Cuaderno ACTIVA, ISSN 2027-8101. No. 4, Julio-diciembre 2012, pp. 91-106
Tecnológico de Antioquia, Medellín (Colombia)

Revisión bibliográfica de los juegos digitales para el aprendizaje de la programación orientada a objetos

Digital games literature review for object-oriented programming learning

Ricardo de J. Botero Tabares
Magíster en Ingeniería
Profesor Facultad de Ingeniería
Tecnológico de Antioquia – Institución Universitaria
rbotero@tdea.edu.co

Recibido: 31 de agosto 2012
Aprobado: 10 de octubre 2012

Resumen

El aprendizaje de la Programación Orientada a Objetos (POO) en las instituciones de educación superior conlleva el estudio de conceptos cuya comprensión se puede agilizar con el uso de herramientas lúdicas basadas en software, como los juegos digitales. El presente artículo expone las principales herramientas existentes en el mercado del software para el desarrollo de videojuegos y algunos marcos de trabajo para el aprendizaje lúdico de la programación. Al final se presentan experiencias universitarias con los juegos digitales para el aprendizaje de la programación de ordenadores.

Palabras clave: juegos digitales, programación orientada a objetos, juegos y programación.

Abstract

Learning Object-oriented programming (OOP) in higher education institutions entails the study of concepts whose understanding can be streamlined by using software-based recreational tools, such as digital games. This paper explains the main existing tools in the market of software development for videogames, and several working frameworks to learn programming in a fun way. At the end, some college experiences with digital games to learn computer programming are presented.

Keywords: digital games, object-oriented programming, games and programming.

1. Introducción

El juego y las llamadas máquinas inteligentes, autómatas y computadoras de todo tipo, han ido evolucionando juntos desde la antigüedad y son reflejo de una forma particular de racionalizar la experiencia del ser humano (Borja, 2007). El uso pedagógico de los juegos para el aprendizaje en varias áreas del conocimiento humano aumenta cada día. Así, cuando las matemáticas se aprenden con juegos se llega a hablar de las *matemáticas recreativas* (Perelman, 1968), porque se fortalece una actitud positiva hacia esta área del conocimiento; dicha modalidad de aprendizaje de las matemáticas no está enmarcada en el currículo tradicional, porque usualmente se piensa que una matemática formal no puede ser entretenida, asumiendo lo formal como contrario a entretenido, es decir, lo monótono. Sin embargo, parte de la matemática se ha desarrollado a partir de juegos, como el problema de los puentes de Köninsberg que dio origen a la teoría de grafos y los juegos de azar que originaron las teorías de probabilidad y combinatoria.

Los juegos tradicionales se conectan con los deseos lúdicos espontáneos de estudiantes en formación primaria y tienen propiedades que favorecen el aprendizaje de las matemáticas (Olfos y Villagrán, 2000). Algunos de ellos, como la escoba (y la escoba fraccionaria), sirven para ejercitar la suma; con el dominó, el ajedrez y Nim y Reversi se practican estrategias; con los juegos de cartas se aplican estrategias de resolución de problemas de agrupamiento y se solucionan problemas parciales; con el juego de la oca, el trivial y el bingo se pueden enseñar conceptos; con el póker se puede iniciar el estudio de las probabilidades; los juegos digitales tipo Tetrix, los simuladores y las batallas de velocidad, entre otros, sirven para mejorar la habilidad espacial.

El diseño de juegos para computador incluye muchos aspectos de computación como gráficas de computador, inteligencia artificial, interacción humano-computador, seguridad, programación distribuida, simulación e ingeniería del software, incluso también se han utilizado en la exposición de conceptos relacionados con las artes liberales, las ciencias sociales y la psicología.

Con el objetivo de lograr una visión general sobre el estado del arte de los juegos digitales en el aprendizaje de la programación orientada a objetos —POO—, se divide este artículo de la siguiente manera: en el numeral 2 se listan las principales herramientas para el desarrollo de videojuegos; luego se exponen algunos marcos de trabajo representativos para el aprendizaje de la programación; posteriormente se presenta una lista de juegos digitales para aprender a programar computadoras, utilizados en algunas instituciones de educación media y superior, tanto nacionales como extranjeras, y, finalmente, se presentan las conclusiones.

2. Marco teórico y trabajos previos

2.1. Herramientas para el desarrollo de juegos

Cualquier lenguaje de programación de propósito general, como C++, Java, C# o Visual Basic.NET, se puede utilizar para crear juegos digitales en 2D y 3D, entre estos últimos los videojuegos. Sin embargo, existen herramientas de uso específico para el desarrollo de videojuegos, resumidas en la siguiente tabla (Genbeta Dev, en línea).

Tabla 1. Algunas herramientas para el desarrollo de videojuegos

Herramienta	Breve descripción
2D Fighter Maker	Especial para crear juegos al estilo de riñas en 2D.
3D Game Studio	Es una aplicación extraordinaria para la producción de juegos en dos y tres dimensiones.
Adventure Maker	Es una aplicación con la que se pueden crear aventuras gráficas propias en primera persona, sin necesidad de tener conocimientos de programación.
Antiryad GX	Recomendable para personas que quieren realizar proyectos independientes, ya que se puede hacer, por ejemplo, un juego para Android que no necesita tantos recursos y tiempo como un juego para PC o PS2.
Basic4GL	Programacion en BASIC para Windows con soporte OpenGL (2D y 3D).
Blender Engine Game	Blender es un programa informático multiplataforma, dedicado especialmente al modelado, la animación y creación de gráficos tridimensionales.
Blink 3D	Es una herramienta para la creación de entornos tridimensionales. Los entornos son 3D inmersivos y pueden ser vistos con el visor Blink 3D desde la web o en un ambiente local.
Blitz3D	Paquete para programar juegos 2D/3D. Basado en Basic.
Construct	Construct parte de la misma base que Game Maker: un entorno de programación gráfico en el que se colocan objetos sobre el escenario para adjudicarles un comportamiento determinado.
Crystal Space	Es un espectacular motor gráfico totalmente gratuito, avanzado y desarrollado en lenguaje C++. Con él se pueden crear todo tipo de juegos en 3D.
Cube 3D Engine	Software libre con opción de múltiple jugador, que permite editar la geometría, los mapas y demás elementos interactivos.
E-Adventure	Es una herramienta de autoría para la creación de juegos, diseñada especialmente para propósitos educativos.

