Una propuesta de enseñanza de programación en escuela media mediante el desarrollo de videojuegos con Etoys

Gonzalo Zabala[1], Ricardo Morán[1], Sebastián Blanco[1]

[1] Universidad Abierta Interamericana, Buenos Aires, Argentina
{gonzalo.zabala, ricardo.moran, sebastian.blanco}@uai.edu.ar

Abstract.
Desde diversas organizaciones relacionadas con la educación y las nuevas tecnologías ha resurgido la propuesta de incluir la enseñanza de lenguajes de programación en la escuela secundaria. En el presente artículo explicamos los motivos para esta propuesta, tanto internacionales como locales, la situación actual en la que se encuentran las escuelas secundarias argentinas, y realizamos una propuesta para introducir la enseñanza de programación mediante la creación de videojuegos.

Keywords:programación, enseñanza, videojuegos, escuela media, Etoys

1 Introducción

1.1 Motivo

Los profesionales de la educación, han definido durante años, con idas y vueltas, cuáles son los contenidos conceptuales que deben enseñarse dentro de las escuelas primarias y secundarias. En los últimos veinte años también se han explicitado las actitudes y procedimientos que se quieren transmitir a las nuevas generaciones. Pero en todo momento, desde los comienzos de la educación, hay un estigma que define en una sola pregunta el ideal docente: ¿cómo enseñamos a pensar a nuestros alumnos? ¿Qué herramientas debemos brindarles para que sean ciudadanos autónomos, con pleno ejercicio de sus libertades y provechosos para la construcción continua de la sociedad en la que viven? Hace no muchos años, leer, escribir y realizar las cuatro operaciones básicas se consideraron suficientes para la formación de los estudiantes. Sería redundante repetir que estos conocimientos, si no están acompañados de comprensión tanto para la interpretación de los textos como para la elaboración de discursos con sentido, carecen de utilidad en la actualidad. Pero también es preciso notar que la mayor parte del pensamiento humano hoy se desarrolla, se comparte y se aplica mediante el uso de las tecnologías de la información y la comunicación (TIC). Es decir, que la lectura comprensiva y la elaboración de textos con sentido hoy se realizan, en un alto porcentaje, utilizando este tipo de tecnologías. Por lo tanto, se debe proveer a nuestros estu-

diantes de conceptos, actitudes y procedimientos relacionados con su uso, y con la capacidad de adaptarse a nuevos paradigmas que surgen de manera vertiginosa en el procesamiento de la información. No hay actividad laboral que no esté cruzada por el uso de herramientas cada vez más avanzadas con respecto a la tecnología que hay en ellas.

Ahora bien, todas estas herramientas están creadas con lenguajes de programación, y a pesar de que han evolucionado sus interfaces aumentando la facilidad de uso, la presencia de la lógica de los lenguajes sigue presente desde la base misma de la arquitectura de los programas. Por ejemplo, muchos de ellos pueden ser usados en un nivel sencillo sin conocer nada sobre el tema, pero poseen un conjunto de opciones que, mediante algún tipo de lenguaje, potencia inmensamente su poder de procesamiento (macros en los procesadores de texto y las planillas, lenguaje SQL para la creación de consultas en bases de datos, lenguajes de configuración y otros).

Más allá de esta justificación pragmática, la elaboración de un programa para resolver un problema permite al alumno aprender poderosas estrategias para pensar, diseñar y encontrar una solución a dicho problema [7]. Esto se debe a que el estudiante lo debe entender profundamente para poder representar en un programa el mecanismo de solución encontrado, y comunicarlo a una computadora usando una sintaxis precisa, que refleje de esa manera su pensamiento [6]. Otra habilidad necesaria para programar es la de dividir el problema en subproblemas. Esto abre la puerta a encontrar esquemas de soluciones que pueden reutilizarse en diversos planteos, aún sin ser de la misma disciplina.

Los conceptos de variables y funciones, los mecanismos rigurosos de pensamiento y de descomposición del problema y la generalización también son atributos de la resolución de problemas en matemática. Es por eso que autores como Feurzeig proponen la enseñanza de programación como un camino complementario al aprendizaje de las matemáticas [2].

Por último, según el observatorio de la Cámara Argentina de Software y Servicios Informáticos, en Argentina hay una carencia de 7000 programadores por año. Por lo tanto, aquellos estudiantes que encuentren su vocación en este rubro, tienen asegurada su inserción laboral prácticamente desde los primeros años de la carrera. Invitarlos a recorrer este camino es abrir una puerta para un futuro promisorio.

1.2 Situación actual

A pesar de lo que presentamos en la sección anterior, en los últimos años ha desaparecido la enseñanza de programación en las aulas argentinas. En la década del 90 surgió una modalidad para el nivel medio conocida como “Bachillerato con orientación laboral”, que permitía una mayor carga horaria para materias específicas a dicha inserción. Bajo este proyecto surgieron muchos bachilleratos con orientación en informática, que contenían dentro de sus programas la enseñanza de programación como eje principal. Sin embargo, a comienzos de la década pasada, muchos de estos programas se reconvirtieron, orientándose hacia el diseño web y las herramientas multimediales. Una demostración de esta merma, es la cantidad de participantes de las Olimpíadas Informáticas Metropolitanas (OMI). A fines de la década del 90 partici-

paban alrededor de 100 estudiantes de 12 instituciones. La tabla que presentamos a continuación, muestra la concurrencia en los últimos años.

Año	Cantidad de instituciones	Cantidad de participantes
2007	8	33
2008	2	17
2010	2	17
2011	6	42
2012	6	34

Para revertir esta tendencia, la Fundación Sadosky ha lanzado un programa destinado a despertar vocaciones relacionadas con las TIC. Dentro del mismo, el proyecto "Desafío Dale Aceptar" invita a los estudiantes secundarios a participar de un concurso de animaciones, videojuegos y chatbot utilizando lenguajes de programación para su creación, con las plataformas "Alice" y "Chatbot" para su desarrollo. Otro proyecto es "Estudiar computación" donde se recopila y organiza en una web toda la información disponible sobre las carreras de informática actuales en nuestro país.

A continuación presentaremos una propuesta para la enseñanza de programación en la secundaria mediante la creación de videojuegos, como un aporte más al diseño curricular de las enseñanzas de las TIC.

1.3 ¿Qué requiere programar?

La programación es una actividad intelectualmente desafiante que involucra un conjunto diverso de actividades y ejercita varias formas de pensamiento.

En primer lugar, el programador debe ser capaz de entender e interiorizar los símbolos y las reglas sintácticas y semánticas del lenguaje de programación que se trate. Si bien no es necesaria una computadora para programar, todo programa debe estar circunscrito a algún conjunto de reglas estrictas que buscan evitar la ambigüedad. Estas reglas constituyen el lenguaje formal en que el programador debe ser capaz, por lo tanto, de expresar sus ideas. Asimismo, el programador debe poder identificar los elementos primitivos que ofrece el lenguaje de programación, los ladrillos más básicos de construcción a partir de los cuales pueden especificarse comportamientos más complejos. Aunque la separación en algún punto es difusa, los elementos primitivos de cualquier lenguaje pueden ser clasificados en dos grupos: datos y procedimientos. Prácticamente todos los lenguajes de programación exponen, en principio, datos primitivos como los números y procedimientos primitivos como las funciones aritméticas básicas. Estas actividades requieren el ejercicio de un tipo de pensamiento analítico, capaz de desagregar lógicamente las partes de un problema y componerlas en un todo inteligible.

Sin embargo, poder entender y atenerse a las reglas de un lenguaje de programación no es suficiente para ser un programador. Quizás sea suficiente para aprender un lenguaje de programación en particular y poder leer un programa ya escrito en el mismo, pero ciertamente la escritura de un programa desde cero requiere otro conjun-

to de habilidades, relacionadas con la habilidad para resolver problemas. Todo buen programador tiene una actitud proactiva y entusiasta al hacer frente a un problema. Esta actitud le permite al programador encontrar las mejores formas de combinar los elementos primitivos del lenguaje de programación a fin de resolver un problema en particular. La resolución de problemas requiere el ejercicio de un tipo de pensamiento creativo, acompañado usualmente por una gran curiosidad.

