Desarrollo de juegos: una metodología eficaz para aprender un lenguaje de programación

Autor/res/ras: Pedro J. García Laencina[(1)] y Pedro María Alcover Garau[(2)]

Institución u Organismo al que pertenecen:
(1) Centro Universitario de la Defensa (CUD) de San Javier, MDE-UPCT.
(2) Universidad Politécnica de Cartagena (UPCT).

Indique uno o varios de los seis temas de Interés: (Marque con una {x})

{ } Enseñanza bilingüe e internacionalización

{ } Movilidad, equipos colaborativos y sistemas de coordinación

{ } Experiencias de innovación apoyadas en el uso de TIC. Nuevos escenarios tecnológicos para la enseñanza y el aprendizaje.

{x} Nuevos modelos de enseñanza y metodologías innovadoras. Experiencias de aprendizaje flexible. Acción tutorial.

{ } Organización escolar. Atención a la diversidad.

{ } Políticas educativas y reformas en enseñanza superior. Sistemas de evaluación. Calidad y docencia.

Idioma en el que se va a realizar la defensa: (Marque con una {x})

{x} Español **{ }** Inglés

Resumen.

Aprender un lenguaje de programación es una difícil tarea para el alumno ya que es necesario un estudio continuado y muchas horas de trabajo. Así mismo, desde el punto de vista del profesor, constituye un desafío exigente y creativo ya que requiere el diseño de actividades y materiales que orienten y motiven al alumno durante su proceso de aprendizaje. Una manera eficaz de conseguirlo es mediante la programación de juegos. Este artículo presenta y analiza distintas experiencias basadas en el diseño y desarrollo de juegos de ordenador como trabajo final de asignaturas de introducción a la programación informática.

Palabras Claves: Lenguajes de Programación, Aprender a Programar, Juegos, Trabajo Final, Motivación.

Abstract.

Learning a programming language is a hard task for students because a continued study and a lot of working hours are required. Also, from the point of view of the

teacher, it is a demanding and creative challenge as it requires the design of activities and materials to guide and motivate students during their learning process. An effective way to do this is by developing games. This paper presents and analyzes different experiences based on the design and development of games as the final assignment task of first courses in software programming.

Keywords: Programming languages, Learning Programming, Games, Final Assignment Task, Motivation.

1. Introducción

Las Tecnologías de la Información y las Comunicaciones (TICs), han introducido cambios revolucionarios en nuestro trabajo, en nuestros modos de relacionarnos con los demás y, en general, en casi todos los aspectos y ámbitos de nuestra vida. Es cada vez más necesario, en la mayoría de ámbitos profesionales, conocer las tecnologías, siempre novedosas y ya habituales. Ya sea en un pequeño comercio familiar o en una compleja planta petroquímica, no hay empresa o negocio que no haga uso de programas de propósito general para la gestión, la comunicación, etc.

Por estas razones, la formación académica en el ámbito de las ingenierías incluye la adquisición de conocimientos y competencias en el desarrollo software y en los lenguajes de programación. También se ve conveniente incluir esta formación en la enseñanza secundaria. Sin embargo, como habrá podido experimentar cualquiera que haya abordado por primera vez el reto de aprender un lenguaje de programación, este proceso de aprendizaje es complicado y tedioso, y requiere bastante estudio. Especialmente exige muchas horas de trabajo delante del ordenador. Y es que, aunque pueda parecer una obviedad, en realidad no hay otro camino: «*para aprender a programar, hay que programar*». Ni mucho menos es suficiente con copiar y entender el código que el profesor explica en clase o que se muestra en un manual o en una página web; el alumno debe llegar a ser capaz de crear su propio código.

Los profesores también comprobamos que la enseñanza de uno de estos lenguajes es una labor complicada, donde hay que acudir con frecuencia a recursos de motivación. Para un alumno que nunca haya programado, no resulta trivial comprender un paradigma de programación (orientada a eventos, o a objetos; o programación estructurada), ni los conceptos asociados a ese lenguaje y ese paradigma, como las estructuras de control y de datos, las exigencias de la sintaxis, etc. Es muy útil que el profesor guíe el proceso de enseñanza-aprendizaje, planteando y resolviendo problemas ya conocidos y resueltos por el alumno en otros ámbitos de conocimiento. La resolución de estos ejercicios prácticos permite introducir los fundamentos de la metodología de programación. De esta forma, es posible despertar el interés del alumno e incentivar su motivación. La clave del éxito está en utilizar las estrategias docentes que consigan cambiar el rol del alumno: que deje *de ser usuario y se convierta en desarrollador de su propio software*; implicándose así en el proceso creativo del código de su programa, utilizando para ello los conocimientos adquiridos en la asignatura.

