Enfoque basado en gamificación para el aprendizaje de un lenguaje de programación

A game-based approach for learning a programming language

Gustavo Javier Astudillo, Silvia Gabriela Bast y Pedro Adolfo Willging
Grupo de Investigación y Desarrollo en Innovación Educativa,
Departamento de Matemática, Facultad Ciencias Exactas y Naturales,
Universidad Nacional de La Pampa. Argentina.
E-mail: astudillo@exactas.unlpam.edu.ar; silviabast@exactas.unlpam.edu.ar;
pedro@exactas.unlpam.edu.ar

Resumen

Muchos de nuestros estudiantes pertenecen a la generación de los "gamers". A partir del éxito de los juegos digitales como productos de entretenimiento, y en tanto formidables motivadores, se viene planteando desde hace un tiempo la posibilidad de aprovecharlos en el contexto educativo. Esta investigación busca identificar un conjunto de juegos digitales con el potencial de utilizarse para diseñar actividades didácticas, específicamente para el aprendizaje de algunos conceptos de la programación de computadoras, como son: algoritmos, variables y estructuras de control. Con base en los contenidos y la secuenciación de los aprendizajes se eligieron dos juegos que se incorporaron en actividades de un taller de introducción a la programación a fin de evaluar su utilidad. El taller fue diseñado sobre la plataforma Moodle, utilizando un enfoque de gamificación. Se presentan aquí una selección de juegos serios enfocados en la programación, los criterios para la elección de dos de ellos y su utilización en un taller para ingresantes, así como algunos resultados de la implementación de dicho taller.

Palabras clave: programación de computadoras; gamificación; juegos serios; estrategias educativas.

Abstract

Many of the students in our classrooms belong to the gamer generation. Because of the success of digital games as entertainment products as well as formidable motivators, the possibility of using them in educational settings is being contemplated from a while. In this research we try to identify a set of digital games with the potential to be used to design learning activities, specifically to learn computer programming concepts like: algorithms, variables, and control structures. Based on the contents and sequencing of learning, two games were chosen and incorporated into the activities of a workshop on introduction to programming in order to analyze their usefulness. The workshop was designed on the Moodle platform using a gamification approach. Here, we present a selection of serious games focused on computer language programming, the selection criteria of two of them and their use in a workshop for incoming students, as well as some of the results obtained from this experience.

Key words: computer programing; gamification; serious games; learning strategies.

Fecha de recepción: Marzo 2016 • Aceptado: Mayo 2016

ASTUDILLO, G.; BAST, S. Y WILLGING, P. (2015). Enfoque basado en gamificación para el aprendizaje de un lenguaje de programación. *Virtualidad, Educación y Ciencia*, 12 (7), pp. 125-142.

Introducción

Los niños y adolescentes actuales pertenecen a una generación que incluye los videojuegos en su vida cotidiana. Los avances tecnológicos propician este fenómeno, ya que posibilitan la incorporación de juegos a través de una variedad de dispositivos. En este contexto, desde el Grupo de Investigación y Desarrollo en Innovación Educativa (GrIDIE) y la cátedra Introducción a la Computación se re-diseñó e implementó un Taller de Introducción a la Programación (TIP). Los destinatarios del mismo son los ingresantes de las carreras Profesorado en Computación, Profesorado en Matemática y Licenciatura en Matemática (Facultad de Ciencias Exactas y Naturales, Universidad Nacional de La Pampa, Argentina).

Las actividades del TIP son de carácter optativo y se ponen a disposición de los ingresantes durante las cuatro semanas previas al inicio del cuatrimestre. Las primeras dos semanas se trabaja de forma virtual (plataforma Moodle) y las dos siguientes –en el marco de la Ambientación Universitaria– de forma presencial.

Los estudiantes de las tres carreras deben cursar, entre sus primeras materias, la asignatura Introducción a la Computación. Allí se abordan los conceptos iniciales de la programación de computadoras utilizando un lenguaje de programación de alto nivel, como son: la noción de algoritmos, variables y tipos de datos (simples y estructurados homogéneos), condiciones, estructuras de control y resolución de problemas con computadoras. El lenguaje utilizado en la asignatura para la aplicación de dichos conceptos es Pascal.

Con base en los diagnósticos realizados, se verifica que, en general, los estudiantes que ingresan a las carreras antes mencionadas, han tenido escaso o ningún contacto con la programación de computadoras en su paso por el nivel medio. Además, las actividades y conceptos asociados a la programación se presentan como un contenido complejo debido a que involucran: la resolución de problemas en un lenguaje que les es ajeno, el uso de estrategias y heurísticas, y la utilización de un entorno de desarrollo. Es así que, en busca de ofrecer a los ingresantes un andamiaje conceptual y procedimental para el abordaje de los contenidos de la asignatura, se diseñó el TIP.

La primera edición de este taller, implementada en 2009, se denominó "Curso de Resolución de Problemas". El mismo focalizaba en la temática homónima y el uso del software Scratch[1] para construir las primeras nociones de programación de computadoras. Con base en el análisis de los resultados y en busca de estrategias innovadoras que permitieran mejorar la participación y motivación de los ingresantes, se llevaron adelante re-diseños con distintos enfoques. En las ediciones subsiguientes el curso pasó de sólo contar con recursos tradicionales (como apuntes y trabajos prácticos) a combinar éstos con algunos de los que ofrece la Web 2.0. Posteriormente, pasó de un formato de curso a taller, y se puso el foco en el aprendizaje de las nociones básicas de programación y en problemas con solución algorítmica. En las ediciones de 2015 y 2016 se incorporaron los juegos serios y la gamificación.

1 Scratch (http://scratch.mit.edu) es un programa para crear animaciones y juegos multimedia utilizando programación basada en bloques que se encastran. Se describe con algo más de detalle en la sección Re-diseño del TIP.

La presente publicación se organiza en seis secciones. En la primer sección se presenta el contexto de investigación; en la segunda se desarrolla el marco teórico donde se exponen conceptos abordados en este trabajo. En la tercera sección, se enuncia el proceso de selección y se presenta el listado de juegos digitales localizados. En la cuarta sección, se dan detalles del re-diseño del Taller y seguidamente, en la quinta sección, se muestran algunos de los resultados obtenidos en la implementación del mismo. Finalmente, en la última sección, se exponen las conclusiones y trabajos futuros.

Marco teórico

Desde 2015 se buscó que las actividades del TIP tuvieran un enfoque innovador basado en gamificación y juegos serios, con el objetivo de favorecer la motivación y la concreción de las mismas. El diseño cuenta con un abordaje pedagógico mixto. El mismo se enfoca en lograr el buen aprendizaje y en generar Zona de Desarrollo Próximo (ZDP) a partir del uso de juegos digitales, pero también, hacer uso de la repetición mecánica de ejercicios para el aprendizaje de reglas e instrucciones.

