Aprender a programar. ¿Solo una moda?

Belén Palop[1]

[1]*CompuEdu@UVa, Didáctica de la Matemática, Universidad de Valladolid*
bpalop@infor.uva.es

RESUMEN

La necesidad mundial de profesionales técnicos y, en particular, de programadores está trayendo a las aulas de Primaria y Secundaria la programación de ordenadores. De la misma manera que no se enseña lengua con el objetivo de alimentar una cantera de novelistas, nosotros defendemos la programación como una herramienta que favorece el desarrollo de competencias tan necesarias hoy día como la visión crítica de la tecnología, el desarrollo del pensamiento lógico, o la resolución de problemas. En este artículo abogamos por la programación como un nuevo lenguaje con el que los jóvenes pueden expresarse y desarrollar la creatividad. Además, defendemos un modelo de enseñanza *con* la programación y no *para* la programación, desmarcándonos del enfoque finalista de abastecer el mercado laboral y centrándonos en aquellas habilidades que se adquieren por el hecho de aprender a programar. Ni todos los niños se apasionan por la programación, ni todos los que programan se convertirán en programadores.

Presentaremos brevemente el lenguaje de programación Scratch, el más ampliamente utilizado mundialmente para la enseñanza de la programación en los niveles iniciales, y discutiremos cómo el grupo CompuEdu@UVa viene trabajando desde 2011 en la elaboración de un modelo didáctico de la docencia de la programación y en la divulgación de esta herramienta

1. INTRODUCCIÓN

Aprender a programar está de moda. Y más aún si tienes 10 o 12 años. Pero no lo han puesto de moda ellos. Está de moda entre aquellos padres que ven la programación como una herramienta más con la que equipar a sus hijos. También está de moda entre las escuelas de informática que ven ahí una manera de estimular la cantera.

También está de moda animar a los chicos a programar entre las empresas tecnológicas que notan la falta profesionales formados en programación ya hoy, pero aún más en el futuro. Y también está de moda entre pequeñas y grandes empresas cuyo objetivo final es cubrir esa demanda y enseñar a los chicos a programar.

En la Universidad de Valladolid, el grupo de Computación Educativa CompuEdu@UVa lleva trabajando desde 2011 en la enseñanza de la programación en niveles preuniversitarios. Gracias a los años de experiencia acumulados en la formación universitaria en carreras técnicas, éramos muy conscientes de las dificultades que aprender a programar supone tanto para los alumnos de Ingeniería Industrial, como para los de Telecomunicaciones e, incluso, para los de Informática. Quizás este background sea el motivo principal de que CompuEdu no tenga en su punto de mira la programación como objetivo sino como herramienta. Si bien consideramos válidas las motivaciones del resto de agentes para animar a los chavales a aprender a programar, nuestra motivación es diferente y, quizás por ello, nuestro modelo didáctico también lo es.

De entre todos los lenguajes de programación y entornos que se proponen para introducir en la programación a los más pequeños, el que más nos ha gustado y con el que más tiempo llevamos trabajando es Scratch, que es un programa desarrollado por el Instituto de Tecnología de Massachussets para introducir a la programación de ordenadores a chavales a partir de los 8 años. La filosofía de Scratch es muy abierta y pretende eliminar las barreras para entrar en el mundo de la programación de los más pequeños. Por una parte, es de uso libre y puede instalarse en ordenadores con requisitos mínimos. Por otra, no permite errores sintácticos puesto que se construye como un puzle de piezas predefinidas. Además, los resultados de cada pequeño avance en el código son inmediatamente ejecutables haciendo la tradicionalmente tediosa tarea del *debugging* mucho más sencilla y acostumbrando a los chicos a comprobar todas las partes del código que escriben.

A pesar de su facilidad de uso, Scratch es una herramienta muy potente y los más de 18 millones de proyectos que hay a día de hoy en el repositorio del MIT así lo avalan.

Desde el punto de vista de un niño, Scratch permite crear desde un sencillo ping-pong en menos de una hora de trabajo, hasta una elaborada animación con objetos modelados en 3D con plastilina. Desde el punto de vista del docente, Scratch permite desarrollar el pensamiento computacional, el pensamiento lógico, las habilidades matemáticas y la resolución de problemas. Todo ello, sumado a los ya conocidos efectos del antecesor de Scratch, el lenguaje de programación Logo, en cuanto a mejoras en la verbalización o el aumento en la tolerancia a la frustración.