Una actitud hacia la resolución de problemas no es suficiente para ser un buen programador, sin embargo. Un tercer elemento faltante, quizás el más difícil y crucial, es la habilidad para identificar patrones en el problema e idear soluciones reutilizables. Esta actividad requiere el desarrollo del pensamiento abstracto, capaz de controlar la complejidad por medio de la construcción de abstracciones que oculten los detalles cuando sea necesario [1]. Todos los lenguajes de programación ofrecen herramientas para la abstracción (por ejemplo: variables, funciones, clases, métodos, etc.) y es responsabilidad del programador la manipulación de los mismos a fin de resolver un problema de la forma más elegante posible.

La programación, asimismo, requiere del programador mucha constancia, dedicación, y capacidad de concentración, sobre todo al momento de corregir programas incorrectos. El programador tiene la dificultad adicional de transitar siempre dentro de una paradoja: debe ser capaz de entender que un programa es una entidad operacional que debe ejecutarse de forma eficiente en una computadora, pero al mismo tiempo debe tener en mente que un programa es un medio para expresar ideas, independiente de la computadora en la cual se ejecuta [4].

La programación es una disciplina joven y, como tal, está en eterno crecimiento, y exige del programador el aprendizaje constante de nuevos conocimientos. Por estas razones, los mejores programadores suelen estar siempre a la búsqueda de nuevos problemas que atacar y nuevos proyectos que encarar.

1.4 Dificultades actuales para enseñar programación

Programar no es una actividad fácil, y enseñar a programar tampoco. En principio, los alumnos suelen tener dificultades con las reglas sintácticas de los lenguajes con los que suele introducirse la programación. La sintaxis presenta dificultades de aprendizaje al obligar a los alumnos a concentrarse en aprender sus reglas incluso en los ejemplos más simples, desviando la atención de lo verdaderamente importante: la resolución de problemas [3]. Si no pueden ser presentados a medida que van siendo necesarios, los elementos del lenguaje pueden llegar a confundir al alumno que se está iniciando, y el profesor muchas veces debe postergar su explicación a la espera del momento propicio en el cual introducir estos nuevos conceptos.

Los alumnos también suelen tener dificultades en relación a la resolución de problemas. No puede enseñarse a resolver problemas sino es a través de la práctica constante. Sólo programando se aprende a programar. Sin embargo, los alumnos en general muestran una falta de actitud hacia la práctica. Esto se ve agravado al considerar el tipo de ejercicios que suelen presentarse en los primeros cursos de programación: ordenar un vector, calcular el factorial de un número, definir la secuencia de fibonacci, entre otros. Ejercicios como estos resultan poco interesantes para la mayoría de los

alumnos debido a que no presentan una recompensa tangible e inmediata para el esfuerzo del alumno. Sin una recompensa inmediata algunos alumnos se desaniman y pierden la motivación para realizar la cantidad de práctica necesaria como para comprender los conceptos que se van dictando en la clase, desarrollando a la larga una frustración que convierte la experiencia de programar en un sufrimiento.

Por otro lado, la situación actual de la enseñanza presenta alumnos con una formación matemática incompleta o incorrecta, que tienen dificultad para manipular y construir abstracciones [4]. El desarrollo del pensamiento abstracto es un componente fundamental en la enseñanza de programación, sin el cual los alumnos no llegan a alcanzar el nivel de formación necesaria para convertirse en buenos programadores.

Todas estas dificultades convierten la tarea de enseñar programación en un desafío que obliga a los educadores repensar las formas y métodos de enseñanza.

2 ¿Por qué videojuegos?

Las razones fundamentales por las que se elige enseñar programación mediante la creación de videojuegos son:

Abstracciones más reconocibles.

Habiendo mencionado la dificultad de abstracción como un problema primordial a la hora de enseñar programación, cabe destacar que los videojuegos son un campo de la programación donde los elementos del dominio del problema son suficientemente reconocibles (personajes, enemigos, bloques, fondo, etc.) como para permitir un enfoque más concreto, dejando lugar a conceptos abstractos como ser el comportamiento de cada una de las entidades y cómo se vinculan entre sí.

Curva amena de dificultad.

La creación de videojuegos no necesariamente implica realizar un producto muy complejo. Existen infinidades de ejemplos que pueden ser realizados por una persona en menos de lo que dura un curso estándar de programación. A medida que el alumno vaya internalizando conceptos, se puede optar por ejercicios de mayor complejidad.

Creatividad y ludo.

Actualmente, los videojuegos son considerados una forma de arte que congrega cada vez más personas de diferentes disciplinas a la hora de su creación. El desarrollar la creatividad haciendo algo artístico multidisciplinario ayuda parcialmente a satisfacer la necesidad de pensamiento creativo para encontrar soluciones dentro de problemas de la programación. Si a esto le combinamos que la idea de desarrollar videojuegos es utilizar la predisposición natural del ser humano a lo lúdico obtenemos que el aprendizaje de programación sea más fluido y entusiasta.

Recompensa tangible.

Al finalizar la creación del videojuego, el alumno no sólo se queda con lo aprendido sino también con un programa funcional en vez de un conjunto de ejemplos poco motivadores [3]. A los alumnos les gusta tener una recompensa mucho más significativa que realizar ejemplos como ordenar números aleatorios. Y es esta motivación en gran parte lo que permite generar la constancia de esfuerzo en el programador.

3 ¿Por qué Etoys?

3.1 ¿Qué es Etoys?

Etoys es una herramienta de autor multimedia y un sistema de programación visual desarrollada por los mismos creadores de Smalltalk, Alan Kay y Dan Ingalls, con el asesoramiento pedagógico de importantes pedagogos como Seymour Papert y Jerome Bruner. Etoys provee una interfaz visual para crear y manipular objetos directamente, mediante el uso de metáforas ya tan extendidas y naturales como clickear y arrastrar. Sin embargo, su aparente simpleza esconde una plataforma de programación en Smalltalk muy poderosa que está disponible para cualquiera que esté dispuesto a explorarla.

En un primer momento, Etoys puede considerarse una herramienta de dibujo pues presenta a los usuarios un conjunto de utilidades comúnmente relacionadas con programas como MS Paint o Photoshop.

Fig. 1. La herramienta de dibujo de Etoys incluye los típicos elementos que podemos encontrar en una herramienta de este tipo

Un dibujo en Etoys puede ser manipulado usando el mouse de un modo similar a un dibujo en Photoshop: se lo puede mover, levantar, agrandar, rotar, duplicar, eliminar, etc.

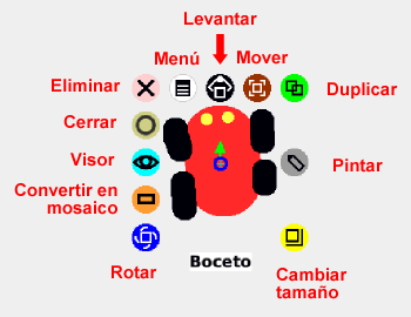

Fig. 2. El "halo" permite interactuar con cada objeto directamente

La diferencia fundamental radica en que los dibujos hechos en Etoys no yacen estáticos en la pantalla sino que pueden adquirir un comportamiento dinámico basado en la construcción de guiones. Siguiendo la terminología del paradigma de orientación a objetos, puede decirse que cada dibujo o elemento gráfico que presenta Etoys en la pantalla es un objeto en sí mismo; con atributos como el ancho, el alto, la posición en la pantalla, entre otros; y con métodos, algunos primitivos, como "avanzar" o "girar", y otros creados por el usuario, llamados "guiones". La interfaz para la creación de guiones es completamente visual, y cada línea de código es representada gráficamente mediante mosaicos que se encastran unos con otros para especificar el comportamiento deseado.