Los juegos digitales

Se puede definir juego digital como "cualquier forma de software de entretenimiento basado en computadora […] usando cualquier plataforma electrónica como computadoras o consolas y que involucra a uno o varios jugadores en un ambiente físico o de red" (Frasca, 2001).

En un principio, las investigaciones respecto de la temática de juegos digitales se focalizaban en las consecuencias negativas del uso de éstos (Felicia, 2009; Connolly, et al., 2012). Actualmente, el énfasis está puesto en la motivación, el desafío, la obtención de competencias especiales, que hacen de los juegos digitales una herramienta con potencial para su aplicación en situaciones de aprendizaje.

Los juegos digitales pueden clasificarse según su objetivo en: para entretenimiento (como Mario Brothers[2]) y juegos serios (como Kokori[3]) diseñados para educar, entrenar o informar (Michael y Chen, 2005).

Varios autores (Lifelong-Learning Programme, 2009; McGonigal, 2011; Kapp, 2012) coinciden en que los juegos digitales deben contar con:

- Objetivos. Se trata de metas que deben alcanzar los jugadores para ganar el juego. Este elemento enfoca, orienta y motiva a los jugadores. Los juegos pueden contar con objetivos explícitos e implícitos. En el caso particular de los juegos serios, es el aprendizaje de un concepto, habilidad o competencia.

- Reglas. Este elemento establece las restricciones y define cómo los jugadores deben conseguir el objetivo. Para Kapp (2012) existen cuatro tipos de reglas: de funcionamiento, fundacionales, de comportamiento e instruccionales. Éstas últimas reflejan conceptos o competencias que, quien

2 Juego digital desarrollado por Nintendo (http://mario.nintendo.com/) en el año 1983. En él dos fontaneros, Mario y Luigi, recorren las alcantarillas de Nueva York y se enfrentan a extrañas criaturas a las cuales deben derrotar para ganar el juego.

3 Juego digital en el cual a partir del control de *nanobots* que recorren la célula se resuelven problemas utilizando conceptos de biología celular (http://www.kokori.cl/).

diseña el juego, desea que sean aprendidas por los jugadores para ser aplicadas o transferidas a otros contextos (Kapp, 2012).

- Desafíos o conflictos. Los juegos están basados en una historia, la cual debe contar con elementos de conflicto para incentivar y comprometer a los jugadores. Alrededor del conflicto se definen los otros elementos (Lifelong-Learning Programme, 2009).

- Competencia. Los juegos propician, en general, la competencia. Esta puede plantearse o bien contra el sistema, contra el propio jugador y/u otros oponentes (multi-jugador). La competencia se convierte en un recurso interesante, en tanto se estimula en el jugador la idea de auto-superación.

- Colaboración/cooperación. Este elemento está presente en la mayoría de los juegos digitales multi-jugador, los cuales promueven la colaboración entre jugadores en pos de conseguir un objetivo común, así como también, la generación de redes sociales, donde los jugadores interactúan entre ellos.

- Retroalimentación. La retroalimentación o feedback indica a los jugadores cuán cerca están de conseguir el objetivo. Puede tomar distintas formas dependiendo del juego: insignias[4], puntos, niveles, barra de progreso o simplemente el mensaje "game over". "Sirve como una promesa de que el objetivo es definitivamente alcanzable y les da motivación a los participantes" (McGonigal, 2011, p.21). El feedback informa el error pero no indica cómo corregirlo, lo que desafía al jugador a analizar lo que hizo (meta-cognición) e intentar mejorarlo, estrategia esta última, muy importante para la resolución de problemas.

- Re-playability. Los juegos deberían poder jugarse más de una vez sin perder el desafío o incrementándolo. El objetivo de esta característica es ayudar a la retención y ofrecer al jugador la oportunidad de desarrollar competencias y habilidades en un ambiente seguro, donde el jugador tiene permiso para "fallar". Según Kapp (2012) "permitir fallar al jugador con consecuencias mínimas alienta la exploración, la curiosidad y el descubrimiento" (p.48).

Otro concepto asociado a los juegos digitales es la gamificación (también denominada ludificación, o en inglés gamification), que hace referencia al "uso de elementos del diseño de juegos en contextos no lúdicos" (Deterding, et al., 2011, p.2). Esto es, aplicar metáforas que se corresponden con los juegos a otros contextos, como el educativo, en busca de mejorar la motivación y aumentar la participación. Como afirman Gallego, et al., (2014) "gamificar es plantear un proceso de cualquier índole como si fuera un juego. Los participantes son jugadores y como tales son el centro del juego […] y deben divertirse mientras consiguen los objetivos propios del proceso gamificado" (p. 2).

La gamificación, no está limitada al contexto tecnológico o digital (Groh, 2012). Ejemplo de esto es la intervención para transformar las escaleras de salida de una estación de subterráneo, en Estocolmo, en un piano y que los usuarios las tomaran como opción en lugar de la escalera mecánica[5]. Sin embargo, como afirma Kaap (2012), es imperativo que todos los elementos que definen a este

4 "Una insignia es un símbolo o indicador de un logro, cualidad, calidad o interés. Una "insignia digital" es el registro en línea de uno de esos logros". Las cuales, además, pueden ser socializadas en distintas comunidades digitales, conformando un portafolio (The Mozilla Foundation, et al., 2012).

5 Puede observarse la experiencia en el video "Piano stairs" de TheFunTheory.com. Disponible en: https://www.youtube.com/watch?v=2lXh2n0aPyw

enfoque estén presentes para hacer un uso genuino del mismo. Así, la propuesta debe estar basada en juegos (reglas, interactividad, retroalimentación, etc.), incluir niveles, recompensas, insignias y/o puntos. También son importantes: la estética y el pensamiento de juego (competición, exploración, la narración o relato de una historia); contar con un objetivo explícito, la motivación y el planteo de uno o varios problemas a resolver.

La aplicación de gamificación en el contexto educativo, más allá de la motivación, busca promover los aprendizajes a partir de la resolución de problemas a través de la interacción con el juego y/o con otros jugadores.

El enfoque pedagógico

Un aspecto importante en toda propuesta educativa es lograr buenos aprendizajes. Es decir un aprendizaje caracterizado por "demostrar (a) un cambio duradero (b) y transferible a nuevas situaciones (c) como consecuencia directa de la práctica realizada" (Pozo, 2008a, p. 162).

