XV JENUI. Barcelona, 8-10 de julio de 2009
ISBN: 978-84-692-2758-9
http://jenui2009.fib.upc.edu/

Animación automatizada de técnicas de diseño de algoritmos

Antonio Pérez Carrasco, J. Ángel Velázquez Iturbide
Departamento de Lenguajes y Sistemas Informáticos I
Escuela Técnica Superior de Ingenería Informática, Universidad Rey Juan Carlos
C/ Tulipán s/n, CP 28933, Móstoles (Madrid)
antonio.perez.carrasco , angel.velazquez@urjc.es

Resumen

Existe la creencia generalizada de que las visualizaciones empleadas durante la enseñanza de algoritmos ayuda al alumno a asimilar más rápidamente y con mayor grado el funcionamiento de los mismos. Es por ello que en los últimos dos años hemos implementado una aplicación, SRec, que ofrece soporte para la generación automatizada de visualizaciones de programas con posibilidad de animación que asisten al profesor en sus clases magistrales así como al alumno en la elaboración de ejercicios y estudio de la algoritmia. Exponemos nuestra motivación, las aplicaciones implementadas y una serie de resultados obtenidos que justifican nuestro trabajo.

1. Introducción

Las visualizaciones han sido siempre consideradas como un elemento de interés para el aprendizaje de la algoritmia. Por su parte, las visualizaciones animadas permiten dar un salto hacia adelante frente a representaciones estáticas al suponer una fuente de información más completa, que permite ver de una forma clara el transcurso de la ejecución de los algoritmos.

Un modo de conseguir mayor efectividad en el aprendizaje es involucrar al alumno en la animación mediante la interactividad. Ésta se perfila como el concepto que permite al alumno sumergirse en la animación para facilitar la comprensión de la misma.

Los alumnos que interactúan con animaciones suelen obtener mejores resultados académicos que aquellos que desarrollan actitudes pasivas frente a las visualizaciones [3].

La interacción con las animaciones supone una conexión entre el algoritmo y el alumno que ayuda al aprendizaje gracias a que se logra asimilar la teoría desde un punto de vista totalmente práctico. La posibilidad de reproducir una animación tantas veces como se necesite, y la funcionalidad de seleccionar la cantidad y el tipo de información que se muestra en cada momento permiten tanto al profesor como al alumno centrarse en determinadas áreas de los algoritmos en cada momento, según las necesidades de la asignatura y/o del alumnado.

Es por ello que se ha desarrollado una aplicación que intenta satisfacer esta demanda. La aplicación, cuyo nombre es SRec, aporta varias vistas complementarias sobre la ejecución de programas recursivos. Estas vistas son específicas para cada técnica de diseño considerada. La aplicación cuenta con múltiples opciones y facilidades educativas que convierten su manejo en una actividad sencilla e intuitiva, minimizando su tiempo de implantación y aprendizaje, facilitando así la labor docente de manera notable.

2. Motivación

En las últimas tres décadas no se ha conseguido generalizar el empleo de visualizaciones de programas debido a la falta de garantías sobre su eficacia en la docencia y al excesivo esfuerzo necesario para generarlas por parte del profesorado.

Si bien las visualizaciones más expresivas son las dependientes del dominio, las visualizaciones genéricas creadas de manera automática a partir del código permiten obtener información acerca de la técnica de diseño con la que fue creado el algoritmo.

No existe una literatura extensa que se enfoque en las visualizaciones basadas en la técnica de diseño salvo [2][1], terreno en el que se adentra la aplicación que aquí se presentará más adelante.

Para determinar cuáles son las visualizaciones que se desarrollarían en la aplicación, se partió de la hipótesis de que las visualizaciones de mayor calidad debían de

haber sido usadas en libros de texto con anterioridad. Haciendo un repaso por más de una decena de conocidos libros de texto, donde se observó que reinaban las visualizaciones dependientes del dominio, que ayudan a entender fácilmente problemas concretos, pero que no ofrecen información específica sobre la técnica de diseño empleada.

Además, se ha constatado que los alumnos son receptivos al empleo de visualizaciones durante las clases, al encontrar en ellas un útil complemento a las explicaciones del profesor.