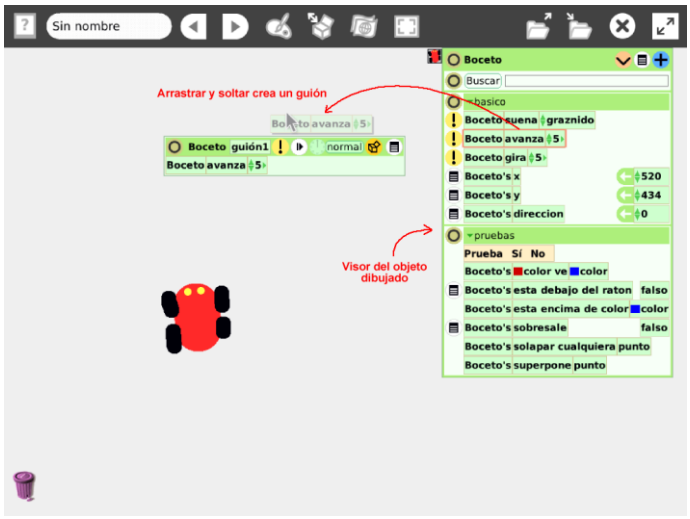

Fig. 3. El Visor es la herramienta a partir de la cual pueden crearse comportamientos para los objetos

La metáfora de los mosaicos como ladrillos de construcción del comportamiento resulta muy útil para prescindir de la sintaxis de un lenguaje de programación convencional: el mosaico o entra o no entra en el guión, no existen errores de sintaxis.

Etoys fusiona la interfaz de usuario con el modelo, todos los objetos tienen interfaz gráfica y se aplica para todos la misma filosofía de manipulación directa (¡incluso para los objetos que componen la propia interfaz!).

Fig. 4. Todos los elementos gráficos, incluso los que componen la interfaz de Etoys, son manipulables de la misma manera

La programación en Etoys presenta características profundamente diferentes que en el resto de los lenguajes de programación, incluso los visuales. Para empezar, los guiones en Etoys no se ejecutan en el sentido tradicional del término sino que, usando la terminología de Etoys, "laten". Los guiones en Etoys se ejecutan dentro de un ciclo implícito para el programador que pretende simular el avance constante del tiempo en el mundo real. Por esta razón, los objetos en Etoys se sienten "vivos", uno interactúa con ellos directamente y ellos responden de manera instantánea, no existe la ilusoria e innecesaria separación entre la edición y la ejecución, pues, así como el mundo real, el mundo de Etoys no se detiene mientras nosotros interactuamos con él. El latido de los guiones puede controlarse sencillamente mediante mosaicos, permitiendo un control muy preciso del comportamiento de los objetos.

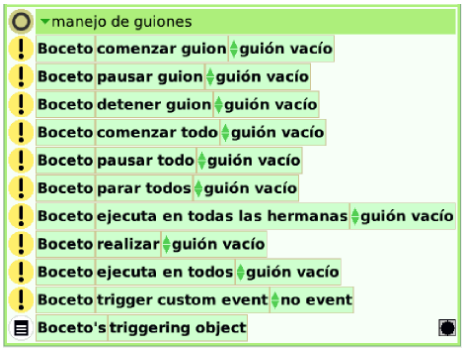

Fig. 5. El latido de los guiones puede controlarse programáticamente mediante mosaicos

Asimismo, no hay un límite a la cantidad de guiones que puede haber activos al mismo tiempo (el límite lo dispone, en todo caso, el hardware), y todos los guiones

están ejecutándose virtualmente al mismo tiempo. Esto expone al alumno de una forma muy temprana y natural a la concurrencia, que no deja de ser la forma en que observamos el mundo.

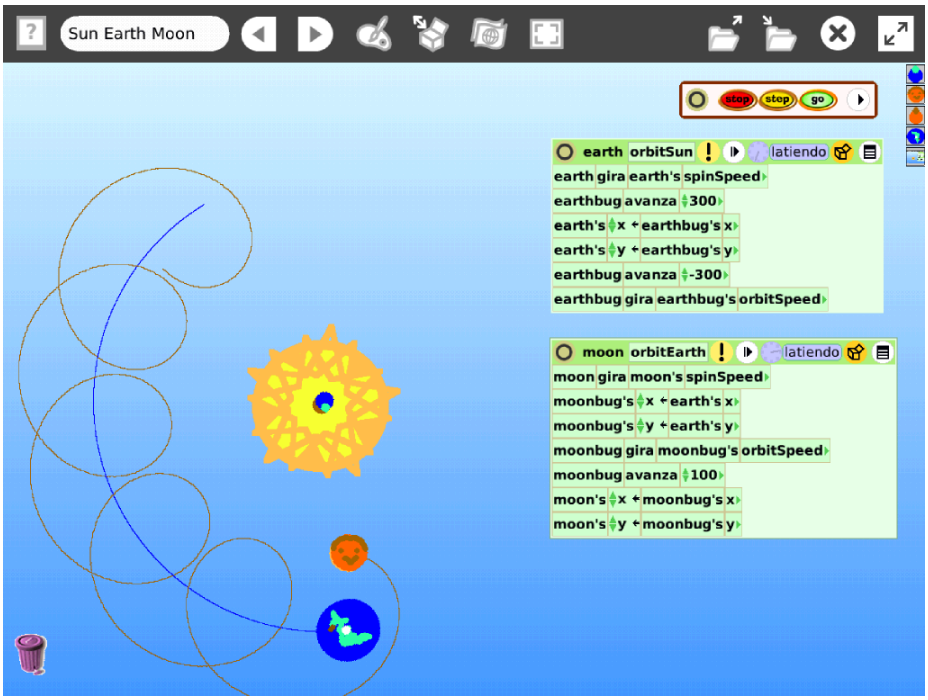

Fig. 6. Un proyecto de simulación de las órbitas de la Tierra y la Luna, con la ejecución simultánea de dos guiones

Por otro lado, prácticamente todos los objetos en Etoys presentan el mismo conjunto de instrucciones primitivas, aunque su interfaz cambie. Es decir que todos los objetos son en esencia iguales aunque usen diferentes "disfraces", esto garantiza la consistencia presentando al usuario siempre la misma funcionalidad.

3.2 ¿Por qué Etoys?

Expresividad y facilidad de modelado.
Como los videojuegos son arte, el entorno de desarrollo debe ser apropiado para facilitar la expresión sin dejar de lado a la programación como pieza fundamental. En Etoys todo lo que uno dibuja puede ser fácilmente transformado en objetos y los mismos poseen propiedades y comportamiento que se puede modificar posteriormente según el tipo de abstracción requerida para el videojuego. Esta fusión entre arte y programación permite utilizar a la computadora como el medio de expresión interactivo para lo que originalmente fue concebida [8]. Muchas veces el acto de dibujar hace facilitar la comprensión del problema, florecer el instinto y la prueba sin tener

que hacer referencia constante a la teoría. Como los dibujos realizados tienen un significado dentro del videojuego, permite que el agrupamiento de responsabilidades hacia los objetos sea más representativo.

Sintaxis sencilla.
El código en Etoys se representa por medio de mosaicos y las reglas sintácticas son bastante minimalistas y apelan a principios mnemotécnicos para su mejor comprensión. Como la graficación y la concurrencia están ya resueltas se resuelve de antemano una parte de la complejidad del problema para hacer un videojuego. Definir variables, objetos, comportamiento o estructuras de control lleva poco tiempo. Entender la sintaxis para programar en Etoys no requiere habilidades lingüísticas avanzadas debido a que está orientado para alumnos con una edad promedio de 10 años y determinados tipos de comportamiento se pueden modelar con menos trabajo ya que se pueden utilizar los colores que se ven en pantalla para hacer detección de colisiones, modelar el tiempo o hacer relaciones aritméticas según la geometría de los objetos en pantalla o viceversa.

Feedback instantáneo.
La característica primordial que tiene un videojuego a diferencia de las demás formas de arte es la interacción; el sujeto que antes era un simple espectador pasa a ser activo. Este intercambio bidireccional necesario entre obra/sujeto hace necesario un entorno de programación en el que se obtenga feedback instantáneo mientras se está programando. Etoys, por su parte, permite la manipulación directa de los elementos del problema pudiendo los alumnos ver instantáneamente los resultados de su interacción. Todos los objetos que uno crea pueden interactuar entre sí en tiempo real dentro de un mundo vivo gracias al modelo de concurrencia que posee Etoys. Esto permite que desarrollar juegos sea mucho más fácil y con menos líneas de código.

