Aprendizaje Lúdico en Laboratorio de Programación

David Bueno, Julio Garralón, José M. Jerez, Antonio Maña
Dept. de Lenguajes y Ciencias de la Computación
Universidad de Málaga
29071 Málaga
e-mail: {bueno,jgr,jja,amg}@lcc.uma.es

Resumen

Es obvio que la motivación del alumno es un factor decisivo en su aprendizaje, como lo son el interés y el gusto por la asignatura que estudia. En especial, en el caso de asignaturas que el alumno no identifica directamente con la titulación que cursa, la motivación es pobre y ello se refleja en los resultados. En este trabajo se presenta un enfoque pedagógico desarrollado por el equipo docente de la asignatura Laboratorio de Programación de la Ingeniería Técnica de Telecomunicación de la Universidad de Málaga. El enfoque, basado en el uso de juegos de ordenador, se ha demostrado muy adecuado.

1. Introducción y motivación

Los factores que influyen en el proceso de aprendizaje son múltiples [] y su interacción es compleja. Uno de los principales es la *motivación* del alumno hacia la materia a aprender[1] [Santos]. Desde este punto de vista advertimos en nuestra asignatura una falta de motivación en los alumnos que en la asignatura homóloga de la titulación en Informática no se producía. Un alumno motivado dedica más tiempo y esfuerzo al estudio de la materia y aprende a disfrutar de los logros que va consiguiendo. Un aspecto también importante en el caso de la enseñanza universitaria es la falta de referencias sobre el propio proceso de aprendizaje por parte del alumno. Hasta que se somete al examen no cuenta con medios para determinar si su evolución es satisfactoria o no.

La programación de ordenadores es frecuentemente calificada de "arte" debido al componente creativo de la misma, y es en este aspecto creativo y lúdico donde creemos que puede encontrarse el elemento motivador que necesitamos. Por todo ello, cuando nos planteamos el objetivo de mejorar nuestro rendimiento como docentes y el de nuestros alumnos como programadores, lo hicimos basándonos en dos elementos principales: por una parte, la motivación del alumno y por otra la definición de pequeños objetivos parciales que permitiesen la autoevaluación y obtención de una "recompensa" por parte de los propios alumnos. Este segundo elemento produce una realimentación sobre el primero, ya que la consecución de cada objetivo tiene un efecto positivo sobre la satisfacción y la motivación del alumno respecto a la asignatura.

La asignatura de Laboratorio de Programación tiene un enfoque eminentemente práctico, de hecho la realización de las prácticas consume aproximadamente un 85% del tiempo dedicado a la asignatura, por lo que consideramos razonable actuar sobre este aspecto de la docencia. Las prácticas tradicionales basadas en ejercicios relacionados con las matemáticas (por ejemplo, hallar la suma, la media, el máximo y el mínimo de una serie de datos contenidos en un array), no resultaban motivadoras para el alumno, el cual tenía la impresión de que no iba a aprender nada que le fuese útil. Específicamente, en relación a la materia, se identificó como punto de mejora el corregir el enfoque tradicional de basar la enseñanza simplemente en el aspecto de síntesis de programas descuidando el de análisis.

Como punto de partida del plan de mejora se realizó un análisis de la materia a impartir [USDE] y se organizaron los diferentes objetivos parciales a cubrir por cada una de las diferentes prácticas para gestionar más eficientemente el siempre escaso tiempo de la asignatura. De este estudio y de la decisión de buscar la motivación del alumno como primer objetivo partió la idea que, una vez desarrollada, presentamos aquí.

2. Trabajos previos

[1] Las asignaturas menos atrayentes tanto para los alumnos de Ingeniería Informática como para los de Ingeniería de Telecomunicación son las de Física y Matemáticas (Cálculo y Álgebra). En el siguiente lugar se sitúan Elementos y Laboratorio de Programación (Santos define a la asignatura Programación como "Pesadilla III").

Pocos son los trabajos que se han realizado acerca de metodologías y herramientas pedagógicas en cursos de introducción a la programación centrados en su aspecto lúdico. Algunos autores han basado parcialmente sus propuestas en elementos motivadores [Peri].