Este artículo presenta una metodología docente basada en la programación de juegos, muy extendidos, y posiblemente conocidos y usados por muchos de los alumnos. A lo largo de los últimos años, los autores han implantado con éxito esta metodología como trabajo final de asignaturas de introducción a la programación de primer curso de titulaciones de grado universitario del entorno de la ingeniería. En las siguientes secciones, además de introducir los principales fundamentos de la metodología docente basada en el desarrollo de juegos, se describen las experiencias desarrolladas bajo esta metodología.

2. Implantación de metodología docente basada en el desarrollo de juegos

Tradicionalmente, ha sido habitual introducir los fundamentos de un lenguaje de programación mediante el planteamiento y la resolución de sencillos problemas matemáticos; es éste un recurso al que seguimos acudiendo. Así, por ejemplo, un ejercicio útil para explicar las estructuras de control iteradas (bucles) podría ser el siguiente: *«Desarrollar un programa que reciba un número entero y muestre todos los enteros que lo dividen. Tenga en cuenta que todos los divisores de un número entero, excepto él mismo, se encuentran entre 1 y la mitad del entero en el que se buscan los divisores.».* Desde un punto de vista de la enseñanza en programación, el uso de este tipo de problemas matemáticos ofrece algunas ventajas (Fritelli *et al.* 2013), ya que (1) los procedimientos de resolución —definidos por algoritmos— son ya conocidos por los alumnos; (2) son ejercicios cortos y sencillos, que permiten identificar claramente los datos de entrada y de salida, así como los algoritmos a programar; y (3) los alumnos son capaces de verificar con simples cálculos que la solución ofrecida por su programa es correcta.

Este tipo de ejercicios matemáticos son adecuados inicialmente para introducir los conceptos, pero tienen los inconvenientes de que no resultan un desafío atrayente para los alumnos y pueden ser —por si solos— insuficientes para que adquieran adecuadamente todas las competencias definidas en la asignatura. Por ello, y una vez que se han impartido los principales conceptos de un lenguaje de programación, conviene que el profesor diseñe e incorpore casos prácticos con un mayor potencial de desarrollo —siempre dentro del contexto y grado de complejidad de la asignatura— y que resulten atractivos al alumnado. En los últimos años, distintos trabajos (Fritelli *et al*. 2013; Kert & Erkoç, 2011) han demostrado la eficacia de *desarrollar juegos de ordenador para aprender programación*. En ese sentido, tal y como indican Fritelli *et al.* (2013), la programación de juegos aporta muchas ventajas para el aprendizaje. Para la mayoría de los alumnos es un reto muy atractivo poder enfrentarse a la implementación de un juego en un programa de ordenador. Al mismo tiempo que el alumno desarrolla y amplía las funcionalidades del código de su propio juego, también desarrolla las capacidades y competencias de la asignatura. El alumno aprende jugando (Bueno *et al.* 2001).

Ahora bien, ¿cómo implantar esta metodología docente? Según la experiencia de los autores resulta oportuno proponer la programación de un juego como Trabajo Final de Asignatura (TFA). Estos trabajos se realizan en las últimas semanas del curso donde ya se ha impartido la mayor parte de los conceptos de la asignatura, necesarios para el diseño y el desarrollo software del juego. En las siguientes

secciones se describen distintos aspectos relacionados con el proceso de implantación de esta metodología.

2.1. Selección del juego y definición de la propuesta de trabajo final

Lógicamente, el primer paso corresponde al profesorado. Éste debe seleccionar el juego en función de los contenidos y del nivel en la adquisición de competencias que persiga la asignatura. No es lo mismo una asignatura básica de introducción al lenguaje C, para la cual se escogen juegos sencillos (como el Ahorcado, Buscaminas, Sudoku, etc), que una asignatura avanzada de laboratorio de programación en C++, donde se pueden seleccionar juegos más complejos.