Adobe Flash	Adobe Flash Professional es el nombre o marca comercial de uno de los programas más populares de la casa Adobe. Flash es una aplicación para la creación y manipulación de gráficos vectoriales con posibilidades de manejo de código mediante el lenguaje ActionScript. Sus programas hermanos son Adobe Illustrator y Adobe Photoshop.
Entidad 3D	Completo sistema para el desarrollo de juegos 3D tipo Quake sin necesidad de programación.
Game Maker	Excelente programa para crear videojuegos de plataformas y muchos otros en 2D.
Glest	Juego de estrategia (RTS) en 3D, completamente personalizable y gratuito.
JClic	Es un conjunto de aplicaciones informáticas para realizar rompecabezas, asociaciones, ejercicios de texto, palabras cruzadas, y muchas otras actividades educacionales (JAVA).
Kaneva	Sistema para la creación de juegos multijugador en línea (MMO games).
Microsoft XNA Game Studio	Herramienta gratuita para el desarrollo de videojuegos para Windows y XboX 360.
Multimedia Fusion	Herramienta para crear juegos, aplicaciones multimedia o presentaciones sin programacion.
PPTactical Engine	Pequeño programa para la creación de juegos de estrategia en tiempo real (RTS).
Pygame	Permite escribir videojuegos en el lenguaje de programación Python.
RPG Maker	Para crear juegos rol 2D.
RPG Toolkit	Programa dedicado a la creacion de juegos en 2D del tipo "Role Playing Games" (RPG's).
The lost Realm of Anoria	Es un programa para crear juegos propios multijugador RPG o juegos similares.
Torque Game Engine	Poderosa máquina 3D para la creacion de videojuegos.
Unreal Engine	Ofrece las plataformas y las herramientas necesarias para crear proyectos 3D.
Verge	Programa para crear juegos de tipo RPG.
Visionarie	Permite la creacion de aventuras gráficas en 2D.

2.2. Marcos de trabajo para el aprendizaje lúdico de la programación

La POO se ha visto beneficiada por una serie de herramientas o marcos de trabajo que facilitan su aprendizaje de una manera lúdica. Algunas de ellas son *Scratch*, *Greenfoot*, *Alice*, *Robocode*, *Minueto*, *Jeroo*, *RoboMind*, *Karel*, *MSWLogo* y *C-jump*. Una breve descripción de cada una de ellas se presenta en los párrafos siguientes.

Scratch (The Lifelong Kindergarten Group, en línea) contiene un lenguaje de programación simple pero eficiente, donde se utiliza la metáfora de "piezas encajables" para animar objetos que se encuentran en la pantalla, con un uso muy sencillo e intuitivo. *Scratch* presenta un entorno de desarrollo que muestra a simple vista todos los elementos necesarios: escenario, objetos y elementos del lenguaje. Se puede tener tantos escenarios y objetos como se desee, utilizando aquellos que ya están disponibles con la instalación estándar de la herramienta, o bien creando otros nuevos. Este es un factor motivacional más a la hora de trabajar con el alumnado desde edades tempranas. Los elementos disponibles no son únicamente dibujos, sino también sonidos. Se pueden utilizar los que vienen por defecto, añadir sonidos nuevos desde la web del proyecto, o incorporar nuestras propias grabaciones, bien a través de la grabadora incorporada en el entorno, o a través de cualquier otra herramienta externa. Prácticamente todo se lleva a cabo moviendo y soltando elementos con el ratón, y modificando con el teclado únicamente valores numéricos, textos y otros objetos. Un ambiente típico de *Scratch* se observa en la Figura 1, donde se aprecian diversas áreas de trabajo para el programador, como el editor, y la ventana de objetos y pestañas para controlar scripts y variables.