En general, la investigación desarrollada en la línea de mejorar el rendimiento o la eficacia de la docencia suele basarse en alguno de estos aspectos:

- *Organización y planificación.* Se trata de preparar el trabajo docente y presentarlo a los alumnos del modo más conveniente para su asimilación. Se incluyen aquí formas de estructurar los contenidos, de diseñar y realizar actividades, etc. Por analogía con el campo que nos ocupa de la programación diríamos que actúa a nivel de la arquitectura de la aplicación [BDA].
- *Medios de expresión y comunicación.* Se intenta dotar a la materia de un medio atractivo de presentación. Siguiendo con la analogía anterior estaríamos actuando en el nivel estético del interfaz de usuario [Santos] [DGST].
- *Fomento de la interacción profesor-alumno*. En las propuestas realizadas en este sentido se pretende fomentar la comunicación bidireccional entre el profesor y el alumno de forma que el primero obtenga datos importantes para el seguimiento de la docencia y el segundo una atención más personalizada a sus necesidades. El problema de este enfoque suele ser la imposibilidad de llevarlo a cabo en las condiciones de clase habituales. Estaríamos centrándonos en el aspecto dinámico del interfaz [FPC] [ShHa].
- *Enfoques dirigidos a algún contenido de la materia.* Son enfoques totalmente adaptados a unos contenidos concretos en una situación determinada. Por su especificidad son muy interesantes pero su utilidad es limitada. Serían el equivalente a un algoritmo eficiente de solución de un problema concreto [ShHa].
- *Motivación al alumno*. Se pretende conseguir un mayor esfuerzo por parte del alumno basado en la idea de sustituir obligación por afición. Estaríamos actuando en el aspecto ergonómico de la aplicación o en la cualidad que en el mundo de los juegos de ordenador suele referenciarse como jugabilidad [Peri].

3. Metodología

El diseño de las prácticas de laboratorio debe tener en cuenta los siguientes aspectos:

- La *limitación de tiempo* que los planes de estudio actuales imponen a las asignaturas es una restricción muy severa para que un alumno novel logre un programa ejecutable libre de errores en horas de laboratorio. Concretamente, la asignatura objeto de análisis en este trabajo tiene asignados 4,5 créditos en un solo cuatrimestre.
- La *coordinación* que requiere la parte teórica con la práctica debe tenerse presente. No se debe exigir al alumno en el laboratorio conceptos no explicados en teoría, ni se debe dejar pasar mucho tiempo entre lo enseñado en teoría y su puesta en práctica.
- La *evolución de la complejidad* de las prácticas debe ir aumentando gradualmente, evitando grandes saltos teóricos entre una práctica y la siguiente.

Examinando los temarios de las asignaturas de programación de primer curso de las escuelas de ingeniería técnica de las universidades españolas, podemos apreciar que todas ellas tienen una estructuración similar (Anexo I), basada en la programación imperativa de alto nivel. La mayoría de estas asignaturas tienen una componente teórica y una práctica, ésta última desarrollada fundamentalmente en laboratorios. Los lenguajes más utilizados son Pascal, C y, en menor medida, Modula-2 y Ada. Últimamente se está empezando a utililizar lenguajes orientados al objeto como C++ y Java en los primeros cursos.

En el caso particular de la Escuela Técnica de Telecomunicación de la Universidad de Málaga, durante el primer cuatrimestre del curso se imparte la asignatura teórica de Introducción a los Computadores, mientras que en el segundo se imparten las asignaturas de Elementos de Programación y Laboratorio de Programación, en la que se utiliza Modula-2 para la resolución de los ejercicios, dado el alto nivel pedagógico que proporciona este lenguaje de programación. No obstante, a partir del próximo curso se utilizará C++ debido a su implantación en el mercado laboral. Cabe esperar que este cambio influya positivamente en la motivación del alumno.

La metodología planteada en este trabajo se basa en el interés que supone para el alumno la realización de prácticas de laboratorio basadas en juegos en los cuales la complejidad de los mismos va incrementándose en función de los conocimientos adquiridos en las clases teóricas. De esta forma, la asignatura de laboratorio de programación se ha dividido en tres ejercicios prácticos con diferentes apartados enfocados a

trabajar detalladamente los conceptos más importantes de cada uno de los temas centrales de la asignatura:

1. Elementos básicos necesarios para la construcción de un programa: estructura general, variables, tipos, estructuras de datos simples, estructuras de control, modularización de programas y entrada-salida elemental.
2. Estructuras de datos complejas y ficheros.
3. Tipos abstractos de datos y diseño e implementación de módulos de librerías.

La primera práctica intenta asentar los conocimientos sobre estructuras de datos simples del lenguaje adquiridos en el primer cuatrimestre del curso, donde el alumno no tiene contacto con el ordenador, lo que dificulta la puesta en práctica de los conceptos adquiridos sobre programación. En ella, se propone la realización de un juego simple consistente en acertar un número generado aleatoriamente por el ordenador. Para ello, el alumno deberá hacer uso de la entrada-salida estándar, los tipos de datos simples y funciones básicas de las librerías que proporciona el compilador. Posteriormente, el concepto de subprograma permitirá al alumno modularizar sus aplicaciones. Juegos dirigidos por menús, conversacionales, siempre con una entrada-salida en modo texto sencilla, serían adecuados para este nivel. En este sentido, los siguientes apartados de este primer ejercicio práctico requerirán que el programa proporcione pistas con el objetivo de acotar el espacio de búsqueda del número a acertar, y que incluso se proporcione la posibilidad de jugar contra el ordenador. Poco a poco el alumno irá tomando consciencia de algunos de los problemas relacionados con la ingeniería del software: sufrirá los efectos laterales de utilizar variables globales, de la falta de modularización con sentido, de la elección inconsistente de identificadores, etc.