Una vez se ha seleccionado el juego, el profesor debe implementarlo (conviene buscar varias posibles soluciones tipo) utilizando los conceptos y recursos explicados en la asignatura. Es necesario, a partir de este trabajo inicial, redactar una memoria suficientemente clara, detallada y auto-contenida, para que el alumno sea capaz de programar el juego por sí mismo.

El documento debe explicar el funcionamiento del juego, y concretar con claridad las especificaciones de lo que la aplicación implementada por el alumno debe lograr hacer. La memoria debe guiar el trabajo del alumno. Según sea la complejidad del juego, conviene proporcionar una descripción de las distintas funciones que se podrían implementar e, incluso, ofrecer el código de alguna o algunas de ellas. De esta forma, el alumno tiene inicialmente acotado lo que debe programar para alcanzar los objetivos mínimos y superar el TFA; el código propuesto también sirve como herramienta de aprendizaje para el alumno, que puede conocer soluciones software de cierta calidad. Desde luego, el alumno siempre podrá ofrecer otras soluciones distintas a las sugeridas en el documento. Siempre habrá alumnos con iniciativa que prefieran desarrollar íntegramente su propio código, y que podrán llegar a soluciones originales que deberán ser calificadas de manera especial, como veremos más adelante.

Es importante que, el tiempo necesario para resolver el TFA esté bien tasado, para que no suponga una inversión de horas muy superior al que corresponda por el número de créditos ECTS de la asignatura.

2.2. Presentación a los alumnos de la propuesta de juego como trabajo final

El alumno debe disponer del documento descriptivo del TFA con antelación suficiente para que pueda leer y comprender toda la información y el código disponible, antes de comenzar con su trabajo de programación. En el caso de la experiencia aquí presentada, el documento se presenta seis semanas antes de la finalización de las clases de la asignatura.

Una semana después de publicar el documento, cuando ya el alumno ha podido tener un primer contacto con la memoria y ya ha podido conocer el alcance del TFA, se realiza una presentación presencial del enunciado, donde se atienden las primeras dudas que el documento haya podido suscitar. En esta presentación se

describen las bases y el funcionamiento del juego, los objetivos que debe cumplir el código desarrollado, la planificación temporal, el procedimiento de entrega del TFA y los criterios de evaluación/calificación.

2.3. Fase de programación del juego y entrega del código desarrollado

Antes de comenzar a programar, el alumno debe entender perfectamente la memoria y resolver todas sus dudas con sus profesores, solicitando las tutorías que sean necesarias. Es conveniente estar pendiente del trabajo de los alumnos en las primeras semanas; aquellos que van retrasando el reto de enfrentarse a la resolución del TFA terminan por percibirlo como un trabajo inabordable y fácilmente se rinden antes de comenzarlo.

Cuando el alumno logra hacerse cargo de qué aplicación debe implementar y cómo ha de hacerlo, y ya ha comprendido el código que se le proporciona, en muchos casos el TFA se convierte en un problema que reta a su inteligencia. Retado por el juego, el alumno aprende a programar jugando. En cuanto un alumno aplicado comienza a desarrollar su propio código, casi de manera inmediata se implica en ese trabajo y pone verdadero empeño en ver correctamente terminado el juego. En bastantes casos los alumnos dedican al TFA muchas más horas de las exigidas en la Guía Académica, porque buscan nuevas prestaciones que mejoren sus primeras versiones: jaleados por un reto casi personal, aprender programación casi se convierte en un *hobby*. Entre algunos alumnos se establece una 'competición interna' por intentar desarrollar el mejor juego. Indudablemente, estas situaciones redundan en el beneficio de su aprendizaje del lenguaje de programación.

Una vez el alumno ha terminado la fase de programación de su TFA dentro de los plazos establecidos, éste debe entregar el código desarrollado utilizando la plataforma de docencia on-line (en nuestro caso, Moodle).

2.4. Defensa y evaluación del trabajo final

Para poder evaluar el TFA de cada alumno, se realizan entrevistas individuales de corta duración: diez minutos aproximadamente. A lo largo de la entrevista con su profesor, el alumno debe mostrar el correcto funcionamiento de su juego, los conocimientos de programación aplicados, las ampliaciones desarrolladas, etc.