Sin embargo, deben darse las condiciones para que estos aprendizajes se produzcan. Una de ellas es, claramente, que los estudiantes tengan los conocimientos suficientes para acceder al nuevo material. Como afirman Pozo y Pérez Echeverría (2009) "nuestro recuerdo y aprendizaje serán el producto de la interacción entre [los] materiales y los conocimientos previos que activamos" (p. 34). Otra de las condiciones es la creación de actividades que, basadas en conocimientos previos, propicien los nuevos aprendizajes, es decir "para que los [estudiantes] comprendan no basta con presentarles la información […] es preciso diseñar actividades o tareas que hagan más probable esa actividad cognitiva…" (Pozo y Pérez Echeverría, 2009, p. 33). También se debe tener en cuenta que las actividades planteadas resulten un desafío (alcanzable) para los estudiantes. "Las tareas elegidas [deben ser] lo suficientemente abiertas como para suponer un reto a los estudiantes, pero también lo bastante cerradas como para tener conocimientos que les permitan representarse la tarea y les permitan avanzar en el proceso de solución" (Pozo y Pérez Echeverría, 2009, p. 50). Además, debe plantearse cómo los conceptos serán presentados a los estudiantes. En este sentido, fue adoptado para el TIP, la diferenciación progresiva. Este concepto es tomado de la teoría de Ausubel y plantea que "el contenido de instrucción debe organizarse en unidades secuenciadas, que vayan de las ideas más generales a las más específicas. El conjunto de la información debe presentarse al inicio de la instrucción y luego diferenciarse progresivamente en cuanto a detalles y especificidad" (Claux el al., 2001, p.48).

A lo anterior, es posible sumarle la idea de Vygotsky (1979) de que la interacción con alguien más experto permite alcanzar más o mejores aprendizajes. Esto es denominado por el autor, la Zona de Desarrollo Próximo (ZDP). El concepto puede definirse como "la distancia entre el nivel real de desarrollo, determinado por la capacidad real de resolver independientemente un problema, y el nivel de desarrollo potencial determinado a través de la resolución de un problema bajo la guía de un adulto o en colaboración de un compañero más capaz" (p. 131). Como afirman Baquero y Limón Luque (2000) es el propio Vygotsky, quien advierte que los juegos generan ZDP. Es decir, además de expertos y compañeros que actúen como guía, también los juegos actuarían como soporte permitiendo alcanzar nuevos conocimientos con mayor eficacia que si el estudiante emprendiera este desafío en soledad.

Tomando en cuenta los conceptos anteriores fue re-diseñado el TIP. Así, con base en que los ingresantes cuentan con conocimientos sobre el uso de juegos digitales, se utilizan juegos serios para propiciar el aprendizaje de la programación de computadoras. Para el diseño de la secuencia de aprendizaje se tuvieron en cuenta los conceptos y habilidades aprehendidas en actividades previas para el abordaje de las subsiguientes, esto fue implementado a través de los niveles del taller. Cada nivel agrega también, un nivel de especificidad conceptual. Por último, la utilización de juegos serios para aprender a programar, podría operar sobre la ZDP de los estudiantes permitiendo alcanzar con mayor efectividad las nociones de programación involucradas en el taller (algoritmo, variables y estructuras de control).

Proceso de selección de juegos digitales

Como parte del proceso de investigación, se indagó sobre fortalezas y debilidades de los juegos digitales como recurso didáctico y la posibilidad de aplicar estos conceptos al diseño de actividades educativas concretas, ésto dio lugar a la definición de una metodología de búsqueda y selección de juegos digitales. En este caso, aplicada a la localización de juegos serios que permitieran el aprendizaje de algunos conceptos asociados a la programación de computadoras para ser incorporados al TIP (algoritmo, variables y estructuras de control).

La metodología permitió obtener un conjunto de juegos serios que tendrían el potencial de utilizarse para diseñar actividades para el aprendizaje de programación de computadoras.

El proceso arrojó una importante cantidad de resultados. Sin embargo, fue necesario realizar una selección con vistas a: (i) la consecución de los objetivos de aprendizaje buscados, (ii) una apropiada secuenciación de los contenidos y de los juegos elegidos, (iii) así como también que pudieran incluirse en la secuencia de aprendizaje propuesta para el TIP.

Resultados. Juegos digitales para aprender a programar

Para la búsqueda de los juegos digitales se llevó adelante una secuencia metodológica basada en Kitchenham (2004). Para esto se definieron un conjunto de preguntas de investigación que tuvieron la función de guiar la búsqueda y generar un conjunto de criterios de inclusión y exclusión. Las preguntas:

a. ¿El software cuenta con la mayoría de los elementos de un juego digital?

b. ¿El juego digital puede ser accedido libremente desde la Web?

c. ¿El juego digital es un juego serio?

d. ¿Aborda la temática de programación de computadoras?

Una vez obtenido el conjunto de juegos digitales que cumplían con los criterios de inclusión, se realizó una nueva selección con vistas a la aplicación de los mismos al TIP. Hecha dicha selección se procedió al análisis de los mismos. Se presentan a continuación los resultados obtenidos en base a la búsqueda realizada.

La búsqueda y la posterior aplicación de los criterios de inclusión/exclusión permitió localizar nueve juegos serios que se enfocan en el aprendizaje de distintos aspectos de la programación de computadoras: desde las nociones básicas enfocadas de manera abstracta, hasta el tipeo de las líneas de código. A continuación se listan y describen brevemente cada uno de ellos:

- Pilas Engine (http://pilas-engine.com.ar/). Es una herramienta que permite crear juegos digitales mientras se aprenden conceptos de programación tales como estructuras de control y uso de variables. Está diseñado con el objetivo de ofrecer ayuda a los principiantes para crear juegos digitales programando en lenguaje Python[6].

- CheckiO (https://www.checkio.org). Este juego propone programar participando de desafíos y desarrollando algoritmos cada vez más complejos. Se trata de avanzar entre distintas estaciones, en las que se van planteando problemas de programación. Los mismos deben resolverse escribiendo textualmente las instrucciones en lenguaje Python. Se hace uso de variables, estructuras de datos y estructuras de control.

- Code.org (http://studio.code.org). Se trata de un juego para el aprendizaje de las nociones básicas de programación mediante lecciones guiadas. Su estructura permite abstraer a los jugadores de la sintaxis propia de los lenguajes. Las instrucciones se presentan de manera icónico-textual en forma de bloques lógicos que pueden encastrarse para realizan acciones específicas. Se introducen de forma sencilla las estructuras de control.

- CodeCombat (http://codecombat.com/). Este juego enseña paso a paso a programar. Permite elegir un reto, un personaje y el lenguaje en el que se van a resolver los desafíos. La opción por defecto es Python, pero permite también lenguajes como: Javascript[7], Coffe Script[8], entre otros. Debe guiarse a un personaje, al que se le pueden adjudicar un conjunto de objetos (utensilios, armas, etc.), utilizando sentencias de programación de forma textual para pasar los distintos niveles.