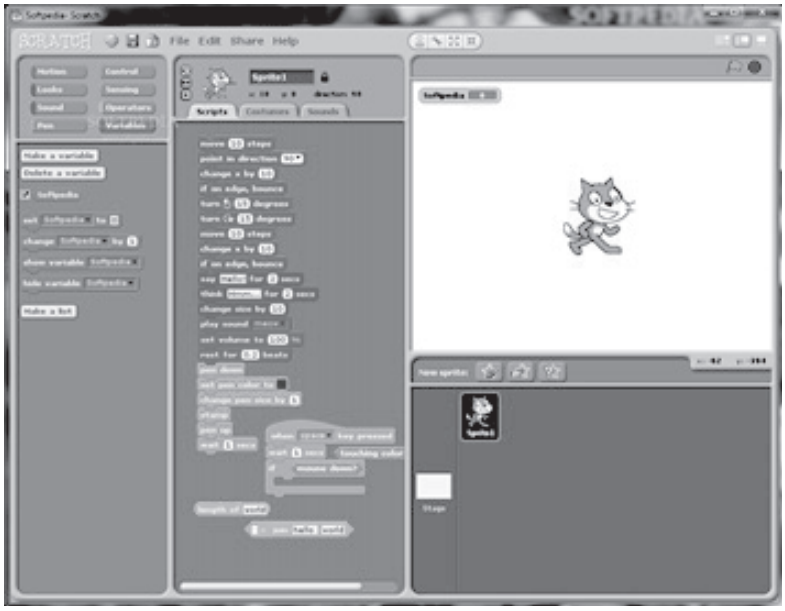

Figura 1. Pantalla típica de *Scratch*

Greenfoot (Kölling, 2008) es un entorno integrado de desarrollo y programación que facilita enormemente la escritura de juegos y simulaciones en lenguaje Java. El programa incluye por defecto un buen número de escenarios de ejemplo y documentación para que los menos expertos puedan familiarizarse rápidamente con su manejo. *Greenfoot* tiene un doble propósito: servir como herramienta didáctica de aprendizaje de programación en Java, o simplemente para crear un juego interesante publicable en la web. Este entorno estaba inicialmente orientado a los usuarios de menor edad, pero ahora se trata de una aplicación apta para personas de cualquier edad. Como se ilustra en la Figura 2, *Greenfoot* combina por igual un entorno de desarrollo visual con forma de rejilla bidimensional adecuado para los principiantes, y un editor convencional para desarrollar código al estilo tradicional con su visor de clases, editor que permite compilar y ejecutar programas.

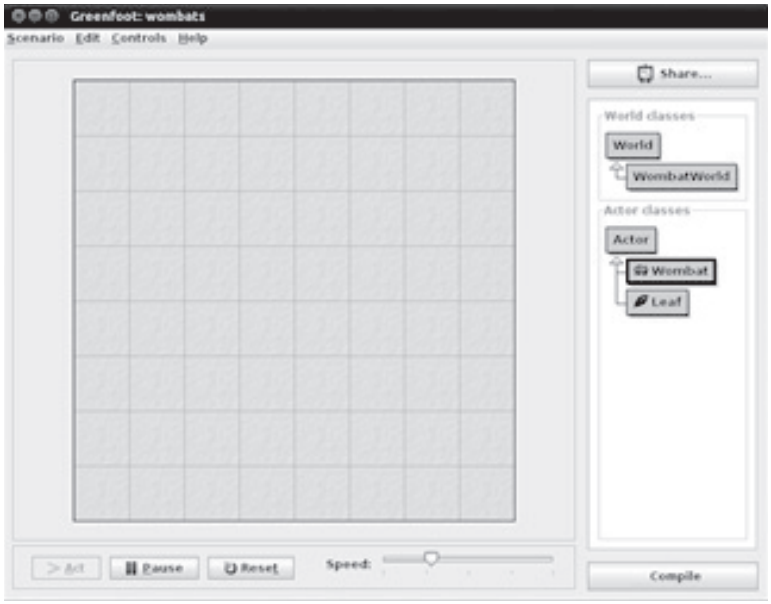

Figura 2. Entorno de desarrollo visual de *Greenfoot*

Alice (McKenzie, 2007) propicia un ambiente de programación en 3D donde los objetos son actores (humanos, animales, objetos de fantasía) que habitan y actúan en mundos (reales o imaginarios). Los estudiantes pueden utilizar *Alice* para la programación de animaciones o juegos interactivos. La pantalla de bienvenida de Alice, expuesta en la Figura 3, proporciona pestañas para el estudio de un tutorial, mundos o proyectos recientes, plantillas, ejemplos y apertura de mundos.

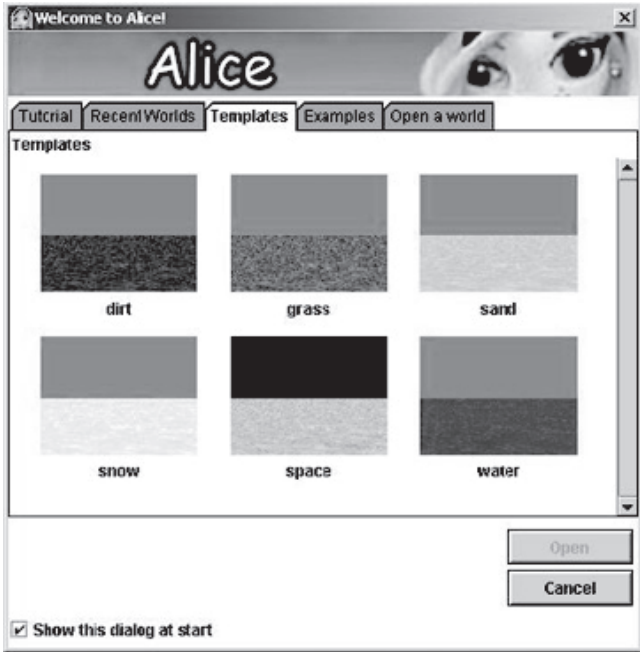

Figura 3. La pantalla de bienvenida de Alice

Robocode (Kobayashi *et al.,* 2003) es un entorno gratuito de simulación de guerras de robots desarrollado por Alphaworks de IBM, en el que hay que programar en Java un tanque para combatir en el campo de batalla contra tanques programados por otros jugadores; una pantalla de juego se aprecia en la Figura 4. *Robocode* es un juego bastante divertido que se emplea didácticamente en un nivel básico para enseñar a programar en Java (cómo escribir código Java, el manejo de una API, manejo de eventos, clases internas, herencia, etc.), o en un nivel más avanzado para la simulación de sistemas software autónomos dotados de inteligencia con una estrategia y unas tácticas diseñadas para ganar la batalla.