Todos los aspectos arriba mencionados son esencialmente nuestros objetivos y se corresponden con la filosofía didáctica de CompuEdu@UVa. El desarrollo de estas competencias es nuestra motivación para introducir a formadores y a niños en la programación de ordenadores y, en particular, explica nuestra elección del lenguaje Scratch y de las metodologías didácticas basadas en la provocación, la pregunta y el reto, más que en el guiado o la exposición de contenidos.

2. El lenguaje de programación Scratch

Scratch consta de una interfaz gráfica con cuatro zonas diferenciadas: el "escenario" en el que se ejecutan los programas; los objetos o *sprites* que realizan las acciones; la zona donde se escribe el código para cada uno de esos objetos; y la zona de instrucciones, donde encontramos todas las piezas con las que podemos escribir nuestro programa apilándolas como si de de un lego se tratase. En la Figura 1 podemos ver la interfaz del editor de Scratch 2.0 con las 4 zonas indicadas.

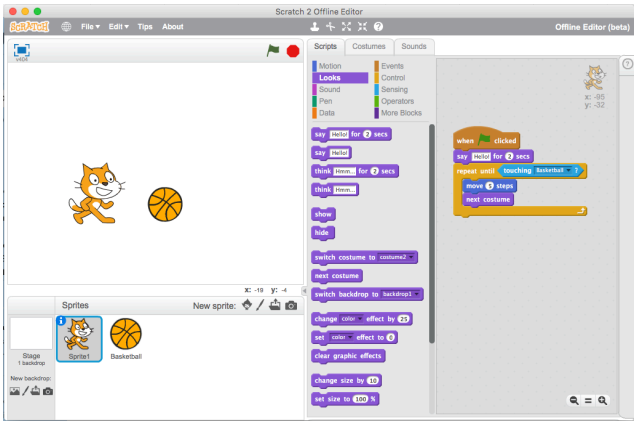

Figura 1. Arriba a la izquierda, escenario con balón y gato. Abajo a la izquierda, objetos o sprites que intervienen en el programa. Columna central, piezas con instrucciones. Columna derecha, programa a realizar por el sprite gato cuando se pulse la bandera verde.

Técnicamente, y desde el punto de vista de cuánta programación se puede llegar a aprender con Scratch, mostramos a continuación algunas de las piezas esenciales de Scratch con ejemplos simples de su uso. Las piezas están organizadas por temas en diferentes pestañas codificadas por colores.

El lenguaje Scratch incluye las tres estructuras de control y el cuidado diseño de las piezas con las que se construyen los programas hace que la construcción de bloques resulte muy intuitivo. Para facilitar la comprensión de la secuencia, cada pieza incluye unas pequeñas muescas en las partes superior e inferior que permite encajar unas piezas con otras. Durante la elaboración de un programa, la interfaz "propone" el encaje de las piezas iluminando conforme acercamos una pieza a otra las posibles ubicaciones de aquella que arrastramos. Tanto los bucles como las selecciones están en la pestaña de "Control" y son identificadas con el color naranja. Vienen definidos gráficamente el inicio y el fin de bloque, de manera que las piezas que controlan un bucle o una selección "abrazan" al código que queremos ejecutar en su interior. En la Figura 2 podemos ver tres ejemplos de piezas de control donde los bucles añaden una discreta flecha que facilita la comprensión de la vuelta atrás o repetición del código, mientras que los condicionales únicamente separan visualmente los diferentes bloques que intervienen. Por supuesto, todas ellas son anidables como se puede observar en las muescas que facilitan la visualización del encaje de piezas.