3. Características de SRec

3.1. Objetivos

Los objetivos iniciales de SRec eran básicamente tres. El primero pasaba por mitigar el excesivo esfuerzo que suponía para los profesores crear nuevas visualizaciones, que suponía un fuerte obstáculo para su empleo, el cual a veces era salvado con visualizaciones de poca calidad, claridad y utilidad.

El segundo era responder a la demanda del alumnado de tener a su disposición la posibilidad de elaborar sus propias visualizaciones para el desarrollo de ejercicios, prácticas y ejemplos. El tercer objetivo era proporcionar una serie de funcionalidades que permitan al alumno centrarse en diferentes aspectos del algoritmo (parámetros, repetición de llamadas, etc.).

Para todo ello, la aplicación diseñada emplea una ventana que ofrece varias vistas. En la figura 1 se puede ver para la recursividad la vista de código (a), la de la traza (b), la de la pila de control (c) y la del árbol de recursión (d).

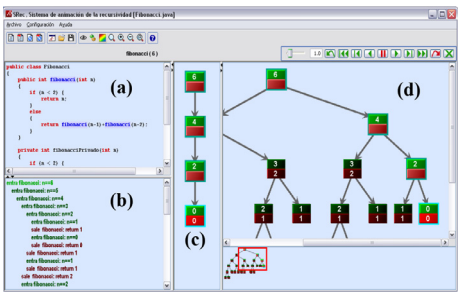

Figura 1. Ventana principal de SRec

3.2. Procesado de clases Java

La aplicación admite como entrada una clase Java, que contendrá al menos un método recursivo. El usuario podrá cargar la clase que desee eligiéndola a través del sistema de archivos en la aplicación.

SRec realizará un complejo proceso de análisis y modificación para crear una nueva clase que contenga los mismos algoritmos pero con sentencias adicionales que permitan a la aplicación crear una traza en la cual se basarán posteriormente las visualizaciones.

Para ello se traduce la clase Java original a un documento XML, se carga en memoria y se introducen estratégicamente mediante la API de DOM nuevos elementos XML que representan las sentencias adicionales. Estas sentencias adicionales recogen los valores iniciales de los parámetros y el valor de retorno de cada llamada recursiva.

Una vez que se han hecho las modificaciones oportunas, se vuelve a traducir a lenguaje Java, por lo que se obtiene una clase equivalente, que será la que finalmente se ejecute. El proceso completo queda ilustrado en la figura 2.

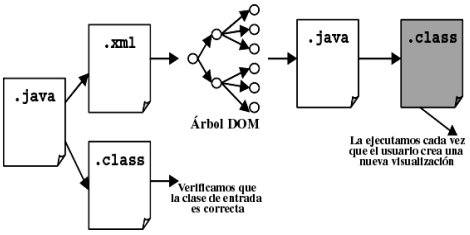

Figura 2. Procesado de clases

Cuando el usuario desee generar una nueva animación, la aplicación le mostrará un listado de los métodos disponibles. Tras seleccionar un método e introducir los valores adecuados para sus parámetros, la aplicación ejecutará completamente el algoritmo y mostrará a continuación las vistas asociadas.

En ese momento, la animación se situará al comienzo de la llamada inicial, mostrando tan sólo el nodo raíz del árbol de recursión.

3.3. Navegación por las animaciones

El usuario puede mediante la barra de animación (presentada en la figura 3 y explicada en la tabla 1) activar la función de animación automática, adelantar o retroceder la animación paso a paso de forma manual, realizar saltos sobre la animación, etc.

Botón	Descripción de su función
	Da un salto de subllamadas hacia atrás
	Retrocede la animación al estado inicial
	Da un paso hacia atrás
	Activa la animación automática hacia atrás
	Pausa la animación automática
	Activa la animación automática hacia delante
	Da un paso hacia delante
	Avanza la animación al estado final
	Da un salto de subllamadas hacia delante
	Cierra la animación

Tabla 1. Valores del cuestionario de la sesión 1

Figura 3. Barra de animación

Los controles de SRec permiten también graduar la velocidad de las animaciones automáticas para ajustarlas al ritmo deseado.

La barra de animación tiene unos mandos similares a los de un reproductor multimedia, por lo que su aprendizaje es rápido e intuitivo.