- Rails for Zombies (http://railsforzombies.org/). El objetivo del juego está centrado en el aprendizaje de las estructuras de control de programación en el lenguaje Ruby on Rails[9]. Se proponen diferentes desafíos distribuidos en cinco niveles. Cada nivel comienza con una lección en video que establece el objetivo. En cada reto se presenta un ejercicio sencillo que se resuelve a partir de la introducción de líneas código. El jugador debe ejecutar el código y así, si es correcto, pasar de nivel.

- Code-Racer (http://www.coderacer.com/). El objetivo del juego es tipear un programa en lenguaje C++[10] en el menor tiempo posible, compitiendo contra la máquina u otros programadores. Se trata de una aplicación multi-jugador que permite desarrollar las habilidades de codificación y velocidad de tipeo. Hay un tiempo de 5 minutos para ingresar el código y se

6 Python (http://www.python.org) es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

7 JavaScript es un lenguaje de programación interpretado para la web.

8 Cofee Script (http://coffeescript.org/) es un lenguaje de programación que se compila JavaScript.

9 Ruby (https://www.ruby-lang.org/es/) es un lenguaje de programación que incorpora tanto la programación funcional como la imperativa. Ruby on Rails (http://www.rubyonrails.org.es/) es un framework, desarrollado en Ruby, para construir aplicaciones web que acceden a bases de datos.

10 C++ es un lenguaje de programación diseñado a mediados de los 80' por Bjarne Stroustrup con el objetivo de extender el lenguaje C para la manipulación de objetos.

genera un ranking en función del valor obtenido en LPM (Lines Per Minute).

- Robocode (http://robocode.sourceforge.net/). Se trata de un simulador de combates entre tanques. Es posible aprender las estructuras de control del lenguaje Java programando cada tanque a partir de la definición de su movimiento, los sensores que detectan otros tanques y los disparos del cañón.
- Lightbot (http://lightbot.com/hocflash.html). Consiste en darle instrucciones a un robot para que recorra un escenario (damero 3D) y alcance su objetivo (encender las celdas azules). Cada orden, representada de forma icónica, se traduce en una acción que debe realizar el robot. Aborda, de manera gráfica e intuitiva, distintos conceptos de la programación: secuencia, estructuras de control, modularidad y recursión.
- Kidsruby (http://kidsruby.com/). Es un juego digital de escritorio que permite a niños/as aprender a programar en lenguaje Ruby. Cuenta con una introducción al concepto de algoritmo, una explicación del uso del lenguaje donde se detallan las distintas sentencias (el jugador puede probarlas) y tres juegos donde se debe programar para completar los desafíos. Se trata de una aplicación libre y de código abierto. Tanto en las explicaciones, como en la programación se usa únicamente el recurso textual.

Selección de los juegos digitales para el TIP

Como afirma Pozo (2008b) "La naturaleza dinámica de los procesos de aprendizaje tiene [...] implicaciones para el diseño de situaciones más eficaces. Una de esas implicaciones tiene que ver con la importancia del orden temporal en las actividades de aprendizaje" (p. 165). En la selección se tuvo muy en cuenta que la secuenciación fuera posible y que no se viera el taller como una secuencia de juegos inconexos (ver tabla 1). También se dio especial importancia a los lenguajes propuestos por cada uno de los juegos, teniendo en cuenta que en la cursada de la asignatura se hace uso del lenguaje Pascal.

Luego de analizar cada uno de los juegos detallados en la sección anterior, se seleccionaron dos para su aplicación en el TIP. Los juegos digitales seleccionados fueron: Lightbot 2.0 y Code.Org.

Tabla 1. Criterios de secuenciación de los juegos digitales y Scratch

Lightbot	Code.org	Scratch
Lenguaje restringido e icónico. Con tutoría visual del sistema para el uso y la resolución de los problemas/ejercicios.	Lenguaje restringido, icónico/textual. Se programa por encastre. Con tutoría visual del sistema para el uso y la resolución de los problemas/ejercicios (por cada nivel que presenta un nuevo tipo de sentencia/instrucción).	Lenguaje no restringido, icónico/textual. Se programa por encastre. Sin tutoría.

Lightbot 2.0 posibilita introducir la noción de algoritmo[11] a través de la secuenciación de instrucciones icónicas (figura 1 en la siguiente página). No son necesarios conocimientos previos de programación para su utilización, permite el ensayo de una solución y no penaliza el error. También es importante que las instrucciones quedan visibles y son iluminadas en tiempo de ejecución. Así

11 Según la Real Academia Española, algoritmo es el "conjunto ordenado y finito de operaciones que permite hallar la solución de un problema" (http://lema.rae.es/drae/?val=algoritmo).

puede seguirse la ejecución paso a paso, detectar y, posteriormente, corregir errores en la secuencia. Se introduce de forma intuitiva el concepto de algoritmo y el de mejora iterativa del mismo.

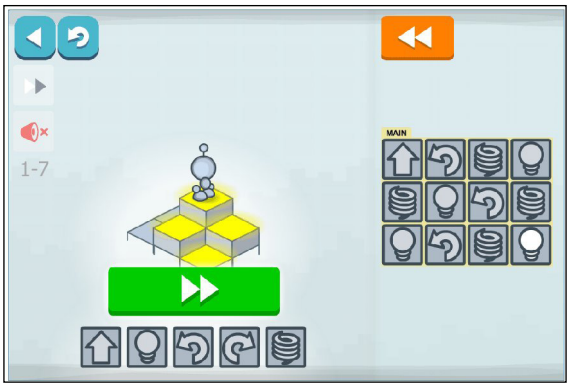

Figura 1. Pantalla de Lightbot. Solución del desafío 7-Nivel 1.

Code.Org (figura 2), por su parte, presenta problemas sencillos y un grupo mínimo de bloques lógicos que permiten resolverlos. Inicia con noción de secuencia, luego incorpora las estructuras de control repetitivas y selectivas. Dado que los bloques y la forma de programar son similares a Scratch, se utiliza como paso previo de este lenguaje.

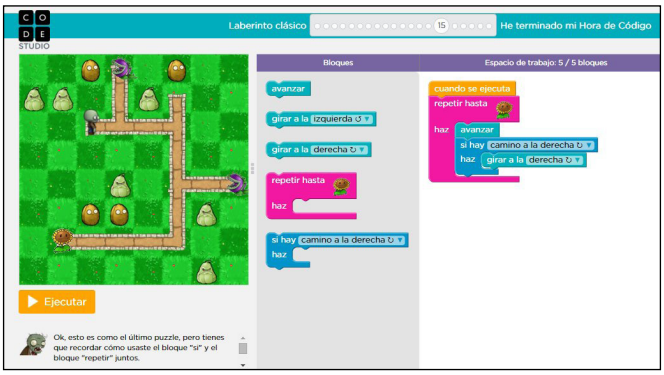

Figura 2: Ambiente del juego de Code.org. nivel 15 “Laberinto clásico”.