De manera general, hay dos posibles categorías discretas para la *calificación*:

- *No Apto*. Tendrán directamente esta calificación aquellos alumnos que no hayan entregado su código antes del día del examen y/o no se presenten a realizar la entrevista personal para la defensa de su código. Lógicamente, si el profesor encuentra a lo largo de la entrevista evidencias suficientes de que el TFA no es completamente original, éste se evaluará como No Apto.

- *Apto*. Tendrán esta calificación aquellos alumnos que hayan entregado su código en tiempo y forma y, además, se presenten a realizar la entrevista personal superándola satisfactoriamente.

Con respecto a la *calificación cuantitativa*, y siempre en el caso de conseguir un Apto en el TFA, éste se puede llegar a calificar desde 0,0 hasta 2,0 puntos adicionales en la nota final de la asignatura. Para realizar esta calificación, hemos desarrollado una *rúbrica*, que permite estandarizar la evaluación del TFA mediante unos criterios específicos, comunes para todo el alumnado. La rúbrica hace la calificación más simple, precisa y transparente. Normalmente, distinguimos dos partes principales en la rúbrica: 1) una primera sección correspondiente a la implementación básica del juego; y 2) una segunda sección destinada a aquellos alumnos que han desarrollado soluciones originales y ampliaciones al TFA. A modo de ejemplo, en el Anexo I, se incluye la rúbrica desarrollada para la calificación de uno de los juegos que se han propuesto.

3. Experiencias con el desarrollo de juegos como trabajo final

A continuación, se describen brevemente algunos de los juegos más destacados que se han propuesto como TFA en asignaturas de introducción a la programación. Para más detalles, toda la documentación relativa a estos TFA se encuentra disponible en: http://hdl.handle.net/10317/3846.

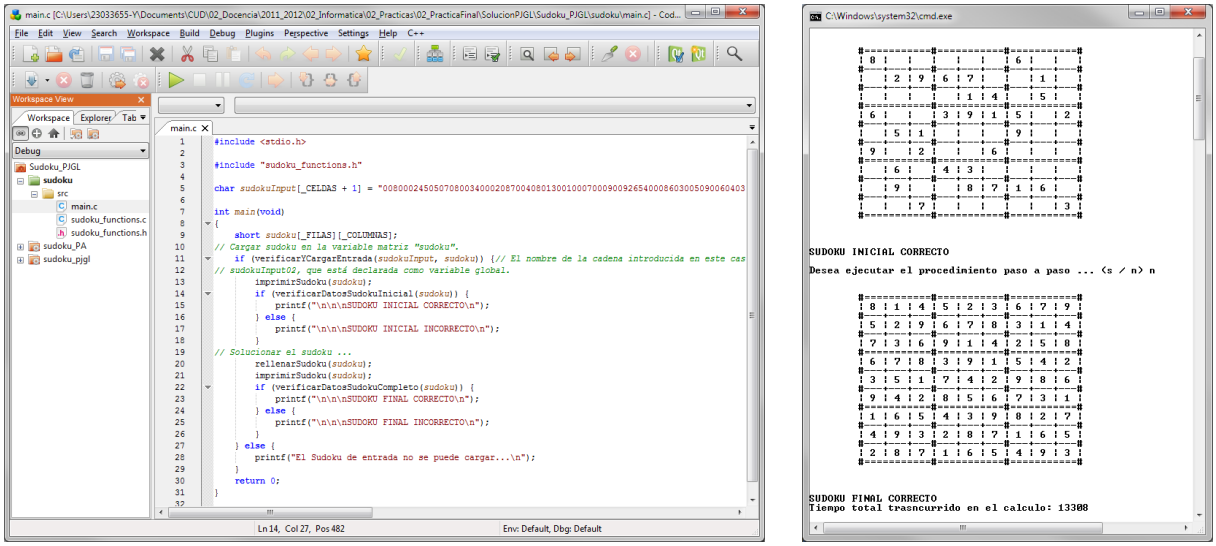
En todos los TFA propuestos, el alumno desarrolla las competencias definidas en asignaturas introductorias a la programación informática: conocer los fundamentos básicos de los ordenadores y los sistemas operativos, emplear herramientas para el desarrollo de software, construir algoritmos atendiendo a los paradigmas de programación, conocer los mecanismos básicos para representar y manipular diferentes tipos de datos, utilizar estructuras de control condicionadas e iterativas, modularizar programas empleando funciones, etc.