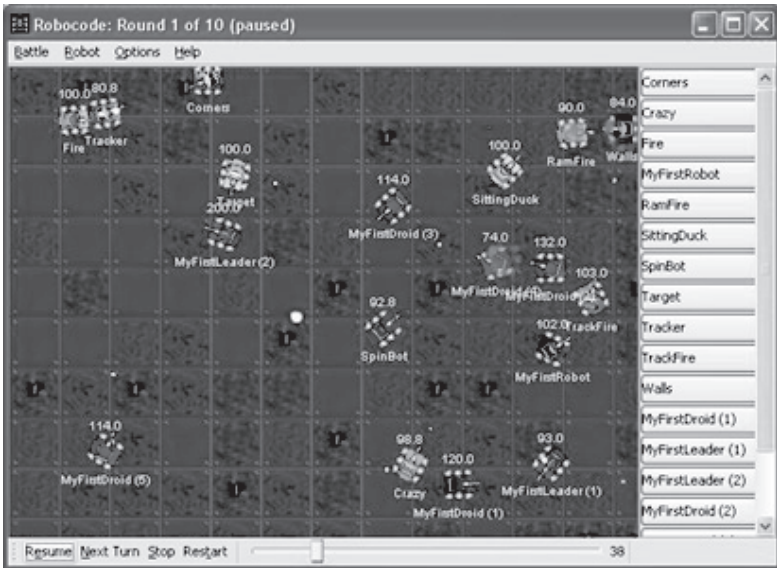

Figura 4. Una pantalla del juego *Robocode*

Minueto (Denault, 2005) es de tipo multiplataforma, con el objetivo de simplificar el proceso de desarrollo de juegos por medio del encapsulamiento de tareas complejas de programación, tales como la programación de gráficos, de audio, de teclado y de mouse, en objetos de manipulación sencilla. Este diseño simple, y la gran cantidad de documentación, permiten que los estudiantes comiencen en un breve lapso el desarrollo de juegos. La mayoría de las herramientas de desarrollo se han diseñado para la industria profesional, y su difícil curva de aprendizaje las hace inadecuadas para el mundo académico, por lo tanto hacen falta herramientas especializadas para el desarrollo de juegos académicos. *Minueto* cumple este cometido, ya que está enfocado para la población de no graduados de ciencias de la computación, lo que permite que los estudiantes desarrollen rápidamente juegos de computador no triviales por medio de la simplificación de varias de las materias de programación inherentes a los mismos como gráficos, sonido, redes y comunicación con el usuario y la máquina. Este software es un marco de trabajo modular; sus componentes principales incluyen un motor gráfico en 2D y un sistema de recepción del teclado. Esta infraestructura modular permite que el programador entregue servicios adicionales cuando crea los módulos de expansión, por ejemplo los de soporte de sonido o de redes.

Jeroo (Dorn y Sanders, 2004) es una herramienta pedagógica que entrega una introducción adecuada y amable a la POO. Esta herramienta se ha desarrollado para ayudar a los programadores novatos en la solución de problemas, mediante una semántica adecuada de las estructuras de control fundamentales y la aplicación de métodos de escritura que soporten la descomposición funcional de las tareas. La sintaxis de *Jeroo* provee una transición suave hacia Java, C++ o C#. La interfaz de usuario tiene una ventana en la cual todo esta visible. El sobresaltado del código fuente, las animaciones sencillas y el panel de estatus continuamente actualizado (ver Figura 5) proveen un ambiente rico de enseñanza y aprendizaje, siendo ideal para el aprendizaje de programadores novatos.

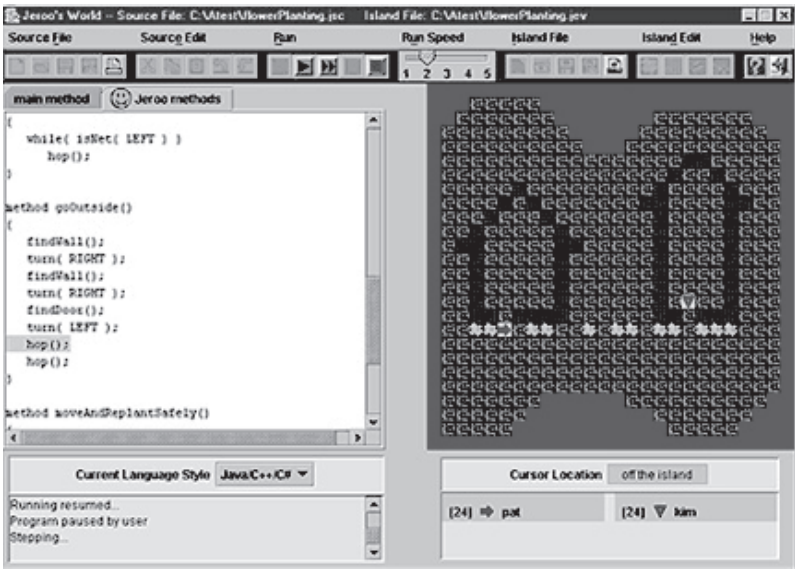

Figura 5. Entorno integrado de *Jeroo*

RoboMind (ResearchKitchen, en línea) es un entorno de desarrollo pensado para acercar los más jóvenes a la programación estructurada. Mediante un abanico de órdenes (agarrar un objeto, mirar, girarse, etc.) se puede generar un programa que haga cosas tan variadas como buscar un punto blanco o resolver un laberinto. Como se observa en la Figura 6, la interfaz de *RoboMind* se divide en tres paneles: un área de escritura de programas, una representación gráfica del robot en su ambiente y un panel de mensajes de error. *RoboMind* apuesta por un lenguaje de propósito específico, una desventaja que se olvida pronto en cuanto vemos al robot dar sus primeros pasos por el mapa.

Figura 6. Interfaz de *RoboMind*

Karel (Pattis, 1981) es una herramienta de aprendizaje que presenta los conceptos de una forma visual, lo cual es menos abstracto que programar en un lenguaje como Pascal o C. El robot *Karel* fue introducido por Richard Pattis para motivar la programación en el lenguaje Pascal. Una pantalla de *Karel,* expuesta en una de las versiones oficiales de la Olimpiada Mexicana de Informática (OMI), se presenta en la Figura 7.