La segunda práctica tiene como objetivo el manejo de estructuras de datos complejas, interfaz de entrada-salida en modo texto e introducción al manejo de ficheros. Para ello, se propuso la implementación del *juego de la serpiente*, en el cual los alumnos comienzan aprendiendo a controlar las teclas especiales de movimiento y las coordenadas de la pantalla mediante simples desplazamientos de puntos a través de los cursores. En posteriores apartados se definen una serie de estructuras de datos para, por ejemplo, almacenar el "cuerpo" de la serpiente, la configuración de la pantalla y del juego, permitir la inclusión de obstáculos, etc. En general, se propone incluir una serie de mejoras en el diseño final que hagan el resultado lo más atractivo posible, con el objetivo constante de mantener siempre un alto grado de motivación en la realización de las prácticas. Al término de este ejercicio, se introduce al alumno en el manejo de ficheros a través del mantenimiento de un histórico de puntuaciones de jugadores.

Un ejemplo de la realización de la práctica y de la motivación del alumnado se muestra en la figura 1, donde se puede apreciar la complejidad final del ejercicio.

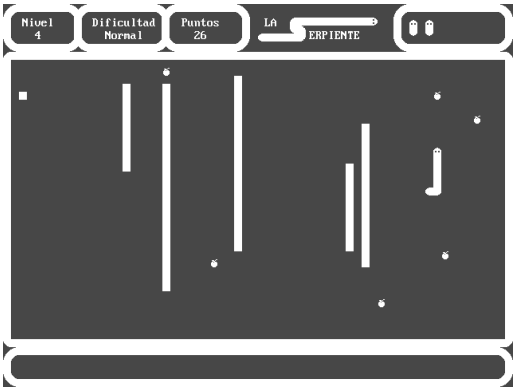

Fig. 1. Práctica de la Serpiente

La tercera y última práctica tiene como objetivo la definición, implementación y utilización de módulos de librerías para la construcción de tipos abstractos de datos, así como el uso de interfaces de usuario en modo gráfico. Para ello, se propuso la realización de un ejercicio práctico basado en el juego de *los barquitos*, donde el alumno debe manejar simultáneamente varias estructuras de arrays de registros. La complejidad de la aplicación debido a los niveles de modularización es tal que se impone la necesidad de definición de bibliotecas para manejo de matrices, ventanas, etc. En esta práctica se hace un uso más extensivo de los ficheros para el almacenamiento de datos correspondientes a jugadores y a estadios intermedios de las partidas. Además, se intenta dotar de "inteligencia" al programa para que juegue contra el alumno. Este apartado permite poner en práctica los conocimientos de recursividad a través de estrategias de búsqueda mediante las cuales se exploran las posibles soluciones para ganar la partida. Por otro lado, se propone también la posibilidad de jugar entre diferentes alumnos a través de un protocolo de transmisión de jugadas contenidas en ficheros vía red local. La utilización de un entorno gráfico

avanzado, unido a la motivación del alumno para mejorar interactividad del juego, dan lugar a aplicaciones bastante atractivas como la que se muestra en la figura 2.

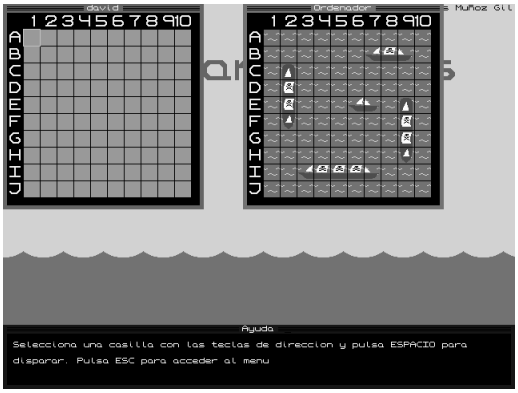

Fig. 2. Práctica de los Barquitos

4. Resultados

Aunque nuestras intenciones de mejorar la docencia de la asignatura en varios aspectos sea buena, es necesario comprobar que nuestras hipótesis son ciertas. Para ello realizamos un cuestionario en el que se eligen como muestra a 62 alumnos. Dichos alumnos eran los asistentes al examen de septiembre, hecho que podría influir negativamente en los resultados pues no son estos alumnos los más satisfechos con la asignatura. Entre ellos el 73% eran repetidores, dato muy útil para poder comparar la metodología pedagógica basada en juegos con la aplicada en años anteriores.