3.1. Sudoku

Los Sudokus son populares rompecabezas numéricos que podemos encontrar en las secciones de pasatiempos de cualquier periódico. Muchos de los alumnos han intentado alguna vez completar un Sudoku. La programación de este juego constituye un reto conocido y cercano para el alumno.

La configuración más habitual del Sudoku es una matriz cuadrada de 9 × 9 celdas, y subdividida en sub-matrices de 3 × 3, que se llaman cajas. Así pues, la matriz de un Sudoku tiene 9 filas, 9 columnas y 9 cajas. Y se presenta de forma que algunas de sus celdas tienen asignado un entero entre 1 y 9. El reto consiste en completar el cuadro asignando a cada celda vacía un valor entre 1 y 9 de forma que no haya dos números iguales dentro de una fila, o dentro de una columna o dentro de una caja.

Atendiendo a la memoria de este TFA, el código final presentado por el alumno debe ser capaz de (1) cargar los valores numéricos iniciales de Sudokus; (2) verificar si cada nuevo valor introducido por el usuario es válido, atendiendo al resto de valores del Sudoku; (3) determinar, después de cada entrada, si el juego ha finalizado correctamente. Algunas mejoras para la aplicación son, por ejemplo, (1) ofrecer la posibilidad de lograr una resolución automática del Sudoku; (2) determinar si el Sudoku introducido tiene solución o es un Sudoku imposible; etc.



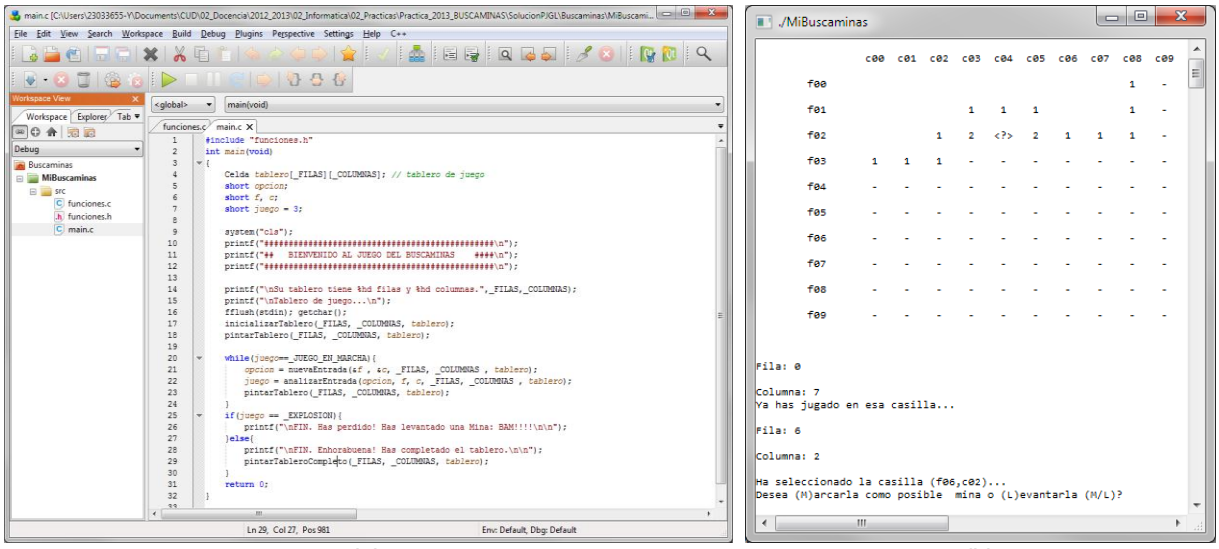
(a) (b)

Figura 1. Implementación del juego del Sudoku: (a) Entorno de desarrollo de código; (b) Ejemplo de salida por pantalla de la resolución automática de un Sudoku determinado.