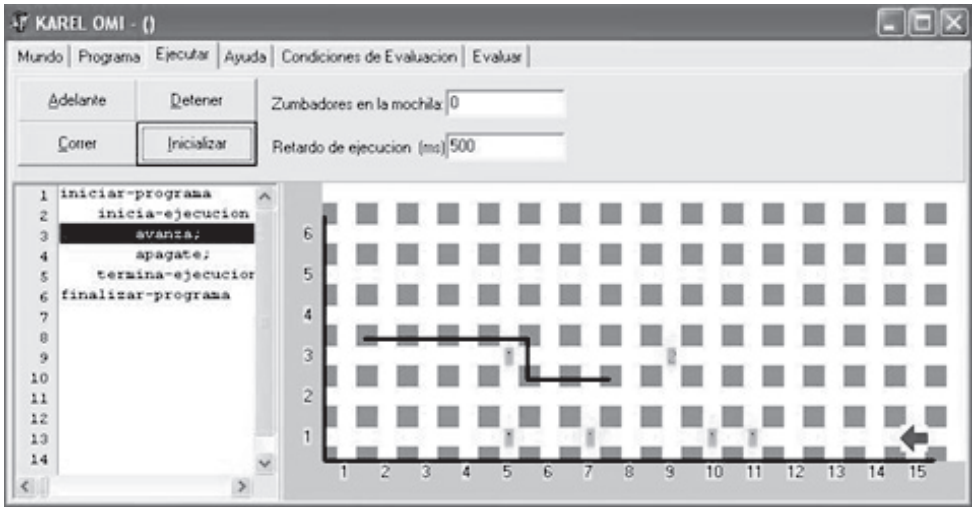

Figura 7. Pantalla Karel en una versión OMI

MSWLogo (An Educational programming language, en línea) es un lenguaje de programación muy sencillo que se utiliza frecuentemente en el ámbito de la educación secundaria. La Figura 8 expone una pantalla de una versión de *MSWLogo* traducida por el Centro Nacional de Información y Comunicación Educativa —CNICE— perteneciente al Ministerio de Educación y Ciencia de España.

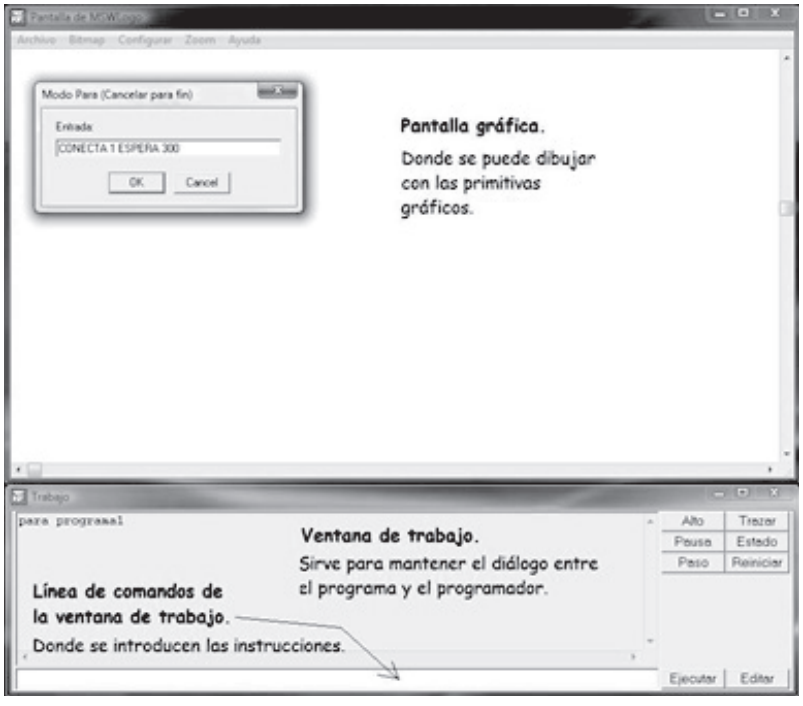

Figura 8. Pantalla MSWLogo de CNICE

C-jump (Singh y Singh, 2007) es el nombre de un juego de mesa ideado en 1999 por el programador ruso Igor Kholodov, que busca enseñar de forma lúdica a niños lenguajes de computación como Java o C++. Una pantalla de tour virtual por el juego se expone en la Figura 9.

