MEDI@ÇÕES
Revista OnLine

EDUCAÇAO
ESCOLA SUPERIOR
POLITÉCNICO SETÚBAL

Un prototipo de rastreador para la formación del profesorado en el lenguaje ScratchJr

J. ÁNGEL VELÁZQUEZ-ITURBIDE
angel.velazquez@urjc.es
DAVID DE VICENTE PEÑA
d.devicente.2018@alumnos.urjc.es
EVA LÓPEZ PUENTE
e.lopezpu@alumnos.urjc.es
Universidad Rey Juan Carlos

Resumen

Los futuros maestros de Educación Infantil deberán tener competencias básicas en programación basada en bloques, pero sus entornos no incluyen mecanismos de control de la ejecución. Se presenta una ampliación del entorno de programación de ScratchJr con varias facilidades: mostrar los valores de los atributos de los personajes, controlar el avance de la ejecución, mostrar el número de iteración en los bucles con contador y mostrar simultáneamente los guiones en ejecución de todos los personajes.

Palabras clave:

Educación Infantil, lenguajes basados en bloques, ScratchJr, formación del profesorado, semántica operacional, rastreador.

Abstract

Future teachers in Early Childhood Education should have basic competences in block-based programming, but the available programming environments do not include mechanisms to control program execution. In the paper, we present an extension of the ScratchJr programming environment with several facilities: displaying the values of sprite attributes, controling execution advance, displaying the iteration number in counter loops, and displaying simultaneously the execution of the sprites' scripts.

Key concepts:

Early childhood education, block-based programming languages, ScartchJr, teacher development, operational semantics, tracer.

Introdución

En la última década se ha asentado la programación basada en bloques como un paradigma a tener en cuenta en la enseñanza de la programación (Bau et al., 2017). Aunque existen experiencias en la universidad (Martínez-Valdés *et al.*, 2017), su uso se concentra principalmente en las etapas preuniversitarias, es decir, Educación Infantil (González-González, 2019; Macrides *et al.*, 2022), Primaria (Fagerlund *et al.*, 2020) y Secundaria (Loockwood & Mooney, 2018).

Uno de los problemas principales para la introducción de la informática en general y la programación en particular en el currículo escolar es la falta de formación del profesorado (Caspersen *et al.*, 2018). Es una carencia que no tiene una solución clara a corto plazo, pero que debe afrontarse pensando a medio y largo plazo. La solución dependerá de las peculiaridades de cada país en el acceso a la profesión docente.

En España, los profesores de Educación Infantil y Primaria se forman en los grados respectivos (Velázquez-Iturbide *et al.*, 2023). Asimismo, los profesores de Educación Secundaria deben cursar el Máster Universitario en Formación del Profesorado de Educación Secundaria, Bachillerato, FP e Idiomas. Normalmente, dicho máster ofrece un itinerario en Tecnología, pero pocos lo ofrecen en Informática o en Tecnología e Informática.

Un profesor bien formado para enseñar cierta materia debe tener conocimientos de la misma, así como conocimientos de su didáctica e instrumentación (Koehler & Mishra, 2009). El conocimiento del contenido suele darse por supuesto en cualquier profesor, pero no siempre es así. Por ejemplo, un profesor que debe sustituir a otro por una baja puede tener lagunas en su conocimiento de la nueva materia a impartir. En el caso de la informática, esta situación no es anecdótica, sino generalizada. Al ser una materia que no se incluye en los grados de Educación ni suele incluirse en el Máster Universitario de Formación del Profesorado, el conocimiento de informática y programación suele ser nulo o, con suerte, superficial.

El aprendizaje de la programación no es sencillo, como ha quedado constatado en innumerables investigaciones; véase, por ejemplo, las revisiones (Gomes & Mendes, 2007; Luxton-Reilly *et al.*, 2018; Robins, 2019). Es cierto que la programación basada en bloques facilita el aprendizaje de la programación porque elimina algunas de sus dificultades, como la memorización de las construcciones del lenguaje o la comprensión de los mensajes de compilación.