Open source.
Etoys está construido sobre un ambiente de objetos muy poderoso llamado Smalltalk, y aunque el mismo está escondido para evitar abrumar al usuario no experimentado, todas las herramientas de desarrollo pueden accederse con sólo cambiar una preferencia.

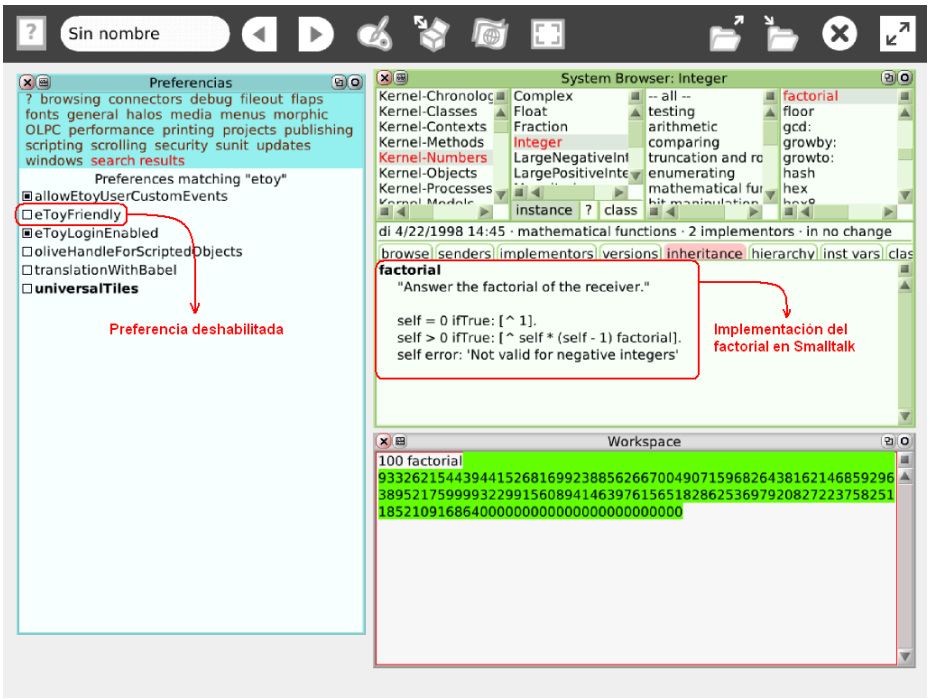

Fig. 7. Etoys permite al usuario acceder a las herramientas de desarrollo Smalltalk

Es más, los guiones mismos que el usuario construye en el fondo son traducidos a Smalltalk, cuyo código puede verse y modificarse directamente en el "Editor de guiones".

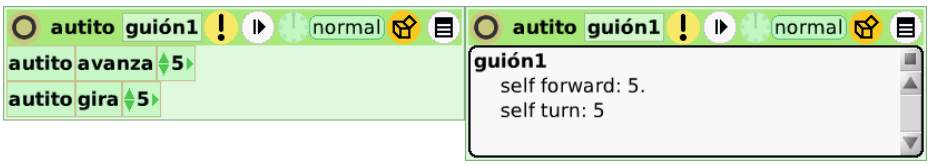

Fig. 8. El editor de guiones puede mostrar tanto el código en formato mosaicos o en formato texto

Esto permite una transición muy natural del entorno visual diseñado especialmente para chicos, al entorno de programación completo.

Gratuito y multiplataforma.
Etoys no tiene costo alguno y puede ejecutarse en Windows, Linux, Mac OS y Sugar. A su vez se encuentra instalado por defecto en las netbooks correspondientes al Plan Conectar Igualdad y Plan Sarmiento.

Comunidad activa y orientado a diferente público.
Etoys posee una comunidad bastante activa en todo el mundo y dispuesta a ayudar ya sea dando charlas o resolviendo dudas puntuales en sus foros. El lenguaje de programación visual está orientado a docentes, desarrolladores y principalmente a alumnos. Además es transversal a todas las disciplinas que se pueden encontrar tanto en escuela primaria como media y se demuestra con su gran portfolio de ejemplos en su sitio web oficial.

Multi idioma.
Contar con un software que permita cambiar a cualquier idioma en cualquier momento de la ejecución es muy útil ya que fomenta la inclusión de distintas culturas permitiendo un intercambio más rico de experiencias. Etoys cuenta con más de 20 lenguajes.

Extensible a material concreto.
En el año 2010, la Universidad Abierta Interamericana realizó una extensión de Etoys llamada Physical Etoys para poder manejar diferentes tipos de dispositivos electrónicos como Kinect, robots Lego NXT, Sphero, plaquetas Arduino y Duinobot, entre otros. De esta manera, con la misma filosofía, se puede aprender programación teniendo en cuenta el entorno físico logrando un mayor significado en la creación y que la complejidad del mundo real defina los resultados.

4 Ejemplos

La mayoría de los ejemplos a continuación fueron diseñados de forma tal que la totalidad del código (o su parte más representativa) pueda presentarse en una captura de pantalla. No fueron desarrollados en función de la calidad del videojuego o sus características de jugabilidad, si bien los alumnos seguramente enfocarán sus esfuerzos en este aspecto.

Se eligieron videojuegos clásicos como ejemplo de modo que no sea necesario explicar sus reglas al lector. Tampoco se explicará a continuación cómo implementar estos videojuegos, pero los mismos podrán ser descargados del sitio web de los autores.

Cada uno de los videojuegos introduce algún concepto (o varios) de programación. Sin embargo, no es necesario que el alumno conozca y entienda formalmente el concepto antes de poder aplicarlo. Por el contrario, la introducción de estos conceptos se hace en Etoys de manera informal y a medida que van siendo necesarios, de modo que los alumnos puedan aprenderlos por medio del propio uso. Una vez el conocimiento está internalizado puede referirse al mismo en cursos posteriores, donde estos conceptos se introduzcan formalmente. El recuerdo de la experiencia entonces ayuda al entendimiento de conceptos que, de otro modo, resultan muy abstractos para ser comprendidos en su totalidad.

4.1 Pong

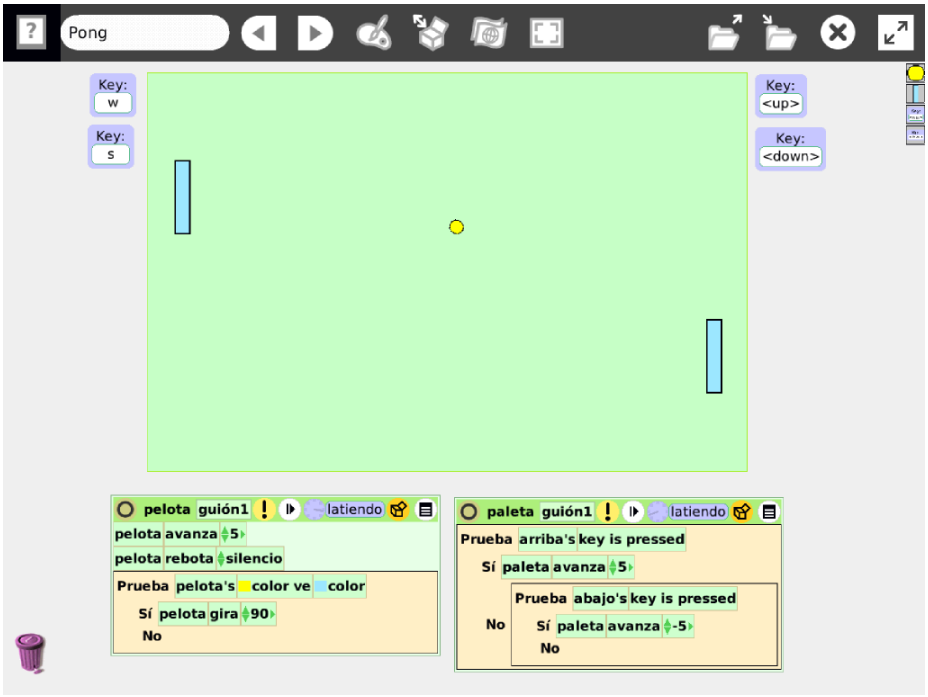

Fig. 9. Captura de pantalla del Pong

El presente juego de Pong fue realizado con sólo 8 líneas de código. Si bien sólo implementa la mecánica básica del juego (no se detecta cuando pierde un jugador ni se guardan los puntos) es un buen ejemplo de un videojuego realmente sencillo.