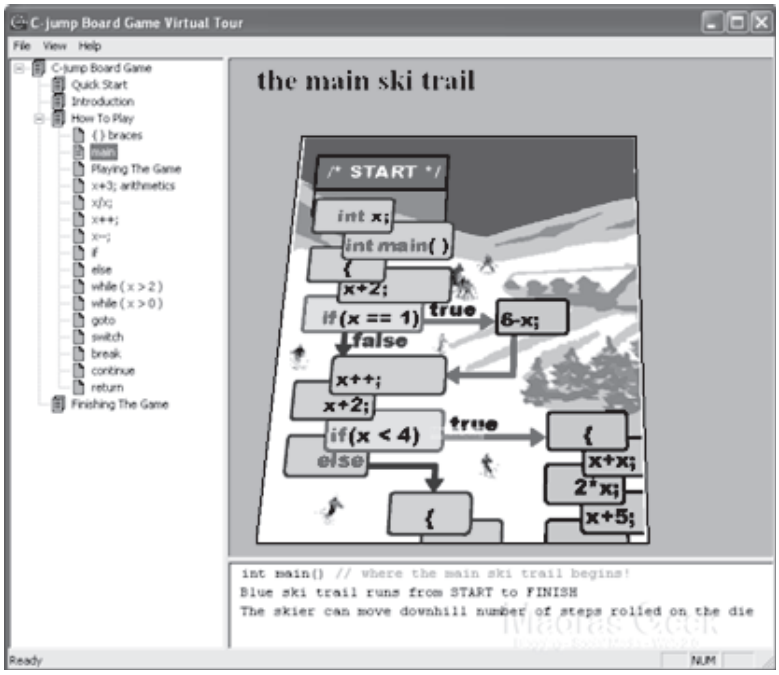

Figura 9. Tour virtual por el juego de mesa C-jump

3. Experiencias universitarias con los juegos digitales para el aprendizaje de la programación

En el departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga se presenta un enfoque pedagógico desarrollado por el equipo docente de la asignatura Laboratorio de Programación de la Ingeniería Técnica de Telecomunicación (Bueno *et al.,* 2001). El enfoque, basado en el uso de juegos de ordenador, incluye tres prácticas: la primera intenta asentar los conocimientos sobre estructuras de datos simples, mediante juegos con una entrada-salida simple en modo texto. La segunda práctica tiene como objetivo el manejo de estructuras de datos complejas mediante la implementación del juego de la serpiente, con el cual los alumnos comienzan aprendiendo a controlar las teclas especiales de movimiento y las coordenadas de la pantalla. La tercera y última práctica introduce el trabajo con librerías para la construcción de tipos abstractos de datos e interfaces gráficas de usuario, mediante el juego de los barquitos, donde el alumno debe manejar varias estructuras de arreglos de registros.

En la Escuela Colombiana de Ingeniería "Julio Garavito" se ha llevado a cabo una experiencia a través de juegos para la enseñanza del paradigma orientado a objetos (Cadavid, 2006). A los estudiantes se les dio una visión de cómo darle a una aplicación características de extensibilidad y adaptabilidad haciendo uso de los elementos de este paradigma, siempre pensando en elementos reutilizables, herencia de clases, polimorfismo y encadenamiento

dinámico. Como práctica lúdica de programación, los estudiantes desarrollan un modelo de clases que represente una pista de baile, compuesta de bailarines que pueden realizar una serie consecutiva de pasos o "estilo" de baile, haciendo uso de una serie de métodos básicos definidos en la clase *Bailarín*, como subir y bajar tanto los brazos como las piernas. El proyecto final es el juego de invasores, cuyo propósito es determinar un marco de trabajo (framework) que tuviera implementada toda la lógica de un juego donde una nave controlada por el jugador se enfrenta a una diversidad de elementos, como asteroides y otros elementos autónomos que se desplazan con determinados patrones que atacan al jugador en caso de detectarlo.

En el Departamento de Sistemas Informáticos y Programación de la Universidad Complutense de Madrid se presenta una experiencia en un curso de Laboratorio de Programación III, en la que se propone la implementación de Sudokus en Java (Recio *et al.,* 2006). La experiencia permite poner en práctica conceptos de estructuras de datos y algoritmia, así como patrones de diseño y programación de dispositivos móviles.

En la Escuela Universitaria de Ingeniería Técnica Industrial de la Universidad Politécnica de Madrid, el Grupo de Sistemas Telemáticos Aplicados a la Educación —GSITAE— del Departamento de Electrónica, Automática e Informática Industrial, ha optado por desarrollar un método docente para la asignatura de Informática Industrial, basado en el diseño y desarrollo de una aplicación gráfica interactiva a modo de videojuego, que incluye los simuladores y otras herramientas de ingeniería (Rodríguez-Lozada, 2008). La enseñanza de la POO se ve facilitada en gran medida por un enfoque unificado de la asignatura que engloba desde las prácticas realizadas en laboratorio a las tutorías, el desarrollo de trabajos personales por los alumnos, la docencia teórica e incluso el material docente. A raíz de los buenos resultados obtenidos, se cree que la motivación y el interés que despierta este enfoque en el alumnado redundan claramente en una mejora de la docencia y en los resultados académicos.

En la Facultad de Ingeniería de la Universidad ORT Uruguay se esbozan diversas líneas de trabajo en la cátedra Programación (Friss, 2008). Algunas de las áreas trabajadas son: diseño de entornos de aprendizaje basados en la Gestión del Conocimiento, uso de morfismos, diseño y uso de *Kinesthetic Learning Activity* y uso de *Scratch*. Además, se están evaluando nuevas herramientas como *Alice 3.0*.

En la Universidad de Utrecht se usan robots para el aprendizaje de la programación (Overmars, 2004). Su principal desventaja es el precio y sus posibilidades limitadas de programación. Sin embargo, los robots son vehículos ideales para la explicación de conceptos tales como "sensado", ciclos de control y tareas paralelas.

En el Departamento de Sistemas de Información de la Universidad de Melbourne, los estudiantes de los primeros niveles deben completar un juego como materia de un semestre (Goschnick y Balbo, 2005). Se tiene en cuenta que la mayoría de los estudiantes de informática no están motivados para la programación, pero el desarrollo de videojuegos los animará en dicho aspecto.

En el Departamento de Ciencias de la Computación y la Educación de la Facultad de Educación de la Universidad de Corea, se trabaja con un juego de cartas basado en Small talk (Seung Bum *et al.*, 2006); lo distintivo de este juego es que se basa en reglas de conversación de uso diario. Los participantes pueden entender los conceptos de la orientación a los objetos en forma evolutiva sin un conocimiento previo de la programación orientada a objetos. En el transcurso del juego, los participantes no utilizan términos técnicos como clase, herencia, instancia, etc., sin embargo usan escenarios que los conducen naturalmente al desarrollo de los conceptos por sí mismos y con sus propios términos.

En las universidades de Northwestern y de Victoria (USA) se propone un método para evaluar la pedagogía basada en grupos con el ánimo de afianzar las capacidades de programación de los estudiantes y buscar su inserción adecuada en la industria (Rankin *et al.*, 2007). El método destaca que se debe

efectuar un mayor énfasis en la incorporación de herramientas industriales para el diseño de juegos en el aula, en vez de aplicar valoraciones de las estrategias pedagógicas.

