R. P. Cook, R. McDaniel. The StarLite Algorithm Animator. *Software-Concepts and Tools*, 16(1):1-11. 1995.
- [7] A. W. Lawrence, A. N. Badre, J. S. Stasko. Empirically Evaluating the Use of Animations to Teach Algorithms. Technical Report GIT-GVU-94-07. Georgia Institute of Technology. 1994.
- [8] J. Mincy, A. Tharp, K. Tai. Visualizing Algorithms and Processes with the Aid of a Computer. *ACM SIGSE Bulletin*, 15(1):106-111. 1983.
- [9] Ponencias del 3rd Workshop on Education in Computer Science y del 4th Ibero-American Congress on Computer Science Higher Education. Celebrados en Panel ´95, Canela (Brasil), Julio-Agosto 1995.
- [10] J. Stasko. Path Transition Paradigm: A Practical Methodology for Adding Animation to Program Interfaces. *Journal of Visual Languages and Computing*, 1(3):213-236. 1990.
- [11] J. Stasko. TANGO: A Framework and System for Algorithm Animation. *IEEE Computer*, 23(9):27-39. 1990.
- [12] URL <http://www.cc.gatech.edu/gvu/softviz/algoanim/algoanim.html>