Re-diseño del TIP

El TIP fue re-diseñado utilizando juegos serios y con un enfoque lúdico y de experimentación. El mismo consta de cuatro niveles. En el primer nivel (Lightbot 2.0), el objetivo es que los estudiantes construyan el concepto de algoritmo e instrucción. En el segundo, con Code.org, una primera noción de estructuras de control. En el nivel 3 y 4 con Scratch la resolución de situaciones problemáticas a través de la programación. En la tabla 2, se presenta un resumen de conceptos abordados y los recursos utilizados en cada nivel.

Tabla 2. Resumen de actividades y recursos abordados en el TIP

Nivel	Recursos	Conceptos abordados
1	Lightbot (sólo el nivel 1 del juego)	Noción de algoritmo
2	Code.org (hora de código/Laberinto clásico)	Noción de estructuras de control
3	Scratch/Pasos de Polya	Noción de variable, estructuras de control y resolución de problemas
4	Scratch	Noción de variable, estructuras de control y resolución de problemas

A continuación se detallan el diseño por niveles y las actividades en cada uno.

Aplicando conceptos de gamificación

Las actividades del curso fueron estructuradas por niveles y se hacen visibles una vez que los participantes o jugadores completan el nivel anterior. Esto también ocurre con las actividades o recursos intra-nivel, que se hacen visibles al acceder a los materiales o al completar las actividades. La organización por niveles se implementó utilizando la opción "Restringir disponibilidad" y activando el "Progreso de estudiante" en Moodle (versión 2.4). De acuerdo a la propuesta, se avanza de nivel cuando el estudiante obtiene una calificación entre 30 y 100 por ciento en la evaluación propuesta para completar el nivel anterior.

La opción de "Condición de finalización de actividad" también se utilizó para las recompensas a los jugadores a través de etiquetas que contienen tanto puntaje, como información adicional.

Nivel 1. Lightbot

En el nivel 1 se utiliza Lightbot para que los estudiantes construyan la noción de algoritmo e instrucciones. A este nivel del taller se accede sin restricciones.

Actividades:

1. Foro de presentación y Encuesta inicial. Actividades que deben hacer (obligatoriamente) para acceder a los primeros contenidos del Taller. El foro propicia la ambientación a Moodle y la interacción. La encuesta, se presenta como un diagnóstico inicial.

2. Video de presentación de Lightbot. Deben ver el video, que explica cómo acceder y el funcionamiento del juego. Esto habilita el enlace al juego.

3. Los participantes deben completar el primer nivel de Lightbot. Se asume que los estudiantes cuentan dentro de sus ideas previas con el conocimiento de los juegos y juegos digitales en particular (reglas, niveles, puntaje, retroalimentación, entre otros).

4. Con la presentación (ítem 2) también queda visible un documento con una equivalencia entre instrucciones icónicas de Lightbot y su forma textual (ver figura 3 en la siguiente página), se busca con esto propiciar la transferencia de los aprendizajes. "La transferencia es uno de los rasgos centrales de un aprendizaje eficaz y satisfactorio [...]. Sin capacidad de transferir lo aprendido a nuevos contextos, lo aprendido es muy poco eficaz" (Pozo, 2008b, p. 167). En nuestro caso, dado que los lenguajes de computación están basados, principalmente, en instrucciones textuales, la idea de que comiencen a utilizarlas en este formato es importante.

Figura 3. Equivalencia de las instrucciones de lightbot en formato textual.

5. Evaluación para pasar de nivel. Para poder alcanzar el nivel 2, los estudiantes deben aprobar una evaluación donde se plantean desafíos con escenarios de Lightbot. Algunos de éstos conocidos y otros desconocidos (ver figuras 4 y 5). La resolución deben hacerla con instrucciones textuales. El examen tiene preguntas que se auto-corrigen y otras evaluadas por docentes. Dado que distintos algoritmos pueden solucionar el mismo problema, la pertinencia y eficiencia del algoritmo planteado por cada estudiante es analizada y el docente le hace devolución al respecto. Además de acceder al Nivel 2 ganan puntos e información. En este caso se presenta de la definición de algoritmo e instrucción.

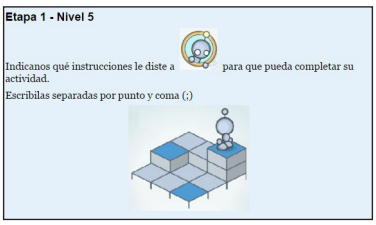

Figura 4. Equivalencia de las instrucciones de lightbot en formato textual.

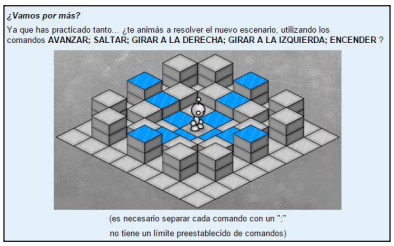

Figura 5. Equivalencia de las instrucciones de lightbot en formato textual.

Los desafíos planteados en Lightbot pueden verse inicialmente como problemas, por tratarse de situaciones desconocidas para los estudiantes. Sin embargo, al avanzar por los niveles el uso de las instrucciones se vuelve una acción rutinaria ya que mantienen su lógica. Pero, como afirma Pozo (2008a) las "teorías asociativas permiten […] el aprendizaje implícito de reglas y procedimientos, [...] cuanto más repetitivas y rutinarias sean [las] condiciones más eficaz se mostrará un aprendizaje asociativo" (pp. 145-146). Se debe tener en cuenta que el aprendizaje de la programación implica el aprendizaje de un conjunto de reglas o instrucciones arbitrarias, por lo que se cree que el enfoque de las

teorías asociativas es apropiado en este caso. Asimismo, se cree que la noción de algoritmo (secuencia de pasos ordenada y finita), así como la de instrucción podrían obtenerse como consecuencia directa de la práctica con Lightbot.

Nivel 2. Code.org

En este nivel se propone a los estudiantes jugar en Code.org. Este juego presenta varias propuestas para aprender a programar. Para el TIP se optó por "La hora de código" y dentro de ella "Laberinto clásico" (https://studio.code.org/hoc/1).

Dado que la metáfora utilizada por el juego es similar a la de Lightbot, los conocimientos adquiridos actúan como base para jugar en Code.org.