Como puede observarse el movimiento de cada paleta y de la pelota están implementados en guiones que laten de manera independiente, ilustrando los efectos de la concurrencia implícita y facilitando la separación de responsabilidades. Asimismo se observa el uso de estructuras condicionales, que en Etoys toman la forma del mosaico "Prueba".

Fig. 10. Estructura condicional (mosaico "Prueba") para detectar las colisiones entre la pelota y la paleta

4.2 Arkanoid

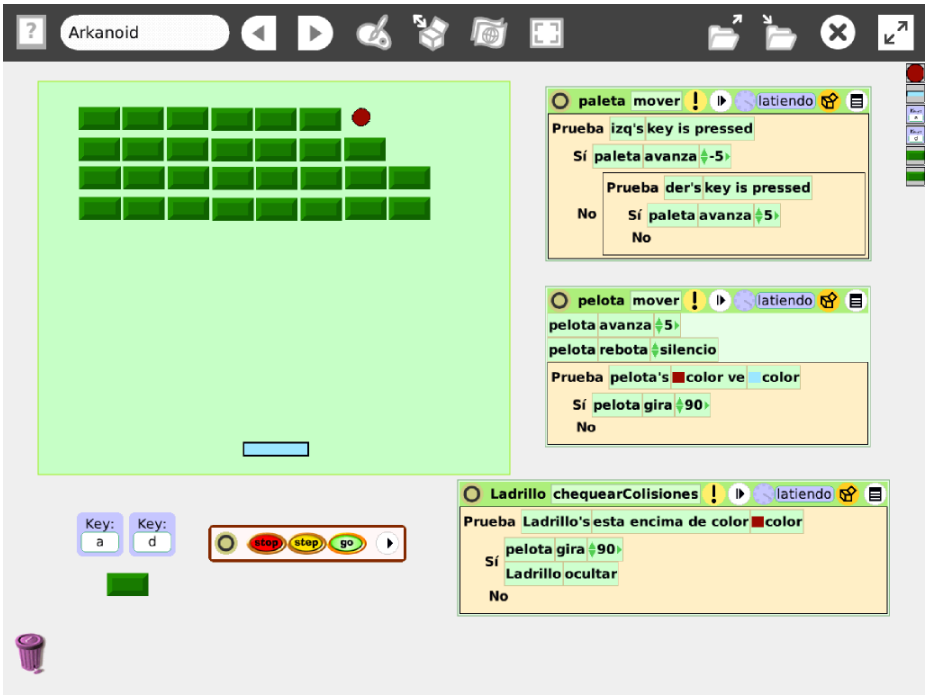

Fig. 11. Captura de pantalla del Arkanoid

El juego de Arkanoid está desarrollado en sólo 11 líneas de código. Presenta sólo la funcionalidad básica, la paleta puede moverse usando el teclado, la pelota rebota del mismo modo que en el Pong, y los ladrillos desaparecen al contacto con la pelota, no se guardan puntos ni se detecta cuando el jugador pierde.

Del mismo modo que en el Pong, las responsabilidades de cada objeto se implementaron en guiones separados: la pelota y la paleta saben como moverse, y los ladrillos como chequear colisiones. En este caso, además, se observa el uso de prototipos para reutilizar el comportamiento del ladrillo: sólo fue necesaria la implementación de un ladrillo, el cual luego fue duplicado y sus copias colocadas en el campo de juego.

4.3 Paperairplane

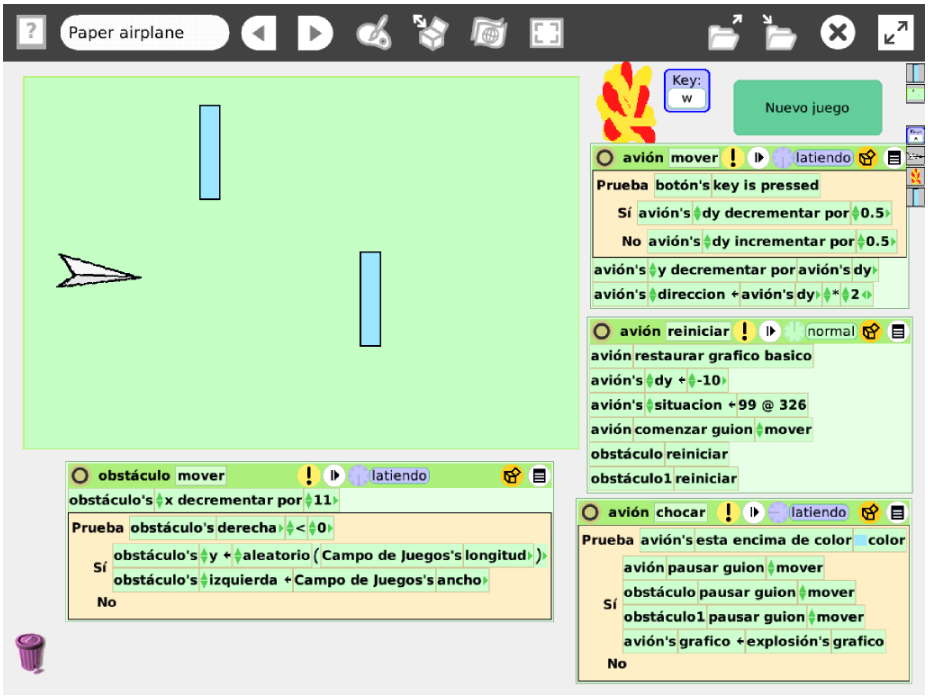

Fig. 12. Captura de pantalla del Paper airplane

Este ejemplo consiste en un avión de papel que debe esquivar un conjunto de obstáculos que van apareciendo en su vuelo. La mecánica básica del vuelo del avión, el movimiento de los obstáculos, y el chequeo de colisiones fue codificado en 14 líneas de código, otras 6 líneas de código se utilizaron para implementar el botón de "Nuevo juego", llegando a un total de 20 líneas de código.

En este caso, para simular la caída del avión con aceleración fue necesario introducir el concepto de variable. Las variables en Etoys tienen un tipo (que puede ser numérico, booleano, gráfico, color, entre otros), sin embargo, Etoys por defecto asigna el tipo Número a todas las variables nuevas por lo que no fue necesario introducir el concepto de tipos para este ejemplo.