Sin embargo, persisten otras dificultades, como comprender la dinámica de la ejecución de los programas, necesaria para depurar programas con errores. La dinámica de los programas se explica mediante modelos conceptuales explícitos o implícitos, frecuentemente llamados "máquinas nocionales" (du Boulay *et al.*, 1981; Sorva, 2013). Actualmente, no existe ninguna máquina nocional claramente especificada para lenguajes basados en bloques (Seppälä *et al.*, 2019).

En esta comunicación abordamos el desarrollo de herramientas de programación que faciliten a los profesores el aprendizaje de la dinámica de los lenguajes basados en bloques. En concreto, se presenta una ampliación del entorno de programación de ScratchJr para facilitar el rastreo (*trace*) de la ejecución de sus programas, de forma parecida a los depuradores de lenguajes textuales. Dicha ampliación permite mostrar el estado de un programa en ejecución y controlar su avance.

La estructura de la comunicación es la siguiente. En la sección primera se presentan las características principales del lenguaje ScratchJr. La sección segunda presenta las ampliaciones realizadas del entorno de programación de ScratchJr para integrar funciones de rastreo propias de un depurador. Finalmente, presentamos nuestras conclusiones y trabajos futuros.

1. ScratchJr

ScratchJr (Bers & Resnick, 2016) es un lenguaje de programación visual derivado del popular lenguaje Scratch (Resnick *et al.*, 2009). ScratchJr se diseñó para introducir en la programación a niños de 5 a 7 años de edad, por lo que permite programar sin necesidad de saber leer o contar. En esta sección veremos una breve introducción al lenguaje y su entorno. Puede encontrarse más información en su página web (https://www.scratchjr.org/).

1.1. Programas ScratchJr

La Fig. 1 muestra la pantalla principal de ScratchJr. Podemos identificar varias zonas:

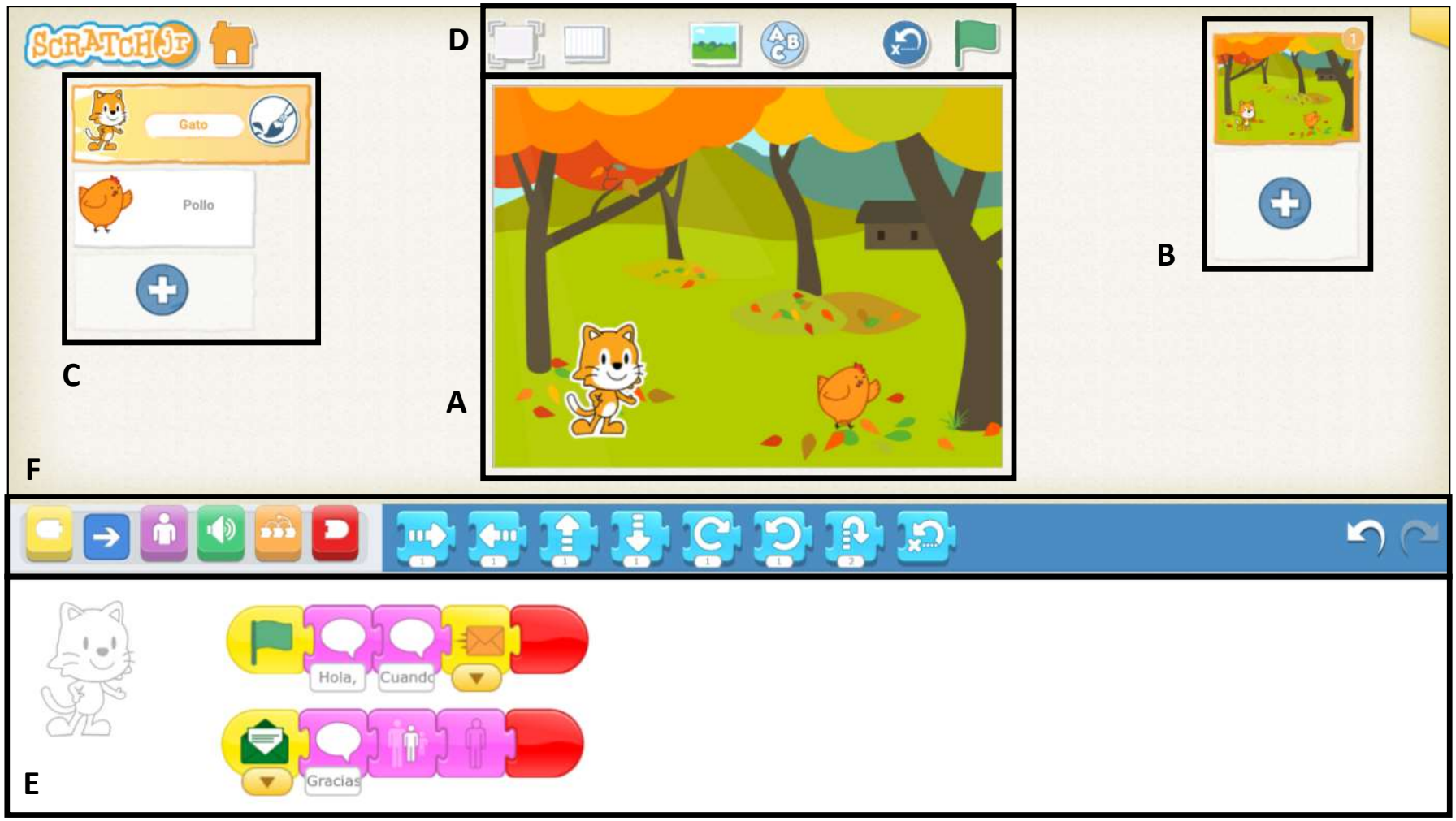

Fig. 1. Pantalla principal de ScratchJr

A. Escenario. Está situado en la parte central de la pantalla y es el área donde transcurre la acción. En la figura, pueden verse un gato y un pollo en una dehesa.
B. Páginas. Esta zona está situada en la parte superior derecha y muestra los distintos fondos en los que transcurrirá la acción del programa. En la figura se muestra la única página existente (el paisaje) más la posibilidad de añadir otras páginas (signo más).
C. Personajes. Esta zona está situada en la parte superior izquierda y muestra los personajes de la página actual del programa. La etiqueta del personaje que se encuentra seleccionado aparece agrandada y resaltada en naranja (en la figura, el gato).
D. Controles. Encima del escenario aparecen varios controles, entre ellos la bandera verde que se utiliza para iniciar la ejecución de un programa.
E. Zona de programación. En esta zona, situada en la parte inferior de la pantalla, se construye el programa del personaje seleccionado. Su programa puede estar formado por varias secuencias de bloques ("guiones" o *scripts*) que pueden ejecutarse en paralelo. Obsérvese en la figura que los bloques de un guion se disponen en horizontal, de izquierda a derecha.
F. Categorías y paleta de bloques. El escenario y la zona de programación están separadas por una franja horizontal. En su parte izquierda aparecen seis iconos que representan las seis categorías de bloques. A su derecha, aparecen los bloques disponibles de la categoría seleccionada. En la figura, se muestran los ocho bloques de la categoría de movimiento (de color azul).

Un programa está formado por los guiones de los personajes de todas las páginas. Un personaje tiene cero, uno o varios guiones (véase el gato de la Fig. 1, con dos guiones). Cada guion se crea arrastrando bloques desde la paleta de bloques a la zona de programación y particularizando los parámetros del bloque, si los tiene.

Existen seis categorías de bloques, cada una distinguible mediante un color:

- Bloques de movimiento (de color azul).
- Bloques de apariencia (de color rosa).
- Bloques de sonido (de color verde).
- Bloques disparadores (de color amarillo).
- Bloques de control (de color naranja).
- Bloques de finalización (de color rojo).