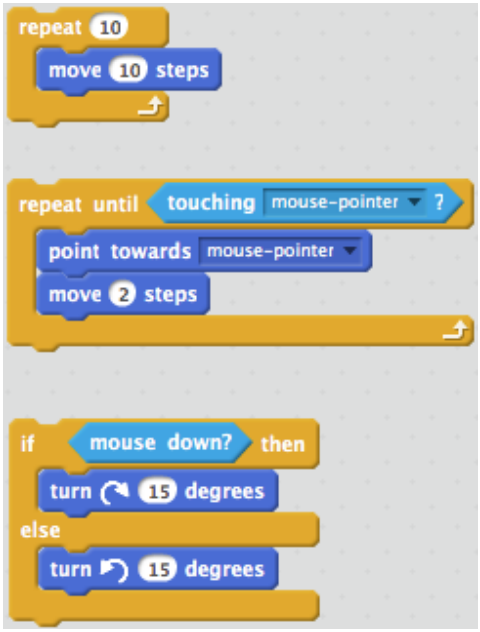

Figura 2. Dos ejemplos de bucles, uno por contador y otro por condición, y un ejemplo de construcción de un condicional doble.

A diferencia de la programación con otros lenguajes, Scratch permite realizar infinidad de programas sin necesidad de declarar una sola variable, concepto que sabemos por experiencia que no es evidente incluso a niveles universitarios. Existe una pestaña específica para el trabajo con variables (en color naranja oscuro), donde se evitan muy inteligentemente expresiones del tipo i=i+1, que todos los que alguna vez hemos impartido programación en niveles iniciales sabemos que puede causar confusión. En su lugar, las piezas para trabajar con variables se limitan a "fijar" para hacer un *set* y a

"cambiar por" como una lamentable traducción de *change by* que admite como parámetro en cuánto se incrementa o decrementa la variable indicada.

Figura 3. Piezas *set to* y *change by* de una variable

Con la filosofía de hacer sencillos los primeros pasos en programación pero no limitar hasta dónde se puede llegar con Scratch, también nos permite en la pestaña de "Datos" crear arrays. Nótese en el ejemplo de la Figura 4 que no hay un tipado de datos y que admite cadenas de caracteres como elementos de una lista, de la misma manera que podríamos hacer la lista de enteros o cualquier otro tipo de dato.

Figura 4. Ejemplo de construcción de un array de cadenas de caracteres y de uso de la pieza de búsqueda de un elemento dentro de un array.

De cara a la construcción de programas más extensos y flexibles, la versión 2.0 de Scratch incluyó la posibilidad de definir módulos procedimentales (no de función) que, incluso pueden recibir uno o más parámetros de entrada como muestra la Figura 4.

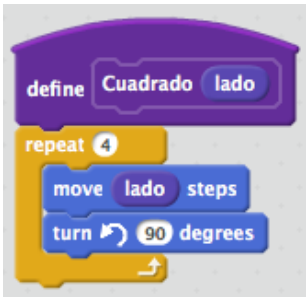

Figura 4. Ejemplo de un módulo con un parámetro de entrada entero para que el actor se mueva siguiendo un cuadrado del lado indicado por el parámetro.

Scratch está dirigido a eventos lo que, curiosamente, resulta muy intuitivo a los niños. Esta orientación a eventos les facilita el trabajo con cada uno de sus personajes y la coordinación entre ellos, siendo capaces los niños de 8 o 9 años de sincronizar a los actores de su animación mediante eventos en una sesión inicial de toma de contacto con Scratch en tan solo una o dos horas.

Figura 5. Ejemplo del código de un sprite que recibe un evento "disparado", comprueba el contenido de la variable con el número de vidas y envía el evento "gameOver" si no le quedan más vidas al personaje.

La versión 2.0 incluye también el concepto de "clon" que, si bien resulta demasiado complejo en los niveles iniciales, es de gran utilidad y una subida en el nivel de abstracción muy interesante para introducir a chavales algo mayores o con más experiencia. Como su propio nombre indica, un clon de un sprite será una copia del mismo que ejecuta el código indicado a partir de la pieza "when I start as a clone" y que puede autodestruirse con la pieza "delete this clon". El nivel de abstracción que exige el tratamiento de los clones es un ejercicio claro de desarrollo del pensamiento computacional, lógico y matemático.

Figura 6. Ejemplo de código para un sprite que, al pulsar la barra espaciadora, se clona y se desplaza hasta tocar el borde de la pantalla. Es interesante observar cómo la pieza *delete this clone* ya no incluye la muesca inferior para encajar nuevas piezas de código a continuación.