3.2. Buscaminas

El Buscaminas es un conocido juego de ordenador que se encuentra disponible en la mayoría de sistemas operativos. Como ocurre con los Sudokus, el Buscaminas es muy popular y su programación es atractiva para los alumnos.

El objetivo del juego es despejar un campo minado sin detonar ninguna mina. El tablero del Buscaminas es un rectángulo formado por tantas filas y columnas como elija el jugador. En cada posición de fila y columna, hay una celda o casilla inicialmente cubierta. En algunas de esas celdas se ocultan minas. El jugador debe despejar todas las celdas del tablero excepto aquellas que ocultan una mina. Si el jugador da la orden de levantar una celda con mina, entonces pierde el juego.



(a) (b)

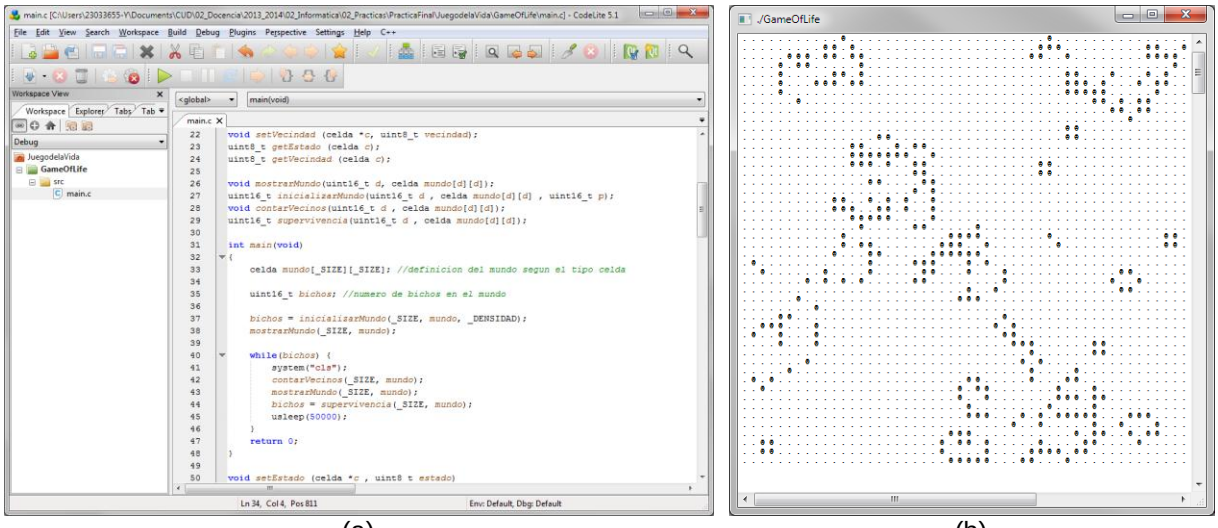
Figura 2. Implementación del juego del Buscaminas: (a) Entorno de desarrollo de código; (b) Ejemplo de salida por pantalla durante una partida del Buscaminas.

El código final presentado por el alumno debe ser capaz de (1) inicializar las posiciones de las minas en el tablero; (2) visualizar por pantalla el tablero; (3) permitir levantar y marcar/desmarcar una determinada casilla a través del teclado; (4) determinar, después de cada nueva entrada, si se ha explotado una mina y se ha terminado el juego, o si se ha finalizado el juego correctamente porque se han logrado identificar todas las minas o se ha logrado levantar todas las celdas limpias. También debe despejar automáticamente aquellas zonas del tablero seleccionadas por el jugador donde no hay ni una sola mina colindante.

3.3. Juego de la vida

Es un programa de simulación inventado en 1970 por J. H. Conway. El juego simula la evolución de una colonia de bichos que vive en un mundo cuadriculado cuyas casillas, en un momento dado, pueden estar vacías o habitadas por un solo ser. Está considerado como un juego de cero jugadores, lo que significa que la evolución del juego queda enteramente determinada por su estado inicial, y no se requiere en ningún momento la intervención del usuario. El comportamiento del sistema está definido por funciones de transición que determinan si los seres nacen, mueren o sobreviven. El Juego de la Vida finaliza cuando no se produce ningún cambio en la población o, por el contrario, no queda ningún bicho en la colonia. Hay otras configuraciones finales del juego más complejas de detectar, como el caso de la presencia de pulsadores o de navegadores: estos casos suponen un reto para los alumnos que no tiene solución trivial.