Las dos primeras categorías permiten mover o cambiar el aspecto de

los personajes. La tercera categoría permite reproducir sonidos. Las restantes tres categorías contienen bloques de control, que deben colocarse en partes distintas de un programa ScratchJr: los bloques disparadores se colocan al comienzo de un guion, los bloques de finalización se colocan al final y los bloques de control, en medio. La forma de cada tipo de bloques está adaptada a este fin.

1.2. Ejecución de programas ScratchJr

Un programa ScratchJr comienza a ejecutarse cuando se produce un evento que es capturado por uno o varios guiones de la primera página. Hay cuatro tipos de evento:

- Al presionar la bandera verde.
- Al pulsar el personaje.
- Al recibir un mensaje.
- Al tocar el personaje.

Normalmente, la ejecución de un programa comienza cuando el usuario pulsa el icono de bandera verde. Una vez que se pulsa la bandera verde, su icono es sustituido por un icono de hexágono rojo, que podrá pulsarse para parar la ejecución. En ese momento, se restaura el icono de bandera verde.

La ejecución comienza en la primera página. Puede cambiarse de página cuando se ejecuta el bloque de finalización "Ir a página". Este bloque produce un cambio de fondo en el escenario, la desaparición de todos los personajes de la página anterior y la aparición de los personajes de la nueva página. Obviamente, se para la ejecución de los guiones activos de los personajes de la página anterior y se inician los guiones de los nuevos personajes que tengan un bloque disparador "Al presionar bandera verde". Por tanto, las páginas delimitan el ámbito de fondos, personajes y guiones.

Los bloques de un guion se ejecutan secuencialmente, comenzando por el situado a la izquierda (un bloque disparador) y terminando por el último a la derecha (normalmente, un bloque de finalización). En cada momento, se está ejecutando un solo bloque. Cuando termina su ejecución, continúa ejecutándose el siguiente, situado a su derecha.

Dentro de una página, puede haber varios guiones ejecutándose en paralelo. Cada guion inicia su ejecución cuando se produce el evento indicado por su bloque disparador. Por tanto, el número de guiones que se ejecutan en paralelo en un programa ScratchJr puede variar en el tiempo. Si suponemos que el programa se ejecuta en un solo procesador, esto implica que los bloques de los distintos guiones activos se

ejecutan de alguna forma alterna y equitativa.

Los bloques de las categorías de movimiento, apariencia y sonido son sencillos de comprender, ya que su efecto en los personajes situados en el escenario es visible (en los dos primeros casos) o audible (en el tercero). Son más complejos de comprender los bloques de control, repartidos entre las categorías de bloques disparadores (basadas en eventos, ya examinados), de control y terminadores (que incluye el bloque "Ir a página", entre otros).

Los bloques de control permiten cambiar la ejecución secuencial de un guion. El bloque terminador "Repetir indefinidamente" introduce todo el guion en un bucle infinito. El bloque "Repetir" es un bucle con contador, es decir, hace que los bloques que agrupa se repitan tantas veces como indica su número, que es modificable por el usuario. La ejecución del bloque "Parar" en un guion de un personaje produce la interrupción de los demás guiones del personaje. Finalmente, tenemos dos bloques de control que permiten actuar sobre el comportamiento temporal de los personajes. El bloque "Esperar" provoca que la ejecución de un guion se pare durante tantas décimas de segundo como indica su número. El bloque "Fijar velocidad" permite variar la velocidad de los movimientos de un personaje.

En total, el lenguaje incluye 28 bloques distintos, aunque algunos son parametrizables mediante colores, números o sonidos. Aunque muchos bloques tienen un comportamiento sencillo de comprender, otros son más complejos o incluso tienen comportamientos imprevistos en algunas situaciones (Velázquez-Iturbide, 2021). Por tanto, un rastreador puede mejorar la comprensión de la ejecución de los programas ScratchJr, sobre todo cuando su comportamiento no es el previsto por el programador.