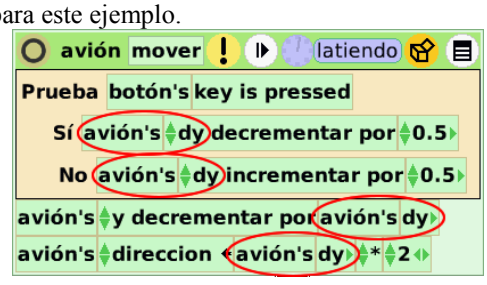

Fig. 13. Uso de variables en el guión que controla el movimiento del avión

4.4 Sokoban

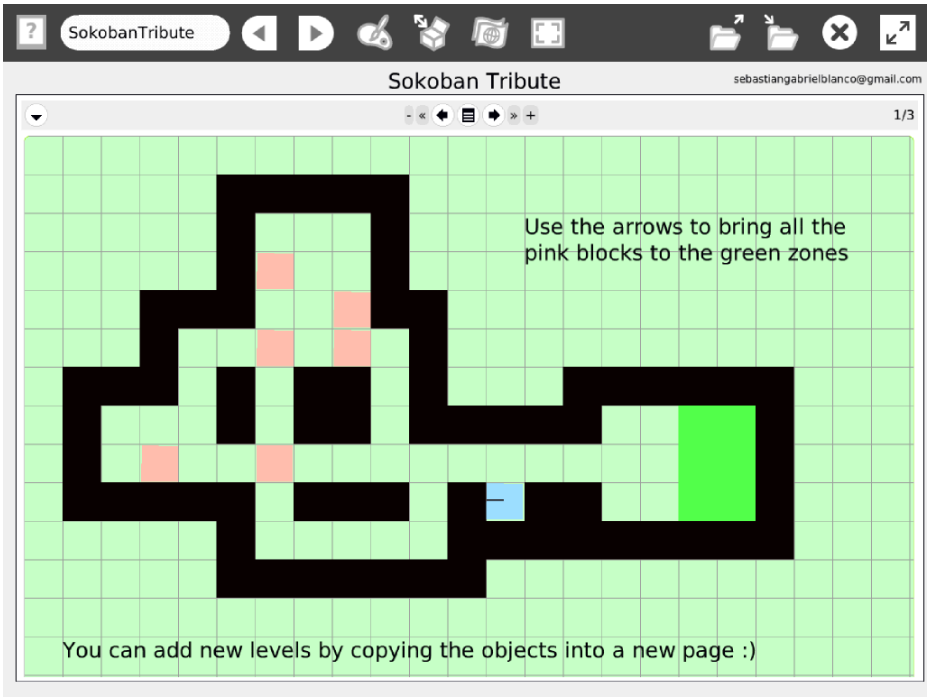

Fig. 14. Captura de pantalla del Sokoban

En este ejemplo se puede ver una réplica del juego Sokoban. Es interesante ver cómo con unas pocas líneas se obtiene el comportamiento de los bloques y del jugador. El concepto de prototipo en este ejemplo se vuelve necesario para ahorrar una miríada de líneas de código ya que a la hora de editar un nivel nuevo, lo único que se debe hacer es copiar los bloques. La concurrencia y las condiciones también se vuelven necesarias debido a que el juego constantemente está fijándose si los movimientos de los bloques o del protagonista son válidos.

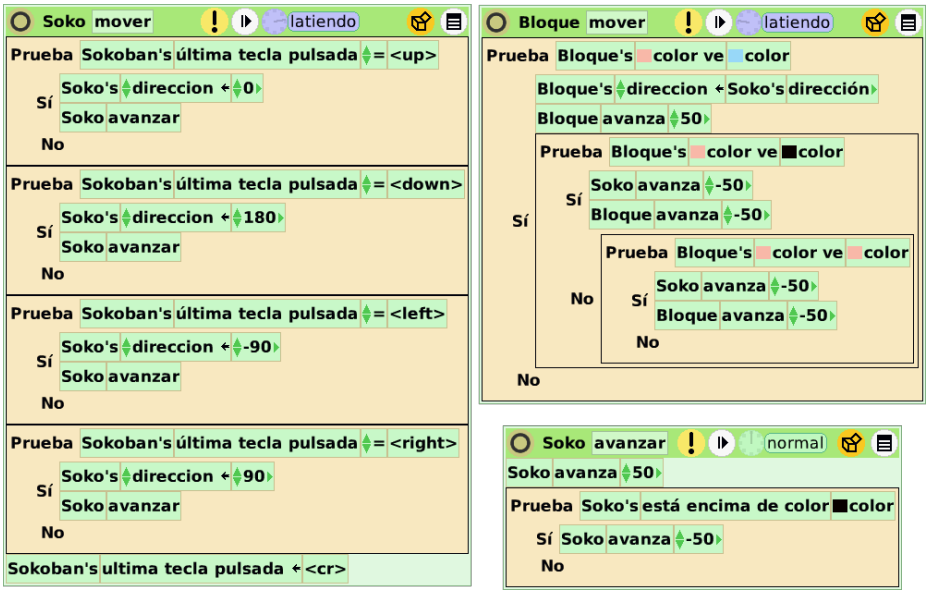

Fig. 15. Código del Sokoban

4.5 SpaceInvaders

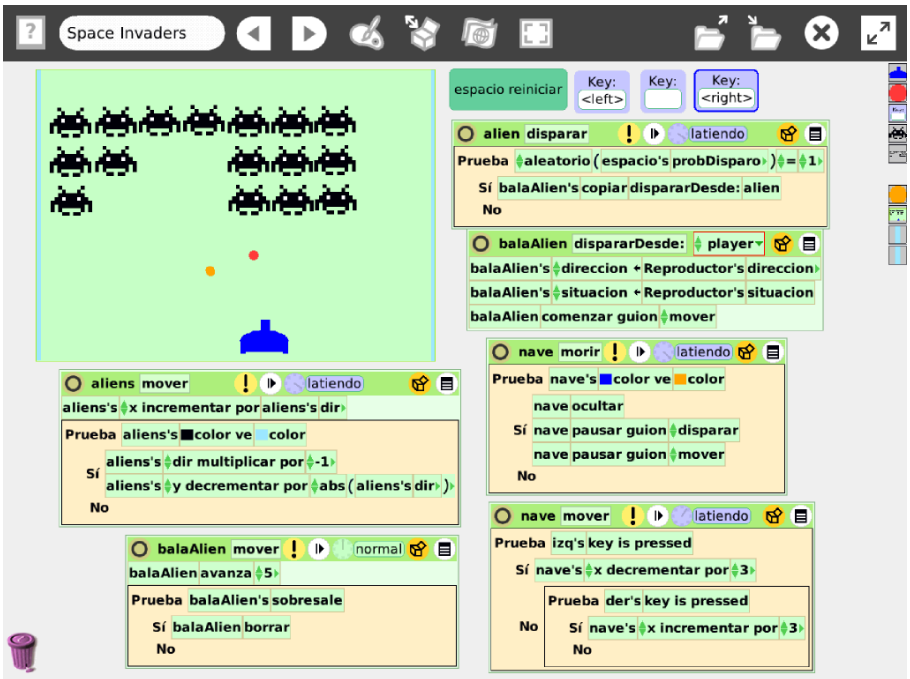

Fig. 16. Captura de pantalla del Space Invaders

Este ejemplo es un poco más complejo que los anteriores, por lo que sólo se muestra en la captura de pantalla la parte del código más representativa.

En este caso se implementó el movimiento coordinado de las naves alienígenas, así como el disparo de ambos bandos (en el caso de la nave terrestre a partir de presionar la barra espaciadora, y en el caso de las naves alienígenas dominado por un valor aleatorio).

Se observa en este caso que, para implementar el disparo, la bala recibe como parámetro el objeto desde el cual se está disparando, de modo que pueda ésta ubicarse en su misma posición y dirección antes de salir despedida hacia adelante. Esta implementación necesitó, pues, del uso de un guión con parámetro, así como de la especificación del tipo del parámetro (por defecto numérico, pero en este caso del tipo "Player").

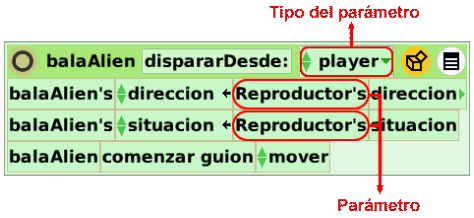

Fig.17. Uso de parámetros en el guión que controla el disparo de un cañón alienígena

Asimismo, el movimiento coordinado de las naves alienígenas requirió el uso de objetos incrustados dentro de otros, lo cual permite especificar una jerarquía de objetos, donde la interacción con el objeto padre afecta también a los objetos hijos. En este caso particular, todas las naves alienígenas se encuentran incrustadas en un rectángulo transparente denominado "aliens", que es el responsable del movimiento de todas las naves.