Al igual que Lightbot, en Code.org los desafíos de cada nivel son inicialmente problemas, sin embargo, al pasar los niveles se tornan tareas más rutinarias. Se abordan, en sus 20 niveles, los conceptos de secuencia, selección y repetición (en lugar de sólo secuencia, como en el nivel 1 de Lightbot[12]). En los niveles 1 al 5 se presenta primero la idea de secuencia. Luego del nivel 6 al 9 la de estructuras repetitivas fijas (se conoce el número de iteraciones), para luego introducir las repetitivas con condiciones (sin entrar en la complejidad de la sintaxis de una condición: operadores relacionales y lógicos) en los niveles 10 al 13. Finalmente, se introducen las estructuras de selección, primero las que bifurcan por un sola rama (niveles 14 al 18) y luego las de dos ramas (niveles 19 y 20). Cabe aclarar que una vez presentada una estructura de control, la misma es utilizada para resolver problemas en los niveles subsiguientes. La noción de estructuras de control (selectivas y repetitivas) podría obtenerse como consecuencia directa de la práctica con Code.org. Esta forma de presentar los contenidos puede lograr en los estudiantes la diferenciación progresiva de los tipos de instrucciones, y permite la apropiación de los conceptos de secuencia, repetición y selección. El uso reiterado y rutinario de las instrucciones hace que las internalicen y se apropien de ellas.

Al nivel "Nivel 2. Code.org" en Moodle, los estudiantes acceden aprobando la evaluación descrita en el ítem 5 de la sub-sección anterior. Este nivel incluye las siguientes actividades y recursos:

1. Presentación de Code.org. Video explicativo sobre cómo jugar.
2. Lo anterior habilita la actividad y el enlace a Code.org.
3. Publicación del certificado que entrega Code.org. Este juego entrega un certificado. El mismo debe ser publicado para acreditar el paso por el juego. Esto da acceso a la siguiente actividad.
4. Actividad evaluativa, llevar al zombi a la biblioteca. Esta evaluación permite pasar de nivel, pero además plantea un problema real. Al ser ingresantes no conocen la ubicación de la biblioteca central de la Universidad y se les pide que lleven a un personaje hacia ella. Se les ofrecen instrucciones textuales y un mapa real de la ciudad.

Nivel 3 y 4. Scratch

Scratch es un lenguaje de programación para crear historias interactivas, animaciones, juegos, música, etc. Está desarrollado por el Lifelong Kindergarten en el MIT Media Lab (Resnick, et al.,

12 Cabe aclarar que Lightbot propone el abordaje de más conceptos, incluso algunos que exceden la propuesta de la asignatura Introducción a la Computación (objeto del taller).

2009). La programación en Scratch se basa en un conjunto de instrucciones icónico/textuales que se combinan o encastran para crear programas (ver figura 6). Estos bloques sólo se acoplan si la sintaxis es correcta, lo que libera al estudiante de la complejidad de la misma y le permite concentrarse en la solución del problema.

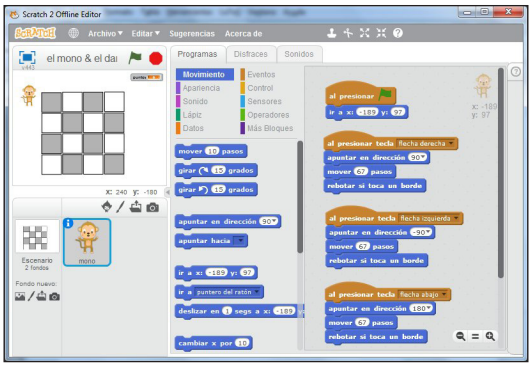

Figura 6. Ambiente de Sratch 2.

El ambiente en el que se programa no arroja errores, por lo que los usuarios no son penalizados, lo que evita la frustración inicial al no poder ejecutar el programa (errores de compilación) y propicia la experimentación. Como afirma Pozo (2008b) "puede decirse que si los aprendices se entrenan sólo en completar ejercicios (tareas cerradas o rutinarias para las que han aprendido ya una solución específica) difícilmente aprenderán a resolver problemas (tareas más abiertas para las que hay que buscar vías de solución) [...]. Sólo entrenándose en la solución de problemas se aprende a resolver problemas" (p. 168). En este sentido, tanto Lightbot como Code.org, presentan un conjunto desafíos (inicialmente problemas) que al avanzar por los niveles se vuelven ejercicios rutinarios. Más aún, al volver a jugar los desafíos serían exactamente los mismos. "Las diferencias entre ejercicios y problemas también dependen […] de los conocimientos previos de quien tiene que resolver el problema. [...] El aumento de la experiencia y del conocimiento influyen en que buena parte de los problemas y toma de decisiones vayan progresivamente tecnificándose y convirtiéndose en ejercicios" (Pozo y Pérez Echeverría, 2009, p. 48). En este sentido, la incorporación de Scratch permite avanzar sobre la resolución de problemas. Este software ofrece el ambiente para programar, pero los problemas o desafíos no son propuestos desde la aplicación por lo que deben ser planteados a los estudiantes desde fuera, como una actividad.

Para acceder al nivel 3, los estudiantes deben aprobar la evaluación descrita en el ítem 4 de la sub-sección anterior. Cabe aclarar que este nivel (y el que le sigue) coinciden con la etapa presencial del taller. Las clases en las que se trabaja con Scratch tienen un problema como hilo conductor, el cual se aborda de forma iterativa, de forma de conseguir en cada iteración una nueva aproximación a la solución final.

El problema planteado, en esta edición del TIP, es realizar un juego en el que se le da a un mono, una secuencia de instrucciones (como en Lightbot) para que logre comer todas las bananas disponibles en un damero. Debe haber botones para: agregar las instrucciones (arriba, abajo, derecha e izquierda), borrar todo y ejecutar las instrucciones (ver figura 7 en la siguiente página).

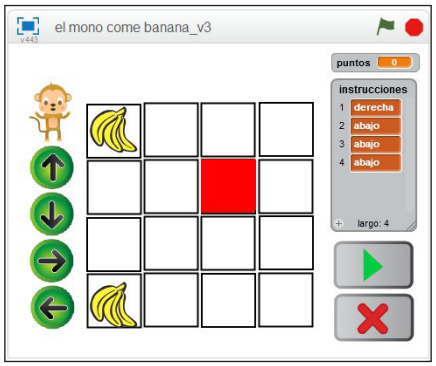

Figura 7. Posible escenario/solución para el juego "El mono come banana".

El nivel 3 incluye las siguientes actividades y recursos:

1. Polya vs Desarrollo de sistemas. Video donde se explican y comparan los cuatro pasos básicos del desarrollo de software con los cuatro pasos propuestos por George Polya para la resolución de problemas.