3.4. Funcionalidades

La aplicación da la opción de seleccionar qué nodos se muestran y cuáles se ocultan en función del método al que representan. Se pueden marcar y desmarcar igualmente los parámetros de los métodos visibles, lo que permite estudiar diferentes aspectos de los algoritmos (valores de índices, datos...).

La aplicación tiene implementada una funcionalidad de zoom que ofrece la posibilidad de ajustar en tamaño el árbol visible en ese momento al panel de la ventana. También se puede graduar numéricamente el zoom del árbol y de la pila según las necesidades que se tenga en cada momento.

Tal y como se aprecia en la figura 4, el usuario puede configurar SRec para que atenúe el color de los nodos que representan llamadas cuya ejecución ha finalizado.

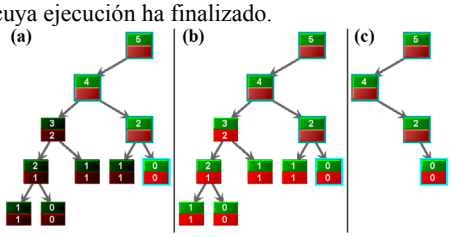

Figura 4. Modos de visualización de nodos de subllamadas ya finalizadas

SRec puede atenuar los nodos de subllamadas ya finalizadas (figura 4, a), mantenerlos en el mismo color que aquellos que representan subllamadas aún en ejecución (figura 4, b) o hacerlos desaparecer de la vista (figura 4,c).

SRec da la facilidad de cargar y guardar animaciones en un formato propio XML con el fin de poder crear una colección de animaciones que eviten al usuario generar cada animación cada vez que se necesite. Cuando el usuario, en otra sesión de uso de la aplicación, cargue esa animación creada por SRec, ésta aparecerá en la ventana la animación situada en el mismo estado y con la misma apariencia que en el momento en que fue guardada. Esto facilita que un profesor pueda mostrar un ejemplo complejo en clase sin necesidad de generarlo durante la misma o que un alumno pueda preguntar una duda concreta sobre un ejemplo creado.

SRec presenta por otra parte la capacidad de generar capturas de la vista del árbol de recursión. De esta forma, se pueden generar imágenes en formato electrónico que pueden incrustarse posteriormente en documentos de ofimática y páginas web, permitiendo a los profesores elaborar material docente y a los alumnos informes de prácticas o proyectos de estudio de una manera mucho más sencilla, cómoda y elegante.

Además, SRec también es capaz de generar animaciones estándar en formato GIF, animaciones que pueden ser empleadas en transparencias o páginas web, aunque no soportan interactividad.

Por último, cabe destacar que SRec incluye un sencillo editor de código que permite al

alumno depurar rápidamente el código que está visualizando. Para ello habilita una ventana adicional (figura 5) que presenta una barra de herramientas, un área de edición y un área de compilación, donde aparecerán los errores que el compilador de Java encuentre en el código introducido cuando el usuario decida emplearlo.

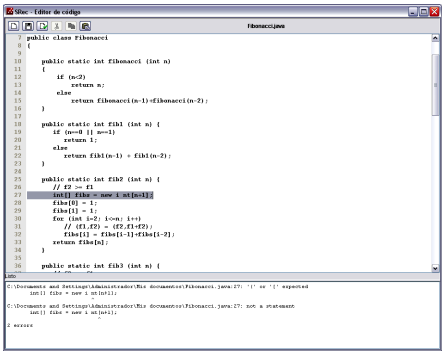

Figura 5. Ventana de edición de código

4. Visualización de la recursividad

Antes de comenzar a implementar un primer prototipo de la aplicación fue necesario especificar qué técnica de diseño se iba a representar y cuáles eran los tipos de visualizaciones que se pretendían mostrar a través de la aplicación. Así, se decidió que se comenzaría a trabajar con la recursividad, aportando para ella un total de tres vistas.

Estas vistas, que se muestran en la ventana de SRec junto con la vista estática del código de la clase cargada, ofrecen información complementaria totalmente sincronizada. La primera de ellas tiene formato textual y muestra la sucesión de subllamadas recursivas realizadas y valores obtenidos durante el transcurso de la ejecución del algoritmo. Es decir, se puede considerar como una traza que permite seguir la ejecución paso a paso de todo el algoritmo.