En la Universidad de Kettering los juegos de computador deben ser construidos en un ambiente divertido con el objetivo de atraer a nuevos estudiantes, aunado a la construcción sólida de herramientas de programación (Baibak y Agrawal, 2007). Se sugieren diferentes clases de juegos para ser asignados en un curso. El diseño de juegos requiere diferentes clases de algoritmos para la gestión de bases de datos: algoritmos de colisión, algoritmos de entrada, algoritmos para el hallazgo de trayectorias, algoritmos gráficos, etc. Un juego completo refleja una comprensión significativa de todas sus porciones algorítmicas con sus ideas subyacentes.

En el Laboratorio de Computación de la Universidad de Kent, muchos intentos se están llevando a cabo para hacer que la introducción a la POO sea menos abstracta y teórica. Se están aplicando diversas técnicas de interacción para dar a los estudiantes experiencias atractivas y concretas con objetos (Kölling & Henriksen, 2005). Recientemente, el entorno *Greenfoot* se ha propuesto como otro paso en este desarrollo.

En el Departamento de Ciencias de la Computación de la Universidad del Estado de Sam Houston se utiliza el ambiente *BlueJ* para desarrollar una implementación funcional del juego de cartas *Blackjack* (Kouznetsova, 2007). Como los conceptos se presentan en el contexto de una aplicación familiar y divertida, la tarea incrementa su nivel de compromiso. Adicionalmente, ya que a los estudiantes se les permite descubrir por sí mismos las ventajas del diseño orientado a objetos, desarrollan un mejor entendimiento del material. Todas las tareas se implementan con el uso del ambiente de desarrollo integrado de *BlueJ,* el cual reduce la sobrecarga del aprendizaje del código de edición, la compilación y la ejecución; este programa también ayuda a que los estudiantes creen fácilmente gráficos incorporables al proyecto. Las ventajas del ambiente de *BlueJ* incluyen su representación gráfica de las clases y objetos en un proyecto con el cual pueden interactuar los estudiantes.

En la Escuela de Negocios Gabelli de la Universidad Roger Williams se está usando *Alice* en remplazo de lenguajes tradicionales como JAVA o Visual Basic (McKenzie, 2007).

En la universidad del estado de Sam Houston se incorpora un juego disfrutable, llamado *Robocode*, que permite el desarrollo de robots virtuales en un juego diseñado en Java (Hartness, 2004). En una clase de inteligencia artificial se entregan a los estudiantes herramientas para el desarrollo de versiones prácticas de los algoritmos, por consiguiente los estudiantes aprecian mejor la teoría y desarrollan una mayor confianza en su aprendizaje. La inteligencia artificial se puede beneficiar de un motor de juegos que puede ser usado o modificado fácilmente. Se ha comprobado que los estudiantes hallan la inteligencia artificial más interesante y accesible con ejemplos y proyectos.

En la Escuela de Ciencias de la Computación de la Universidad McGill se trabaja con *Minueto* (Denault & Kienzle, 2006), en un marco de trabajo para el desarrollo de juegos en Java diseñado específicamente para estudiantes de pregrado.

En las universidades del estado de Iowa y del estado del Noroeste de Missouri se usa *Jeroo* para introducir la POO (Dorn y Sanders, 2004).

En el Departamento de Ciencias de la Información y Tecnología de la Universidad Estatal de Pennsylvania, campus Altoona, se tiene un modelo pedagógico denominado PBL (Aprendizaje Basado en Problemas) que enfatiza en el rol de los problemas de la vida diaria aunado con procesos de descubrimiento colaborativos para el aprendizaje (Ryoo *et al.,* 2008). En un ambiente típico, a los estudiantes se les proporciona un problema desafiante y real de tamaño significativo, el cual es relevante a los objetivos de aprendizaje de un curso dado. Los estudiantes son animados para que resuelvan el problema en un semestre, utilizando *Alice* y con la mínima ayuda del instructor del curso. Fuera del método tradicional de enseñanza orientado a

la clase magistral, PBL aunado a *Alice*, se hace mayor énfasis en el rol del instructor como facilitador, para la preparación de problemas significativos e interesantes y para la creación y organización de los materiales del curso, de manera que los estudiantes tengan la dosis justa de información en cada clase para que de forma incremental desarrollen la solución final.

En el semillero de investigación STC Célula TdeA. Net de la Facultad de Ingeniería del Tecnológico de Antioquia – Institución Universitaria, se trabaja con el juego *CoquitoDobleO* (Botero, 2012) con el objetivo de incentivar el aprendizaje de conceptos básicos del paradigma de programación orientado a objetos, como clase, objeto, método, sobrecarga de métodos, herencia y polimosfismo, estudiados en las asignaturas Lógica de Programación I y Construcción del Software I.

4. Conclusiones y trabajos futuros

Los procesos de enseñanza-aprendizaje de la POO en instituciones de educación superior se deben dinamizar con herramientas que aumenten los grados de motivación y comprensión de temas en los estudiantes, y que les aporten a los profesores elementos didácticos adicionales para mejorar la docencia apoyada en tecnologías de la información y la comunicación. Este trabajo complementario se puede realizar con herramientas para el desarrollo de videojuegos (*Blender Engine Game*, *Microsoft XNA Game Studio*, *Unreal Engine*), con marcos de trabajo para el aprendizaje de la programación (*Scratch*, *Greenfoot*, *Alice*, *Robocode*) y con el desarrollo de prácticas en lenguajes de programación de propósito general como Java o C#, encaminadas al desarrollo de aplicaciones basadas en juegos (*CoquitoDobleO*, sudokus en Java, programación de robots, etc.).