2. El gato y el damero. Se presenta el problema de mover un gato (objeto) a través de un damero (escenario) con las flechas del cursor. Este problema involucra el uso de escenarios, eventos e instrucciones. Dado que "una estrategia de enseñanza será más eficaz [...] cuando incluya actividades que ayuden o guíen a los [estudiantes] en la activación de esas ideas…" (Pozo y Pérez Echeverría, 2009, p. 38), se opta por resolver el problema en clase con la guía del docente. El objetivo es mostrar estrategias y heurísticas para resolver problemas, discutir las posibles soluciones y mostrar el razonamiento experto. Como afirman Pozo y Pérez Echeverría (2009) "al plantear problemas en el aula y utilizarlos para aprender se produce un doble efecto, por un lado se aprende y se enseña a tomar decisiones de manera reflexiva, por otro se ayuda a la creación y al desarrollo de heurísticos que facilitan la resolución de esos mismos problemas" (p. 49). Los estudiantes deben publicar la solución para acceder a la próxima actividad.

3. Trabajo práctico 1. Se proponen tres problemas sencillos para resolver con Scratch. Los estudiantes no tendrán aquí la guía del docente, aunque pueden usar el foro para realizar consultas. Se busca que los estudiantes puedan aplicar lo visto en clase en otras situaciones. Pero también, que experimenten, debido a que ya conocen cómo funciona el programa y la mecánica de programación.

4. El mono come banana. Se presenta el problema de hacer que un mono "coma" unas bananas que se encuentran dispersas al azar en un damero. Cada vez que come gana 100 puntos. Con base en la actividad anterior (estrategia para resolver problemas: basarse en uno similar ya resuelto), se cambia el personaje que por defecto ofrece Scratch por un mono (ejercicio). Este problema involucra el uso de varios objetos (bananas), las instrucciones del bloque control (selección y repetición) y se introducen las variables y operadores para ubicar las bananas al azar y para calcular el puntaje. El problema se resuelve en clase con la guía del docente. Los estudiantes deben publicar una solución a la que se le agrega un desafío adicional (problema): un cuadro rojo en el damero que si es tocado por el mono, éste vuelve al inicio y se le descuentan puntos. Se busca que al realizar esta actividad puedan transferir lo aprendido, ya que la solución es similar a lo trabajado en clase. La entrega es evaluada por un docente, quien le hace una devolución

(puede volver a entregar las veces que sea necesario). Si el programa es correcto acceden al próximo nivel.

5. Trabajo práctico 2. Se proponen dos problemas sencillos para resolver con Scratch. Se busca que los estudiantes resuelvan otros problemas con las instrucciones recién aprendidas, aplicándolas a nuevas situaciones y que experimenten con las mismas.

El nivel 4 incluye las siguientes actividades y recursos:

1. El mono come banana (versión 2). Se agregan, al problema ya planteado, nuevos desafíos: el mono tiene 3 vidas y las pierde de a una al tocar el cuadrado rojo, el jugador puede ganar (comer todas las bananas) o perder el juego (quedarse sin vidas), y hay en el damero bananas en mal estado o verdes que al comerlas hacen que se pierdan puntos. Dado que se trata de desafíos que pueden ser resueltos a partir de los conceptos y bloques ya utilizados, se hace el planteo del problema y se deja que cada estudiante (o grupo) diseñe una solución. Un aspecto que influye en que "una actividad esté más cerca de un ejercicio o de un problema es quién ejerce el control de la tarea. […] En el caso de las actividades de aprendizaje, serán simples ejercicios cuando el profesor tome [las] decisiones por los alumnos..." (Pozo y Pérez Echeverría, 2009, p. 49). Al finalizar la clase se socializan las soluciones. Los estudiantes deben publicar su solución, un docente las evalúa y hace una devolución. Deben aprobar la tarea de programación para acceder a la próxima actividad del nivel.

2. Trabajo práctico 3. Se propone a los estudiantes la programación con Scratch de un juego sencillo.

3. El mono come banana (versión 3). Aquí se retoma el planteo inicial del problema. Se incorpora el manejo de variables estructuradas (listas) para almacenar las instrucciones. Se cambia el comportamiento del mono para que siga las instrucciones (recorrer la lista). Se incorpora el concepto de programar un botón (objeto conocido por los estudiantes por el uso de otras aplicaciones) con el evento Al hacer clic. Se retoma el uso de estructuras de control repetitivas y selectivas. Se incorpora el uso de estructura repetitiva de tipo fijo (repetir n). Este tipo de estructura fue utilizada en los niveles 6-8 de Code.org. El problema se resuelve en clase con la guía del docente. Los estudiantes deben publicar la solución para finalizar el taller y ganar el juego (son evaluados como en los otros casos).

4. Trabajo práctico 4. Se propone la realización de un juego similar al PAC-MAN[13].

Al finalizar el TIP se solicita a los participantes que contesten una evaluación final donde se indaga sobre los conocimientos abordados en el Taller y que se retomarán al inicio de la asignatura Introducción a la Computación. Esta actividad estuvo únicamente disponible para los estudiantes que completaron exitosamente, al menos, el nivel 3.

Cabe destacar que en la edición 2015 del taller se re-diseñó la estética de los materiales, lo que se continuó en 2016. Además de ofrecer materiales atractivos, se buscó la integración del TIP con la asignatura a través de una estética en común, donde los materiales y actividades de cada etapa (1. Virtual, 2. Presencial y 3. Cursada) se distinguen por un color particular que la identifica.

13 Videojuego arcade creado por Toru Iwatani. Para más detalles visitar http://pacman.com/en/

Resultados

La utilización de juegos digitales, juegos serios y gamificación en el TIP se implementó durante las ediciones 2015 y 2016.

En ambas ediciones se pudo observar un mejoramiento en la participación de los ingresantes resolviendo las actividades propuestas, en comparación con ediciones anteriores. Cabe recordar que el TIP es optativo y la participación (o no) en él no es requerimiento para cursar Introducción a la Computación.

Durante 2015 se observaron buenos resultados en la evaluación final. La misma, fue realizada por 25 de los 57 participantes y el 70% tuvieron un promedio por encima de 7 en la calificación final. Esto muestra un dominio aceptable de los conceptos. Debe tomarse en cuenta que: de los 57 inscriptos, 44 contestaron la encuesta inicial, y de ellos el 80% no habían utilizado nunca un lenguaje de programación.

Respecto a la cursada, el 79% de los estudiantes que aprobaron el primer parcial habían completado las actividades del TIP.