La segunda de las vistas muestra en todo momento el estado de la pila de control; esto es, el conjunto de subllamadas recursivas cuya ejecución aún no ha finalizado. La apariencia de los nodos que se muestran es siempre idéntica a la vista principal, la del árbol de recursión, con el fin de poder establecer más fácilmente la identificación de las subllamadas representadas. Esta vista ayuda a controlar el uso de memoria que realiza el algoritmo.

Por último, la aplicación muestra una tercera vista asociada a la recursividad, la de mayor importancia y mayor cantidad de información suministrada, que contiene el árbol de recursión.

Esta vista permite seguir la ejecución completa del algoritmo gracias a que recoge todas las subllamadas mostrando las dependencias entre ellas. Cada nodo del árbol aporta los valores de los parámetros de entrada para esa subllamada y también el valor obtenido tras la ejecución de la misma en dos celdas diferenciadas situadas una sobre la otra, respectivamente.

Como se puede apreciar en la secuencia de la figura 6, donde se ve una parte de la ejecución del algoritmo de Fibonacci, al activar una llamada recursiva sólo resultan visibles sus parámetros de entrada. Posteriormente se van activando los nodos de sus subllamadas recursivas. A medida que se van obteniendo resultados en los casos base (que aparecen en la parte inferior de cada imagen de secuencia) éstos se van mostrando hasta resolver la llamada que los generó (que aparece en la parte superior).

La llamada que actualmente está siendo objeto de resolución consta de un marco que la diferencia del resto (en la figura 6 este marco es de color claro), mientras que las llamadas que están pendientes de resolverse están contenidas en un marco de otro color (más oscuro en la figura 6). Estos marcos ayudan a identificar los mismos nodos en la vista de pila ya comentada, donde también aparecen estas celdas con el mismo formato.

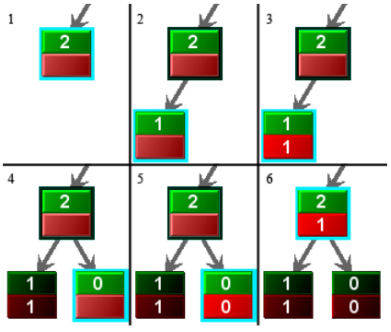

Figura 6. Transcurso de la animación

La vista del árbol de recursión incorpora un visor de navegación que ayuda a ver el contexto de la porción de árbol que aparece en la vista. Así, en su interior alberga un marco rojo que enfoca la parte que se ve en la vista. Mover con el ratón este marco permite trasladar la vista automáticamente a donde quede situado el marco, lo que facilita la visualización de partes concretas del algoritmo.

5. Visualización de la técnica "Divide y vencerás"

La recursividad es, sin duda, una de las herramientas más empleadas en el diseño de algoritmos, si bien es cierto que existen otras técnicas de gran importancia como la programación dinámica o la comúnmente denominada "Divide y vencerás".

Esta técnica, basada en la recursividad, se suele aplicar sobre estructuras de datos de una o varias dimensiones (arrays y matrices fundamentalmente), que van sufriendo un proceso de división hasta alcanzar un caso sencillo abordable de manera directa sin dificultad.

Es por ello que se ha trabajado en la visualización de esta técnica aportando vistas específicas que acerquen a los alumnos a esta técnica de la manera más intuitiva y clara posible.

La primera de las vistas es el árbol de recursión. Este árbol presenta un comportamiento muy similar al presentado para la recursividad, pero deja ver con detalle el estado de la estructura de datos que está siendo objeto de estudio. En cada nodo del árbol se muestra la estructura y los valores que se verán afectados en la subllamada correspondiente, primero con sus valores contenidos al iniciarse la ejecución de la correspondiente subllamada recursiva, y posteriormente con los valores que ha adquirido tras la finalización de dicha subllamada y de las que dependen de ella. Esta característica resulta de gran interés para entender mejor la evolución del estado de la estructura y la forma en que poco a poco ha ido sufriendo modificaciones en su contenido.

La segunda de las vistas que se presentan, y que siempre se emplea de manera alternativa a la primera propuesta, es la vista cronológica, que permite ver cómo se centra el algoritmo en las diferentes partes de la estructura en cada paso del mismo.

Esta vista, mostrada en la figura 7 con los pasos numerados cronológicamente sobre un algoritmo de ordenación, se centra en el estado de la estructura y muestra paso a paso cómo va modificándose su contenido.