3. Modelo didáctico

Como hemos podido ver en la sección anterior, Scratch es un completo lenguaje de programación que permite una exploración tanto en anchura como en profundidad de qué significa programar. Nuestra experiencia es que el paso de Scratch a lenguajes de texto como podría ser Python es muy sencillo gracias a la imagen mental que Scratch

permite crear, haciendo que los chavales puedan pasar gradualmente del concepto visual de, por ejemplo, un bucle que engloba una serie de acciones que se deben repetir, a una indentación particular, unas llaves que abren y cierran, o cualquier otro modo de representación más abstracta que el lenguaje textual al que se enfrenten a continuación necesite.

En estos años de experiencia docente tanto con adultos como con niños, hemos podido ir elaborando nuestro propio modelo didáctico sobre cómo introducir la programación y, en particular, la programación con Scratch. Es muy importante preguntarnos antes de comenzar para qué queremos enseñar a programar. Según cuál sea nuestra respuesta a esta pregunta, variará el cómo queremos enseñar a programar y, por supuesto, qué vamos a enseñar.

Como ya hemos indicado anteriormente, CompuEdu@UVa busca en la enseñanza de la programación dar una nueva manera de expresión a los chavales, al tiempo que adquieren una serie de competencias transversales de gran importancia en su futuro. Cuando hablamos entonces de para qué programar, nosotros nos orientamos al desarrollo de la creatividad, del pensamiento matemático, lógico y computacional, de la capacidad de abstracción, o de la resolución de problemas. También buscamos efectos "colaterales" como el aumento de la tolerancia a la frustración, la persistencia en la tarea o la visión crítica de la tecnología, dado que los situamos en el papel del creador y no del consumidor, haciéndoles plantearse cómo todo avance tecnológico proviene de una mano humana que decide qué y cómo.

Con las premisas anteriores, buscamos trabajar desde metodologías dinámicas y abiertas donde los chavales sean los protagonistas de su propio aprendizaje. Evitamos el uso de pizarras, limitamos las exposiciones de contenidos y no proporcionamos enunciados cerrados. Por el contrario, realizamos propuestas abiertas que puedan motivar diferentes desarrollos por cada uno de los niños, admitiendo que cada uno a su nivel llegará un paso más allá de donde se encontraba. Por citar un ejemplo de cómo buscamos atender a la diversidad que tenemos en las aulas, hace algunos años dimos a un grupo de unos 20 chicos una propuesta de realizar un modelo humano que saludase con el brazo al presionar una tecla. Los alumnos que se estaban iniciando, dieron una respuesta basada en realizar varios dibujos de un personaje variando la posición del brazo, de manera que al visualizarlas rápidamente en secuencia, el personaje parecía saludar. Otros alumnos atacaron el problema creando un sprite para cada parte del cuerpo que les permitiera animar el brazo con independencia del resto del cuerpo. Así, el brazo se anclaba al hombro y tomaba este como centro para realizar el giro y simular el saludo. El grupo más avanzado pudo enfrentarse a una construcción de brazo y antebrazo donde el movimiento coordinado con respecto al hombro y al codo exigían una elaboración mucho mayor. También en el aspecto de los proyectos procuramos ser muy flexibles, dado que hay perfiles de niños cuyo mayor interés está en la parte de la programación, así como hay otros que se inclinan por la parte del diseño y realmente se preocupan por el acabado del producto. Desde escenarios en blanco o con un garabato, hasta sprites diseñados pixel a pixel, cada uno puede utilizar el proyecto como un modo de expresión con el que nosotros cumplimos nuestro objetivo de darles voz en ese nuevo lenguaje y fomentar su creatividad.