En 2016, hubo 61 ingresantes en condiciones de hacer el TIP. De ellos 11 no accedieron nunca y 4 accedieron pero no hicieron ninguna actividad. Esto deja un total de 46 participantes. De este total, 45 completaron el Nivel 1, 38 el Nivel 2, 28 el Nivel 3 y 13 el Nivel 4. Los 28 que completaron el Nivel 3 tenían acceso a la evaluación final. De ellos, sólo 17 la completaron. En promedio, quienes realizaron la evaluación, obtuvieron una calificación de nueve. Como en el año anterior, del total de participantes, el 80% (37 de los 46) indicaron que no conocían ningún lenguaje de programación.

Conclusiones y trabajos futuros

En este trabajo se han presentado, por una parte, una lista de juegos serios que pueden ser utilizados para el diseño de actividades educativas que favorezcan el aprendizaje de las nociones básicas de programación de computadoras. Por otra, una secuenciación de actividades basada en gamificación y juegos serios.

Esto da cuenta de la posibilidad de la utilización de una estrategia basada en juegos digitales para el aprendizaje de la programación.

Además, con base en la secuenciación propuesta para el re-diseño del TIP, se presentaron los resultados obtenidos hasta el momento, en las dos ediciones del Taller en las que se aplicó el enfoque de juegos y gamificación. Estos resultados muestran que es posible realizar una secuenciación de contenidos en base a la utilización de juegos serios y gamificación para el aprendizaje de la programación de computadoras, así como también, la forma de implementarlo sobre la plataforma Moodle. Los juegos serios elegidos cumplieron con los objetivos propuestos. Fueron recursos adecuados para introducir los conceptos de algoritmo, variable y estructuras de control, así como para motivar y resolver problemas.

Si bien los resultados expuestos en la sección anterior son preliminares y requieren de un análisis más minucioso, indicarían, junto con las encuestas de satisfacción que se le hicieron a los estudiantes,

que tanto la propuesta metodológica, como la utilización de juegos favorece el desempeño de los estudiantes en la cursada.

Además, dado que nuestros estudiantes pertenecen en su mayoría a carreras de formación docente, el haberlos expuesto tempranamente a actividades mediadas por TIC, particularmente con el uso de juegos digitales y gamificación, representa un aspecto valioso en sí mismo para su formación profesional.

Como trabajos futuros se plantean los desafíos de: definir estrategias que permitan incrementar el número de ingresantes que completan el TIP, replicar el enfoque al interior de la cursada de la asignatura para lograr una mayor integración y, dada la actualización de la plataforma Moodle donde se desarrolla el curso, la incorporación de insignias como forma de recompensa en el TIP; así como también, revisar otras opciones de juegos serios que permitan una mejor aproximación al uso de un lenguaje de alto nivel donde las instrucciones se presentan de forma textual.

Referencias bibliográficas

BAQUERO, R., y LIMÓN LUQUE, M. (2000). *Introducción a la psicología del aprendizaje escolar*. Bernal: Universidad Nacional de Quilmes.

CLAUX, M. L., KANASHIRO, Y. y YOUNG, A. M. (2001). *Modelos psicológicos de la instrucción*. Lima: Ministerio de Educación del Perú.

CONNOLLY, T.; BOYLE, E.; MACARTHUR, E.; HAINEY, T. y BOYLE, J. (2012). A systematic literature review of empirical evidence on computer games and serious games. *Computers y Education*, 59 (2), pp. 661 - 686.

DETERDING, S.; KHALED, R.; NACKE, L. y DIXON, D. (2011). Gamification: Toward a definition. En *CHI 2011 Gamification Workshop Proceedings*. Vancouver, BC, Canada.

FELICIA, P. (2009). *Videojuegos en el aula: manual para docentes*. Bruselas: European Schoolnet.

FRASCA, G. (2001). *Videogames of the oppressed: Videogames as a means for critical thinking and debate* (Tesis). Georgia: Georgia Institute of Technology.

GALLEGO, F.; MOLINA, R. y FARAÓN, L. (2014). Gamificar una propuesta docente. Diseñando experiencias positivas de aprendizaje. Conferencia presentada en *XX Jornadas sobre la enseñanza universitaria de la informática*, Oviedo, España. En línea: http://hdl.handle.net/10045/39195

GROH, F. (2012). Gamification: State of the Art Definition and Utilization. En N. Asaj, B. Könings, M. Poguntke, F. Schaub, B. Wiedersheim, y M. Weber (Eds.) *4th Seminar on Research Trends in Media Informatics*, Alemania: Ulm University. pp. 39-46.

KAPP, K. (2012). *The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education*. San Francisco: John Wiley y Sons.

LIFELONG-LEARNING PROGRAMME (2009). *Production of Creative Game-Based Learning Scenarios - A Handbook for Teachers. ProActive: Fostering teachers creativity through Game-Based Learning*. En línea: http://www.ub.edu/euelearning/proactive/documents/handbook_creative_gbl.pdf.

Mc GONIGAL, J. (2011). *Reality Is Broken: Why Games Make Us Better and How They Can Transform the World*. New York: The Penguin Press.

MICHAEL, D. y CHEN, S. (2005). *Serious Games: Games That Educate, Train, and Inform*. Muska y Lipman/ Premier-Trade.

POZO, J. (2008a). Capítulo 3. Las teorías del aprendizaje: la integración entre diferentes niveles y sistemas de aprendizaje, en *Aprendices y maestros: la psicología cognitiva del aprendizaje*. Madrid: Alianza. pp. 121-148.

POZO, J. (2008b). Capítulo 4. Los rasgos de un buen aprendizaje, en *Aprendices y maestros: la psicología cognitiva del aprendizaje*. Madrid: Alianza. pp. 121-148.

POZO, J., y PÉREZ ECHEVERRÍA, M. (2009). Capítulo 2. Aprender para comprender y resolver problemas. En J. I. Pozo y M. del P. Pérez Echeverría (Eds.), *Psicología del Aprendizaje Universitario: la Formación en Competencias*. Madrid: Ediciones Morata. pp. 31-53.

RESNICK, M.; MALONEY, J.; MONROY-HERNÁNDEZ, A.; RUSK, N.; EASTMOND, E.; BRENNAN, K.; MILLNER, A.; ROSENBAUM, E.; SILVER, J.; SILVERMAN, B. y KAFAI, Y. (2009). Scratch: Programming for All. *Commun. ACM*, 52 (11), pp. 60–67. En línea: http://doi.org/10.1145/1592761.1592779.

THE MOZILLA FOUNDATION, P2P UNIVERSITY, y THE MACARTHUR FOUNDATION. (2012). *Open Badges for Lifelong Learning*. En línea: https://wiki.mozilla.org/images/5/59/OpenBadges-Working-Paper_012312.pdf [14/03/2016]

VYGOTSKY, L. (1979). *El desarrollo de los procesos psicológicos superiores*. Barcelona: Crítica.