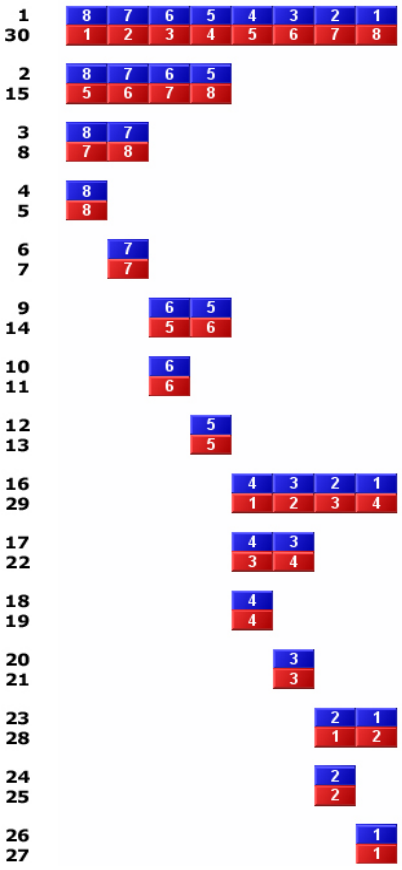

Figura 7. Transcurso de la animación (vista cronológica) para la técnica de "Divide y vencerás"

El paso cronológico queda establecido mediante la posición de los datos en el eje vertical. Así, la primera llamada al algoritmo se sitúa en la posición más alta de la vista, mientras que a medida que se van realizando nuevas subllamadas éstas se van situando en la parte inferior ordenadamente.

Esta vista resulta muy intuitiva al establecer de manera muy clara el orden en que se actúa sobre la estructura. Algunas de las opciones de configuración que presenta están destinadas a clarificar y contextualizar los valores que se van mostrando. Así, por ejemplo, permite que en cada paso se muestre sólo la parte de la estructura que está siendo objeto del algoritmo en esa subllamada recursiva o bien que se muestre la estructura completa, diferenciando con colores las partes tratadas y no tratadas en cada subllamada.

La componente cronológica es el principal activo de esta vista, ya que ayuda en gran medida al alumno a seguir el transcurso de la actuación del algoritmo sin distraerse en otros elementos del algoritmo.La tercera vista también se centra en el estado de la estructura, pero conserva un enfoque muy diferente. Esta vista, que está visible en todo momento, se limita a representar la estructura de forma detallada en su estado actual. Emplea líneas y colores para representar las partes de la estructura que ya han sido tratadas, las que están siendo tratadas en la parte actual y las partes que aún no se han modificado por el algoritmo.

El empleo de colores ayuda a controlar el estado de todas las partes, mientras que el empleo de líneas tiene como objetivo distinguir las distintas partes en que ha sido dividida la estructura.

Durante la visualización de arrays, las líneas dividen el array en varias partes. En el ejemplo de la figura 8, ya se ha ordenado la primera parte del vector, mientras que de la segunda mitad ya se ha tratado la primera posición. En ese instante el algoritmo trabaja en la segunda posición y, tras ordenar estas dos posiciones, se centrará en la siguiente división, la de las dos últimas posiciones.

Figura 8. Vista de estructura (arrays)

Con las matrices, las líneas divisorias se sitúan por encima de las posiciones de la matriz permitiendo ver de igual forma qué partes se han manejado de manera aislada y cuáles permanecen aún pendientes. En concreto, en la figura 9 se presenta un ejemplo donde los dos cuadrantes superiores ya han sido tratados por el algoritmo.

El tercer cuadrante (parte inferior izquierda) está siendo objeto del algoritmo. Dentro de él, ya se ha manejado la primera parte, mientras que actualmente la segunda es en la que se encuentra centrado el algoritmo en el instante capturado. Quedan por tanto otras dos partes más de ese cuadrante por manejar, así como el último de los cuadrantes de la matriz general.

Justo debajo de la matriz aparece de manera esquemática un dibujo de la misma para mostrar en qué partes se ha dividido la matriz, cuáles han sido ya tratadas y cuáles se encuentran aún pendientes, con el fin de complementar al resto de información suministrada.