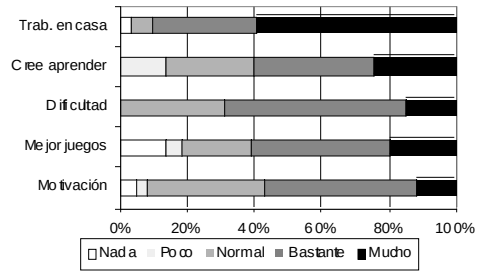

Fig. 3. Resultado del cuestionario

Las preguntas se plantearon dando 5 respuestas posibles (nada, poco, normal, bastante y mucho). En la figura 3 se muestra en cada fila los porcentajes de cada posible respuesta para cada pregunta. Analizando las preguntas podemos ver que la motivación del alumno es alta (mucho o bastante) en un 58% y solamente poco o nada motivado en un 8%. Este hecho es bastante significativo en una carrera en la que la informática no es el objetivo fundamental. Por otro lado, comparando con años anteriores el 61% prefiere aprender con juegos a como se hacía antes. Sólo un 18% prefería los métodos tradicionales. Los alumnos consideran las prácticas con una dificultad alta en un 69%, lo que parece consecuente con el hecho de que el 90% de los alumnos utilizan el ordenador en casa mucho o bastante para la realización de las prácticas.

Además de las preguntas cerradas en el cuestionario se dejó un apartado para un comentario abierto. Los más repetidos y relevantes son:

- La asignatura ayuda a afianzar los conocimientos de las asignaturas teóricas
- Se pide utilizar un lenguaje que pueda servirles en el mercado laboral
- Se necesita mucha dedicación
- Con los juegos la asignatura se hace más amena

5. Conclusiones

Como puede verse, aunque el contexto en el que se realizaron los cuestionarios podría parecer negativo, los resultados han sido realmente satisfactorios, consiguiendo una respuesta del alumnado bastante positiva en cuanto a motivación y resultados.

En este sentido, creemos conveniente destacar la mayor asistencia a clase respecto a cursos anteriores después de la finalización del primer ejercicio práctico. El aumento de la motivación en el alumnado también queda reflejada en la mayor utilización del ordenador en casa para la resolución de los ejercicios, los más avanzados haciendo uso, incluso, de recursos de programación no explicados a ese nivel del curso.

Anexo I

Patrón básico del temario de una asignatura de laboratorio de programación en titulaciones en ingeniería técnica de universidades españolas:

Tema 1. Entorno de trabajo: sistema operativo y entorno de programación.
Tema 2. Elementos básicos del lenguaje: tipos de datos y declaración de variables, expresiones

sencillas, entrada/salida.
Tema 3. Estructuras de control: decisiones y bucles.
Tema 4. Procedimientos y funciones. Uso de bibliotecas.
Tema 5. Estructuras de datos estáticos: tablas, registros, conjuntos.
Tema 6. Estructuras de datos permanentes: ficheros.
Tema 7. Estructuras de datos dinámicas: punteros.
Tema 8. Tipos de datos abstractos.

Referencias

[BDA] Berglund, A.; Daniels M.; Almstrum, V.; *A Smorgasbord of Pedagogical Dishes*. Proceedings of the Second Australasian Computer Science Education Conference. Melbourne, Australia, 1997. Accesible en http://www.docs.uu.se/docs/cse/papers/dishes.html

[DGST] Daniels, M.; Gal-Ezer, J.; Sanders, I.; Teague, J: *Teaching computer science: Experiences from four continents*. Proceedings of the ACM SIGCSE Computer Science Education Symposium. Philadelphia. 1996.

[FPC] Fincher, S.; Petre, M.; Clark, M., (Eds.). *Computer Science Project Work - Principles and Pragmatics*. Springer-Verlag. ISBN: 1-85233-357-X. 2001.

[Peri] Peri, J.; Godoy, D.: Utilización de acertijos lógicos como ejercicios motivadores para la enseñanza de la programación lógica. CACIC98. 1998. URL: http://gidis.ing.unlpam.edu.ar/downloads/godoyd/CACIC98.html

[Santos] Santos, L. C. *Prototipo Didáctico para la Enseñanza de la Programación en Computación*. Comunicación personal.1998

[ShHa] Sheard, J.; Hagan, D.L. *A special learning environment for repeat students*. Proceedings of 4th Annual Conference on Integrating Technology into Computer Science Education ITiCSE'99 (Krakow, Poland. June 1999).

[USDE] U.S. Department of Education. Office of Educational Research and Improvement. *An Educator's Guide To Evaluating The Use Of Technology In Schools And Classrooms*. 1998