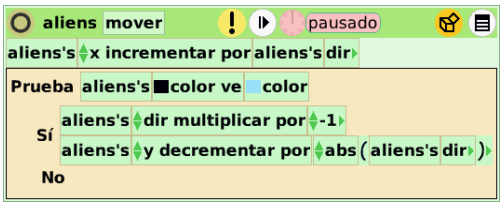

Fig. 18. El guión que controla el movimiento coordinado de las naves alienígenas

Finalmente, el botón de reiniciar envía el mensaje "reiniciar" a todos los objetos que componen el juego. Esto demuestra dos conceptos muy importantes. Por un lado, se introduce el protocolo de colecciones, a través del cual un objeto puede conocer e interactuar con los objetos contenidos dentro de él. Por otro lado, se introduce informalmente el concepto de polimorfismo: cada objeto implementa el método "reiniciar" a su manera, y el sólo envío del mensaje "reiniciar" disparará diferentes comportamientos dependiendo del objeto que reciba el mensaje.

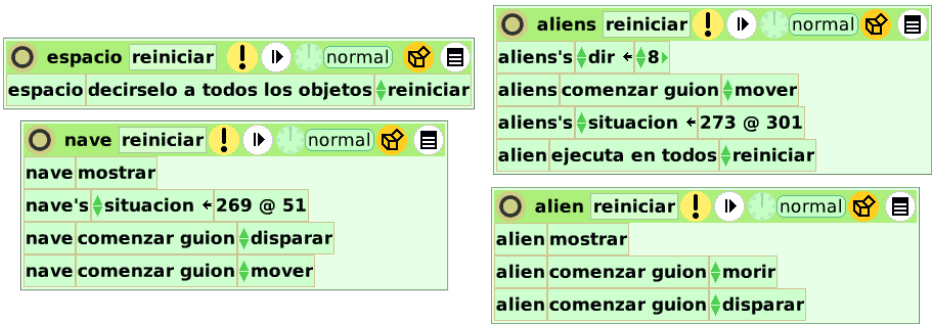

Fig. 19. Varios objetos implementan el método “reiniciar” de diferente forma

4.6 Bomberman

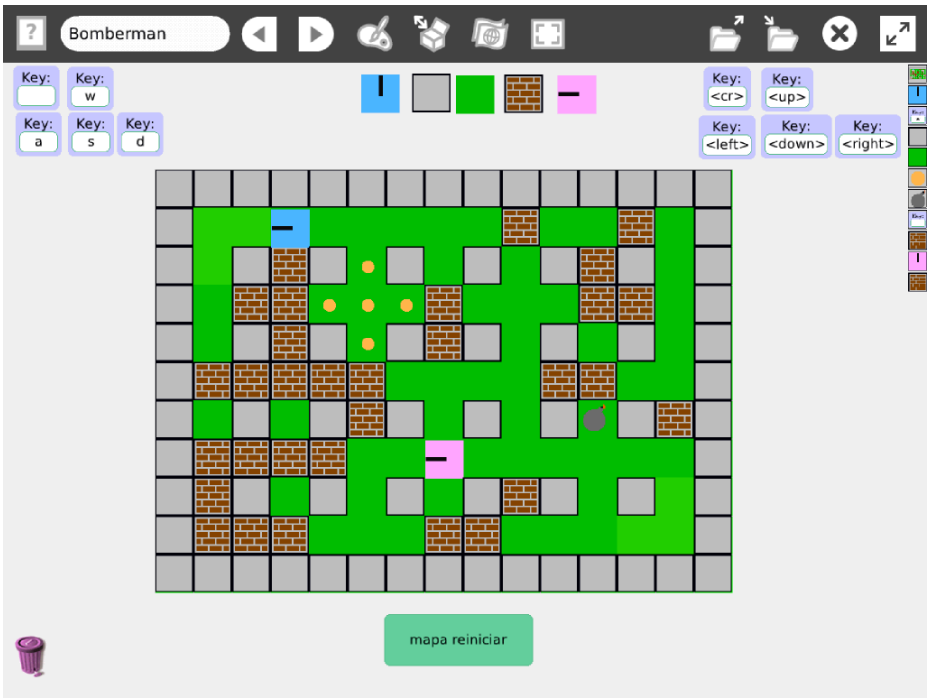

Fig. 20. Captura de pantalla del Bomberman

Este ejemplo es bastante más complejo que los anteriores, por lo que su código no entra en la captura de pantalla.

Se implementó el funcionamiento básico del juego de Bomberman para dos jugadores, la ubicación de bombas y la explosión de las mismas destruyendo todas las paredes aledañas.

Como en los casos anteriores cada guión se ejecuta de manera concurrente; se utilizaron estructuras condicionales; cada pared es una copia de la pared prototipo; se

usaron variables para definir la duración de la bomba sin explotar y la duración de la explosión; la ubicación de las bombas requiere de guiones con parámetros; y el botón de reiniciar hace uso del polimorfismo y del protocolo de colecciones.

Un concepto nuevo que se introduce en este ejemplo es el de guiones recursivos. En este caso, la ubicación aleatoria de las paredes al reiniciar el juego se logró a partir de un guión que reubica la pared en un lugar al azar y, en caso de detectar una colisión con otra pared, el guión se evalúa a sí mismo a fin de encontrar otra posición.

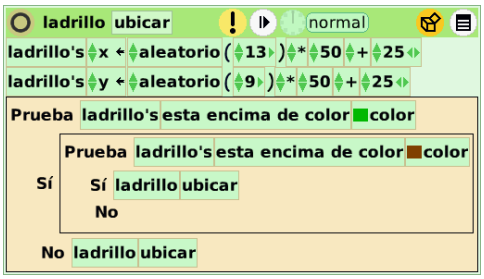

Fig. 21. Recursividad en el guión para ubicar aleatoriamente las paredes

4.7 Asteroids

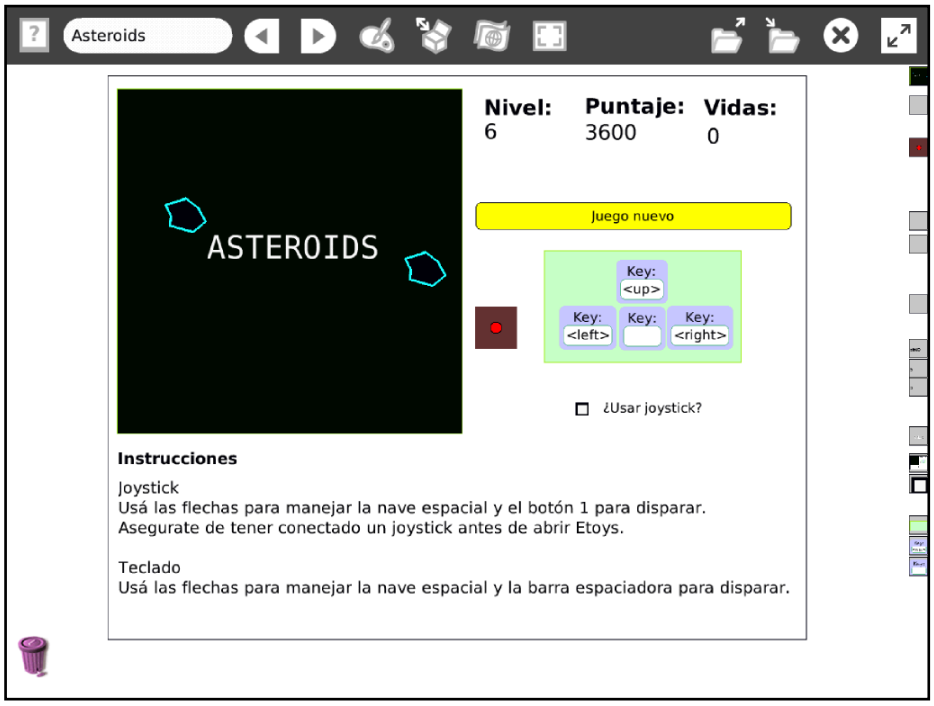

Fig. 22. Captura de pantalla del Asteroids

Por lejos, este ejemplo es el más complejo y sólo se usará a modo demostrativo de las posibilidades que ofrece Etoys en cuanto a los videojuegos que pueden llegar a implementarse. En este caso se modeló casi en su totalidad el clásico videojuego de Asteroids, con animaciones para la explosión de la nave y los asteroides, puntaje, niveles con creciente dificultad, y jugabilidad tanto con el teclado como con un joystick usb.