Debido a las configuraciones y patrones resultantes a lo largo del Juego de la Vida, éste ha suscitado un gran interés entre matemáticos y científicos en general, y particularmente en las ciencias de la computación. En nuestro caso, nos sirve para plantear un desafío de programación.



(a) (b)

Figura 3. Implementación del Juego de la Vida: (a) Entorno de desarrollo de código; (b) Ejemplo de salida por pantalla de la colonia de bichos durante una simulación del juego.

Para este TFA, el código final presentado por el alumno debe ser capaz de (1) inicializar la población de seres/bichos en un mundo, especificando el tamaño del mismo y la densidad inicial de seres; (2), mostrar por pantalla la evolución de los seres a lo largo de la cuadricula que define el mundo; (3) determinar cuándo el Juego de la Vida ha alcanzado un estado permanente o se ha llegado una configuración de una colonia vacía —sin ningún bicho—.

4. Conclusiones y trabajos futuros

En las asignaturas de introducción a la programación informática, es necesario hacer comprender al alumno que su reto de aprendizaje se alcanza programando, y no fundamentalmente estudiando un manual: «*para aprender a programar, hay que programar*». El profesor debe incorporar estrategias docentes que permitan incrementar el grado de motivación e implicación del alumno durante este proceso de aprendizaje. Siguiendo este enfoque, este artículo presenta una metodología basada en el desarrollo de juegos de ordenador como trabajos finales de asignatura. Se describe el proceso y las distintas etapas necesarias para la implantación de esta metodología; y se presentan algunos de los juegos que se han utilizado como trabajo final.

Algunas posibles líneas futuras a desarrollar son emplear mapas conceptuales (Jerez *et al*. 2012) que relacionen las competencias recogidas en la Guía Académica con las tareas a desarrollar en los trabajos finales y, además, incorporar mecanismos que permitan medir con precisión la influencia del desarrollo de juegos en la motivación de los alumnos para aprender un lenguaje de programación (Bahadir Kert & Fatih Erkoç, 2011).

Bibliografía y Referencias.

Bueno, D., Garralón, J., Jeréz, J.M., Maña, A. (2001) Aprendizaje Lúdico en Laboratorio de Programación. *VII Jornadas sobre la Enseñanza Universitaria de la Informática*. Mallorca: España.

Bahadir Kert, S. y Fatih Erkoç, M. (2011) Computer games for as programming education tool. *1st International conference The Future of Education*. Florence: Italy.

Frittelli, V., Tartabini, M., Teicher, R., Steffolani, F., Serrano, D., Fernández, J., Bett, G., Strub, A. (2013) Desarrollo de Juegos como Estrategia Didáctica en la Enseñanza de la Programación. *1er Congreso Nacional de Ingeniería Informática/Sistemas de Información (CoNAIISI 2013)*. Córdoba: Argentina.

Jerez, J.M., Bueno, D., Molina, I., Urda, D., Franco, L. (2012) Improving Motivation in Learning Programming Skills for Engineering Students. *International Journal of Engineering Education*, vol. 28, no. 1, pp. 202-208.

Anexo I. Rúbrica para la Evaluación del Juego de la Vida

EVALUACIÓN TFA: JUEGO DE LA VIDA		
ALUMNO		
	Máxima puntuación	**Puntuación obtenida**
Presentación e implementación básica		
Claridad del código	0,1	
Uso de comentarios	0,1	
Archivos de código	0,1	
Presentación en la ejecución de la aplicación	0,1	
Entrevista: Evidencia de la autoría	0,2	
El Juego evoluciona correctamente	0,2	
TOTAL PRIMERA SECCIÓN	0,8	
Implementación de nuevas funciones		
Estructura toroidal de la rejilla	0,1	
Redefinir requisitos de nacimiento y muerte	0,2	
Variaciones para definición de estado inicial	0,2	
Determinar el final del juego	0,3	
Otras mejoras introducidas	0,4	
TOTAL SEGUNDA SECCIÓN	1,2	
TOTAL EVALUACIÓN TFA	2,0	