El estudio de estas herramientas lúdicas se puede promover en los semilleros de investigación o aun en las asignaturas prácticas de la estructura curricular, donde el desarrollo de aplicaciones de software sea el eje primordial. Dicho estudio debe ser gradual y permanente, de tal forma que los juegos digitales se incluyan en el currículo y en la evaluación, desde asignaturas de introducción a la programación en los primeros niveles hasta materias de semestres medios y avanzados.

Referencias

Baibak, T. & Agrawal, R. (2007). *Programming games to learn algorithms.* ASEE Annual Conference & Exposition. Honolulu, Hawaii.

Borja, F. (2007). Del juego antiguo al juego de computadora. Papel histórico del juego en el desarrollo de la tecnología digital. *Revista de comunicaciones y tecnologías emergentes, Vol. 4, No. 2.*

Botero, R. (2012). La lúdica de juegos en el aprendizaje de la programación orientada a objetos: un prototipo en C#. Tesis de maestría. Universidad EAFIT, Escuela de Ingenierías, Medellín.

Bueno, D. *et al.* (2001). *Aprendizaje lúdico en laboratorio de programación.* VII Jornadas sobre la Enseñanza Universitaria en Informática. Palma de Mallorca.

Cadavid, H. (2006). *Desarrollo de juegos como base para la comprensión de temas fundamentales de la programación orientada a objetos*. VIII Congreso Colombiano de Informática Educativa. Cali, Colombia.

Denault, A. (2005). Minueto: An under graduate teaching Development framework. *Master thesis. School of Computer Science, McGill University, Montreal.*

Denault, A. & Kienzle, J. (2006). *Minueto: A game development framework for teaching Object-Oriented software design techniques.* School of Computer Science, McGill University. Disponible en: http://gram.cs.mcgill.ca/papers/denault-06-minueto.pdf

Dorn, B. & Sanders, D. (2004). *Using Jeroo to introduce object-oriented programming.* 33rd ASEE/IEEE Frontiers in Education Conference, Boulder, Colorado.

Friss, I. (2008) Scratch: Applications in Computer Science. 1.38th ASEE/IEEE Frontiers. In Education Conference, Saratoga Springs, NY.

Genbeta:Dev (s.f.). Desarrollo y software. Disponible en: http://www.genbetadev.com/

Goschnick, S. & Balbo, S. (2005). *Game-first programming for information systems students.* In Proceedings of the Second Australasian Conference on Interactive Entertainment (IE 2005) (pp. 71-74). Sydney, Australia: Creativity & Cognition Studios Press.

Hartness, K. (2004) Robocode: Using games to teach artificial intelligence. *Journal of Computing Sciences in Colleges,* Vol. 19, No. 4.

Kobayashi, K., Uchida & Watanabe, K. (2003). *A study of battle strategy for the Robocode.* SICE Annual Conference in Fukui, Fukui University.

Kölling, M. & Henriksen, P. (2005). Game programming in introductory courses with direct state manipulation. *ACM SIGCSE Bulletin,* Vol. 37, No. 3, pp. 59-63.

Kölling, M. (2008). *Greenfoot: A highly graphical IDE for learning object-oriented programming.* In Proceedings of the 13th annual conference on Innovation and technology in computer science education (ITiCSE '08). ACM, New York, NY, USA.

Kouznetsova, S. (2007). Using Blue J and Blackjack to teach object-oriented design concepts in CS1. *Journal of Computing Sciences in Colleges*. Vol. 22, No. 4.

McKenzie, W.B. (2007). *An alternative approach to introductory object oriented programming: A case study in programming with Alice.* Proceedings of the AIS SIG-ED IAIM 2007 Conference, Montreal, Quebec.

Olfos A., Raimundo y Villagrán, C. *Eduvina (2000). Actividades lúdicas y juegos en la iniciación al álgebra.* (Ponencia). V Jornada de Innovación en la Enseñanza de la Matemática, Universidad de Viña del Mar, Chile.

Overmars, M. (2004). Learning Object-Oriented Design by creating games. Utrecht University. Disponible en: http://www.cs.uu.nl/research/techreps/repo/CS-2004/2004-057.pdf

Pattis, R. (1981). *Karel the Robot: A Gentle Introduction to the Art of Programming.* New York: John Wiley & Sons, Inc.

Perelman, Y. (1968). *Matemáticas recreativas.* Barcelona: Ediciones Martínez Roca.

Rankin, Lechner, T. & Gooch B. (2007). *Team-based pedagogy for CS102 using game design.* The 34th International Conference and Exhibition on Computer Graphics and Interactive Techniques SIGGRAPH '07, San Diego, California.

Recio, J. *et al.* (2006). *Aprendizaje de técnicas avanzadas de Programación Orientada a Objetos mediante programación de juegos.* XII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2006), Bilbao. Disponible en: http://www.robomind.net/en/docProgrammingStructures.htm

Rodríguez, D. (2008). *Enseñanza de programación orientada a objetos mediante el desarrollo de aplicaciones gráficas interactivas.* VIII Congreso TAEE, Universidad de Zaragoza, España.

Ryoo, J. *et al.* (2008). *Teaching Object-Oriented Software Engineering through Problem-Based Learning in the Context of Game Design.* Proceedings of the 21st Conference on Software Engineering Education and Training - Charleston, SC.

SeungBum, K. *et al.* (2006). *Smalltalk card game for learning Object-oriented thinking in an evolutionary way.* OOPSLA 2006, Portland, Oregon, USA.

Singh, R. & Singh, J. (2007). *Learning Computer Programming Using a Board Game - Case Study on C-Jump*. Master's thesis, Multimedia University.

The Lifelong Kindergarten Group, MIT Media Lab. (s.f.). Scratch. Disponible en: http://scratch.mit.edu/