A pesar de su complejidad, el código del presente ejemplo no utiliza conceptos diferentes a los ya mencionados en los ejemplos anteriores, sólo los combina de formas diferentes para llegar al resultado deseado.

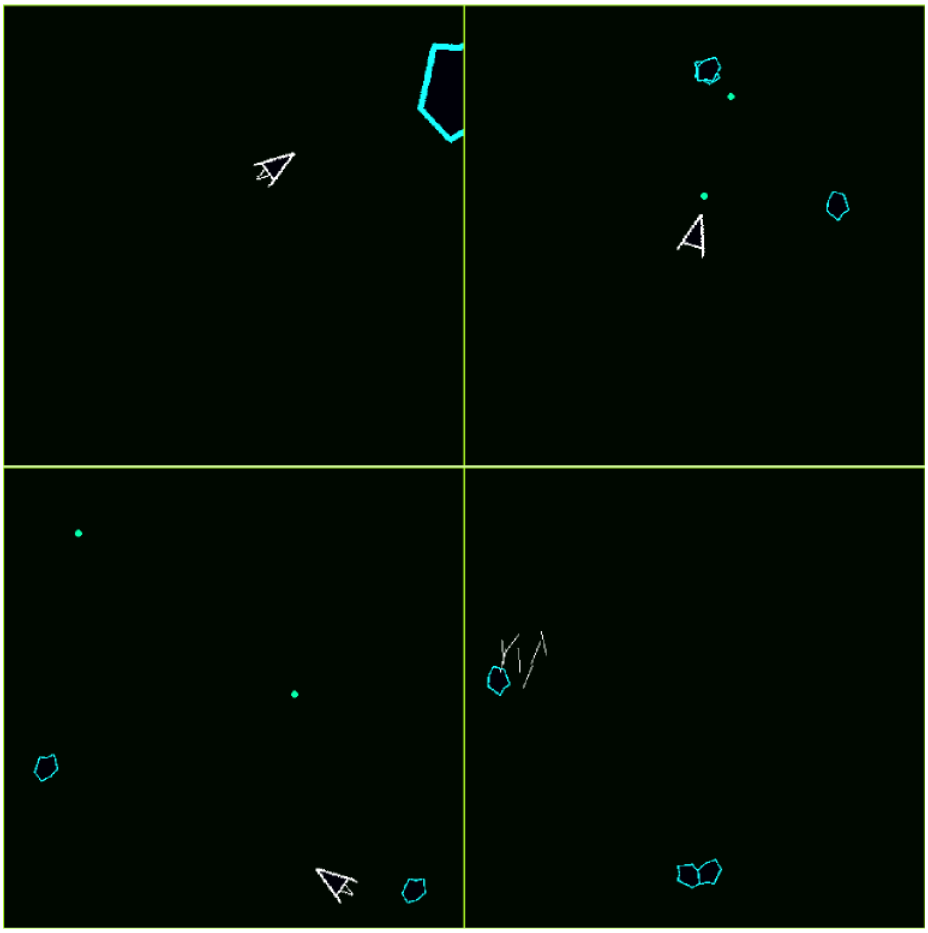

Fig. 23. Jugando al Asteroids en Etoys

5 Conclusión y trabajos futuros

Lo que hemos presentado en el punto anterior es sólo la punta de un inmenso iceberg de conocimiento. Etoys, como ambiente de objetos, permite el modelado de diversas áreas de conocimiento, inclusive más allá de las ciencias duras. En las actividades que hemos propuesto, se conjugan muchos conocimientos distintos relacionados con la lógica, la matemática y la física, en un marco de entretenimiento que evita el aburrimiento y la desmotivación. Como hemos visto, la programación necesaria es sencilla, y las herramientas gráficas que se utilizan no requieren de conocimientos expertos para utilizarlas. Sin embargo, nos quedan algunos asuntos que consideramos fundamental resolver para lograr una efectiva y duradera inserción de la enseñanza de la programación en las aulas de secundaria.

5.1 ¿En qué disciplina incluimos estas actividades?

En casi todos los primeros años de las secundarias, existe una materia denominada "Informática". En un momento sus contenidos estaban relacionados con el aprendizaje de herramientas de oficina tradicionales, como procesadores de texto, planillas de cálculo y generadores de presentaciones. Pero hoy en día, los estudiantes llegan a la secundaria conociendo su uso en mayor o menor medida. Es por eso que consideramos que Informática es un buen espacio curricular para la enseñanza de programación. **Aún así, creemos que es una buena herramienta para todas las disciplinas. Es decir, el uso de la programación es completamente transversal. Por lo tanto, ceñir su uso a una disciplina en particular es quitarle sentido.** ¿Es posible programar en Prácticas del Lenguaje, en Ciencias Naturales, en Artes? Sí, pero aquí surge otro problema fundamental.

5.2 ¿Quién enseña a programar, si en general los docentes no saben hacerlo?

Podríamos caer en la tentación de generar cientos de cursos de capacitación para docentes, que suman un trabajo más a su ya ardua tarea cotidiana. Es por eso que nos sumamos a la propuesta de Adrián Paenza [5]: *"[...]qué pasaría si los alumnos y los docentes aprendieran juntos? Es decir, ¿qué pasaría si todos los días (...) los alumnos tuvieran en todos los colegios y escuelas del país, una hora en donde la educación se transforma en algo "horizontal": todo el mundo aprende al mismo tiempo. Por supuesto, puede haber (o mejor dicho, debería haber) literatura suficiente (sencilla) para que entre todos intenten resolver los problemas que allí están planteados. Algunos podrán un poco más. Otros un poco menos. Algunos necesitarán más ayuda, otros menos. Pero dentro de la misma escuela (o colegio), habrá grupos que podrán cooperar con los que tienen más dificultades. En ese caso, las diferencias de edades y de grados y de "jerarquías" deberían quedar de lado."* Con esta idea, podríamos sumar a docentes de todas las materias a usar en algún contenido herramientas de programación que permitan estudiar el problema desde otra perspectiva. Pero como bien dice Paenza, es necesario proveer a docentes y alumnos una bibliografía sencilla,

pautada, con orden de dificultad creciente, que les permita la resolución de problemas con éxito.

Es por eso que estamos proyectando el desarrollo de un conjunto de propuestas de actividades para todas las disciplinas usando Etoys como espacio de experimentación y reflexión. El objetivo es generar un repositorio web donde claramente se explicite por cada actividad la disciplina a la que pertenece, el contenido curricular que encara y un conjunto de pasos precisos para llevar adelante la dinámica en el aula. Entendemos que de esta forma estamos sumando un recurso más a los ya presentes en la web, pero con una mirada que aún no hemos encontrado en dichas propuestas.

Referencias

1. Abelson, Hal, y Gerald Jay Sussman. Structure and Interpretation of Computer Programs. MIT Press, 1984.
2. Feurzeig, W., S. Papert, M. Bloom, R. Grant, y C. Solomon. «Programming-language as a conceptual framework for teaching mathematics.» Interactive Learning Environments (Interactive Learning Environments) 19 (2011): 487-501.
3. Malan, David J., y Henry H. Leitner. «Scratch for Budding Computer Scientists.» ACM SIGCSE Bulletin 39(1) (2007): 223-227.
4. Martínez Lopez, Pablo E., Eduardo A. Bonelli, y Federico A. Sawady O'Connor. «El nombre verdadero de la programación.» 10° Simposio sobre la Sociedad de la Información, SSI 2012, 2012.
5. Paenza, Adrián. «Educación horizontal.» Página 12, 9 de Mayo de 2013: Contratapa.
6. Papert, Seymour. Mindstorms. Children, Computers and Powerful Ideas. Nueva York: Basic Books, Inc. Publishers, 1980.
7. Saeli, Mara, Jacob Perrenet, Wim M.G. Jochems, y Bert Zwaneveld. «Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective.» Informatics in Education, 2011: 73-88.
8. Victor, Bret. «Stop Drawing Dead Fish.» ACM SIGGRAPH, 2012.