La elaboración de propuestas abiertas que admitan diferentes niveles de consecución no es una tarea fácil. Una vez decidido que queremos que nuestros alumnos aprendan a utilizar, por ejemplo las variables, debemos proporcionar un contexto donde las variables sean absolutamente imprescindibles, por lo que debemos explorar qué caminos podrían encontrar como alternativa a esa creación. Podría contar aquella vez en que propuse un ejercicio donde debían contar las vueltas que daba un coche a un circuito antes de salirse de la carretera. Dado que el circuito era largo, nadie conseguía dar más allá de dos o tres vueltas. Tuve que aceptar una solución donde un sprite cambiaba de aspecto en cada vuelta siendo en la primera vuelta un cero, en la segunda un uno y así hasta 9 disfraces. Aprendí que para contar las vidas no son imprescindibles las variables y, desde entonces, damos o quitamos puntos a los personajes según la acción del juego procurando utilizar cifras grandes.

Quizás parezca contradictorio para aquellos docentes que aún no hayan probado las metodologías activas que, a mayor autonomía para los alumnos, mayor es el nivel que alcanzan. Habitualmente, tendemos a pensar en "el niño de 9 años hace", sin pensar en lo que "el niño de 9 años puede llegar a hacer". La metodología que seguimos cuando trabajamos con niños constantemente nos hace sorprendernos por haber minusvalorado hasta dónde podrían los chicos llegar. Es evidente que esto hace que, en ocasiones, nos pongan en auténticos apuros porque nos exigen mayor dominio tanto de programación como de Scratch con sus dudas. Por otra parte, cuando ellos mismos resuelven sus problemas y nos presentan soluciones inesperadas, el aprendizaje se invierte y la admiración se hace mutua.

Si bien hay muchas iniciativas conectando la programación con el mundo físico (mediante la robótica, la electrónica, la impresión 3D, etc), nuestra experiencia es que la creación de programas sin conexión con el mundo físico les permite desarrollar en mayor medida una de nuestras variables objetivo: la creatividad. La programación de ordenadores es una manera de dar a los chavales un nuevo idioma con el que expresarse. Ya sea por medio de un determinado videojuego, ya sea una animación, o un diseño de un personaje, nos encontramos constantemente con niños que nunca se han sentado a pintar con unos lápices y un papel, o aquellos que jamás se sentarían a escribir una redacción, que se enfrentan al proceso creativo delante del ordenador con otra energía porque este idioma sí les ha permitido plasmar sus creaciones.

4. CONCLUSIONES

Si la selección de propuestas es suficientemente abierta, la programación con Scratch puede convertirse en un gran aliado en los niveles de Primaria y Secundaria tanto en el aula como en casa para el desarrollo de habilidades transversales a la programación. Afortunadamente, Scratch está suficientemente bien desarrollado como para permitir también la enseñanza de la programación como un objetivo en si mismo y, de hecho, animo a los docentes de las asignaturas de programación en el primer curso de universidad a que prueben a utilizarlo durante dos o tres semanas al inicio del curso. Ya se han realizado experiencias piloto en universidades comprobando que, incluso a los 18, facilita la introducción a la programación proporcionando un modelo visual previo al modelo abstracto.

En la Universidad de Valladolid, tanto en el Campus de Valladolid como en el de Segovia, CompuEdu@UVa dirige los Clubes de Jóvenes Programadores para chavales de 8 años en adelante, y participa anualmente en una jornada mundial de difusión conocida como el Día del Scratch (Scratch-Day). Tanto para estas actividades como para las muchas otras en las que participamos como la formación de formadores, la asistencia a maestros y profesores de Primaria y Secundaria y otras jornadas de formación y difusión, siempre agradecemos el apoyo de estudiantes y profesores que se quieran embarcar en este precioso proyecto. Cualquier persona interesada en colaborar con nosotros, puede contactar por email escribiendo scratch@infor.uva.es, seguirnos en twitter en @Scratch_UVa o en nuestra web http://scratch.infor.uva.es

BIOGRAFÍAS

Belén Palop es Doctor en Informática por la Universidad Politècnica de Catalunya. Después de más de 15 años como profesora universitaria en el área de Lenguajes y Sistemas Informáticos, desde Septiembre de 2015 pertenece al área de Didáctica de la Matemática. Dedicada en los últimos años a la divulgación de la enseñanza de la programación y al trabajo con maestros y niños, investiga actualmente en la búsqueda de metodologías que permitan mejorar la competencia matemática a partir de enfoques integradores como a través de la programación de ordenadores, las ciencias naturales o la ingeniería.