Durante la visualización de esta técnica se mantiene en la ventana de la aplicación la vista estática que contiene el código Java de la clase, igual que ocurría durante la visualización de la recursividad.

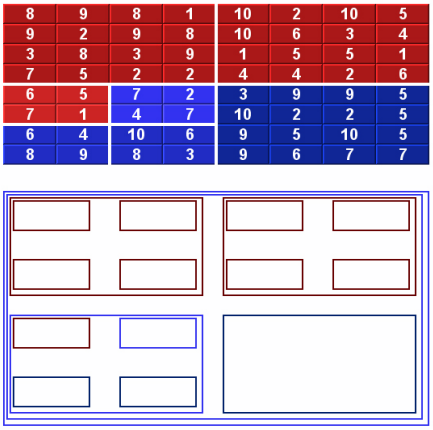

Figura 9. Vista de estructura (matrices)

6. Resultados de su uso

SRec ha sido empleado en clases magistrales desde el curso 2007/2008 para apoyar al profesor en sus exposiciones ante el alumnado. Éste a su vez también ha tenido oportunidad de utilizar el software para la realización de prácticas y repaso de ejemplos y ejercicios propuestos.

A lo largo de los cursos 2006/2007 a 2008/2009 se han realizado tres evaluaciones de usabilidad entre diferentes grupos de alumnos para realizar una medición tanto del grado de aceptación de SRec y de las animaciones que produce como de los cambios registrados en los resultados académicos del alumnado tras su comienzo de utilización.

Estas sesiones tuvieron una duración de dos horas. En ellas se plantearon varios ejercicios relacionados con la programación y análisis de algoritmos. El primero de ellos siempre fue realizado por el profesor con el fin de presentar la aplicación e introducir al alumnado a un manejo básico de la misma. Los siguientes ejercicios se enfocaban en familiarizar al alumno con la aplicación y con las posibilidades que le ofrecía. Posteriormente se les planteaba un ejercicio que les exigía cierta capacidad de análisis y diseño de algoritmos.

En la primera sesión se les planteó la depuración del algoritmo de ordenación *mergesort*, mientras que en la segunda se les planteó el análisis del algoritmo de *competición*, con recursividad múltiple. En la tercera sesión, que tuvo lugar con una versión más avanzada de SRec, similar a la que aquí se presenta, se les planteó un problema donde tenían que eliminar la redundancia debida a la recursividad múltiple empleando técnicas de memorización y tabulación, ofreciendo posteriormente un cálculo de la complejidad de los algoritmos resultantes.

Pregunta	Media
SRec es fácil de usar	3,88
SRec me ha ayudado a analizar algoritmos y encontrar el error	2,63
SRec me ha ayudado a analizar algoritmos para comprobar que la solución propuesta es la correcta	4,13
Calidad general de SRec para analizar la recursividad	3,38
Calidad del menú principal	3,75
Calidad de los controles de animación	4,38
Calidad de la vista de traza	3,75
Calidad de la vista de la pila de control	4,00
Calidad de la vista del árbol de recursión	4,25
Calidad de la configuración de las visualizaciones	3,88
Interacción con los paneles	3,63
SRec me ha gustado	3,63

Tabla 2. Valores del cuestionario de la sesión 1

Pregunta	Media
SRec es fácil de usar	4,50
SRec me ha ayudado a analizar algoritmos para analizar qué llamadas se realizan en tiempo de ejecución	4,29
SRec me ha ayudado a analizar algoritmos para identificar la dependencia de llamadas recursivas	4,36
Calidad general de SRec para analizar la recursividad	4,29
Calidad del menú principal	4,07
Iconos	3,86
Calidad de los controles de animación	4,50
Calidad de la vista de traza	4,00
Calidad de la vista de la pila de control	4,04
Calidad de la vista del árbol de recursión	4,43
Calidad de la configuración de las visualizaciones	3,82
Interacción con los paneles	3,89
SRec me ha gustado	4,26

Tabla 3. Valores del cuestionario de la sesión 2

Pregunta	Media
SRec es fácil de usar	4,2
SRec me ha ayudado a analizar algoritmos y determinar las llamadas recursivas que se realizan	4,19
SRec me ha ayudado a analizar algoritmos para identificar la dependencia de llamadas recursivas	4,05
Calidad general de SRec para analizar la recursividad	4,00
Calidad de iconos	3,57
Calidad de los controles de animación	3,71
Calidad de la vista del árbol de recursión	4,00
Calidad del visor de los árboles grandes	3,86
Configuración de formatos	3,76
Configuración de zoom	3,71
Interacción con los paneles	3,80
Proceso de generación de una animación	4,00
Proceso de almacenar / cargar una animación	4,14
Visualización almacenada en un fichero de captura	4,10
SRec me ha gustado	3,95

Tabla 4. Valores del cuestionario de la sesión 3

Las sesiones de usabilidad se realizaron separadas en el tiempo y sobre versiones de SRec cada vez más completas y complejas. Esto requiere de un mayor esfuerzo por parte del usuario a la hora de aprender el manejo de la herramienta, si bien destaca el dato de que la

pregunta con mayor puntuación de los cuestionarios 2 y 3 es la que refleja si SRec resulta fácil de usar.

Respecto a los resultados académicos, en la sesión número 2 se apreció que 26 de los 28 alumnos que llevaron a cabo la práctica la entregaron correctamente, y de ellos 23 emplearon SRec para apoyarse en la realización de la misma.

Tras la sesión 3 los alumnos entregaron una práctica similar, y destaca el hecho de que se dobló el número de aprobados con nota *Sobresaliente* respecto a prácticas anteriores de la asignatura junto con que espontáneamente los alumnos indicaron en un alto porcentaje en el apartado de conclusiones del informe de prácticas que SRec les había ayudado en gran medida a entender mejor el funcionamiento de los algoritmos propuestos así como a depurar los algoritmos que ellos mismos debían implementar.

En general, el alumnado ha sido muy receptivo a SRec, encontrando en la aplicación una herramienta de apoyo que le ha ayudado en la consecución de buenos resultados académicos gracias a las animaciones, que permiten realizar un seguimiento muy visual de la ejecución de los algoritmos paso a paso.

7. Conclusiones y trabajos futuros

Se ha presentado una aplicación informática, SRec, que genera de forma automatizada la animación de la ejecución de programas codificados en lenguaje Java. Como ya se ha visto, SRec ofrece numerosas facilidades educativas orientadas a facilitar la labor docente de los profesores y el aprendizaje de los alumnos. Algunas de ellas fueron incorporadas después de las sesiones de evaluación de la herramienta, ya que el objetivo es que la aplicación sea totalmente afín a las necesidades tanto del profesorado como de los alumnos.

Sin embargo, en asignaturas de programación y algoritmia, contar con una herramienta que es capaz de mostrar de manera visual la ejecución de programas recursivos y de la técnica de "Divide y vencerás" puede no ser suficiente para abordar toda la temática de las mismas.

Por ello actualmente se encuentran en desarrollo algunas extensiones de SRec de cara a un futuro próximo. Una de estas extensiones pasa por incorporar animaciones más específicas para otras técnicas de diseño no contempladas aún en el trabajo realizado como "Programación dinámica" y "*Backtracking*", ambas basadas en la recursividad, igual que la técnica de "Divide y vencerás". Para estas técnicas aún está en fase de estudio qué tipo de visualizaciones se implementarán.

Otro de los trabajos que quedan planificados para el futuro es permitir que SRec pueda exportar sus animaciones en un lenguaje estándar de animación de algoritmos como SVG. Hasta ahora, SRec emplea un formato propio, basado igualmente en XML, pero resulta de gran interés hacer uso de SRec para generar animaciones sin esfuerzo y poder emplearlas en otras aplicaciones.

Agradecimientos

Este trabajo ha sido financiado por los proyectos TIN2004-07568 y TIN2008-04103 del Ministerio de Educación y Ciencia.

Referencias

[1] Ciesielski, V., MacDonald, P. *Using animation of state space algorithms to overcome student learning difficulties.* Proc. 6th Annual Conference on Innovation and Technology in Computer Science Education. ACM Press, 97-100, 2001.

[2] MacDonald, P., Ciesielski, V. *Design and evaluation of an algorithm animation of state space search methods.* Computer Science Education 12, 4. 301-324, 2002.

[3] Naps, T.L., Roessling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger S., (y autor). *Exploring the role of visualization and engagement in computer science education*. SIGCSE Bulletin, 35(2): 13-152, 2003.