1.3. Implementación de ScratchJr

El lenguaje ScratchJr está implementado en JavaScript. Para este trabajo, se dispuso de una implementación no "oficial" de ScratchJr (https://github.com/jfo8000/ScratchJr-Desktop/). Su uso presentaba dos ventajas: por un lado, disponer de una implementación abierta sobre la que desarrollar el rastreador, y por otro, que permitía generar el entorno para plataformas Windows (la versión oficial solamente está disponible para dispositivos móviles). En concreto, esto último facilitará su uso en la docencia reglada de formación del profesorado.

2. Un prototipo de rastreador de ScratchJr

En esta sección presentamos las decisiones de diseño y las ampliaciones realizadas del entorno de ScratchJr para desarrollar el prototipo de rastreador.

2.1. Diseño del rastreador de ScratchJr

La extensión del entorno de programación de ScratchJr para albergar un rastreador tuvo en cuenta diversas consideraciones de diseño, de las que más más relevantes son las propias funciones del rastreador y la interfaz de usuario. Veamos ambas brevemente.

El diseño del propio rastreador depende de dos cuestiones principales. Por un lado, debe determinarse la información que constituye el estado de un programa, tanto para visualizarlo como para usarlo en el control del avance. Esta información se había determinado experimentalmente (Velázquez-Iturbide, 2021), pero se corroboró analizando la implementación del lenguaje.

Por otro lado, están las propias facilidades de avance de la ejecución, que requieren cierto control del estado de ejecución del programa. Se tomó como fuente de inspiración las facilidades de rastreo proporcionadas por los entornos de programación para lenguajes textuales, incluso de programación funcional (Pareja Flores *et al.*, 2007). Se identificaron varios modos de avance equivalentes a otros existentes. Asimismo, se constató la ausencia de información sobre el estado de ejecución de los bucles “Repetir”, es decir, el número de iteraciones ya realizadas en un momento dado. Por último, se vio la conveniencia de examinar la ejecución simultánea de guiones dentro de cada página.

En cuanto a la interfaz de usuario, se cuidó que los cambios producidos en la misma fueran coherentes, dentro de lo posible, con su diseño original. Esto se refleja en el diseño o selección de los diversos iconos y colores.

Los atributos de los personajes han presentado una dificultad añadida. ScratchJr está pensado para niños pequeños, probablemente prelectores (Flannery *et al.*, 2013). Por esta razón, su interfaz prescinde casi completamente de texto. En un primer momento, se pretendió realizar algo parecido con los atributos. Sin embargo, sin un texto era difícil distinguir los atributos y sus valores. Dado que el rastreador está concebido para profesores, finalmente se decidió poner el nombre cada atributo. Por coherencia, el rastreador los muestra en el idioma activo (ScratchJr soporta 12 idiomas).

Tanto las funciones como la interfaz de usuario se diseñaron y desa-

rrollaron de forma incremental, mediante prototipado sucesivo. El primer autor fue el encargado de realizar su diseño, el segundo autor de su implementación y el tercero, de su revisión. En todo caso, los tres autores realizaron reuniones periódicas, en las que se analizaron alternativas y las sucesivas versiones del prototipo.

2.2. Atributos de un personaje

En ScratchJr no existen variables. Sin embargo, cada personaje tiene ciertos atributos cuyos valores son parte del estado del programa, junto con información de control de su ejecución –página activa, guiones activos, etc. (Velázquez-Iturbide, 2021)–.

Una primera ampliación fue hacer visibles los valores de los atributos de cada personaje, ya que algunos eran fácilmente determinables de forma visual, pero otros no. La Fig. 2 muestra que cada personaje tiene 8 atributos, cuyo valor puede consultarse entre la zona de personajes y el escenario, para que la consulta visual sea rápida (el escenario no se muestra en la figura). En cada momento, se presentan solamente los valores de los atributos del personaje activo (en la figura, el personaje de color morado llamado Te). Obsérvese que, cuando un atributo sólo toma dos valores, éstos se representan mediante iconos (atributos orientación, visibilidad y diálogo).

Fig. 2. Pantalla principal de ScratchJr

Los atributos mostrados son los siguientes. Algunos valores son los que tiene internamente el procesador de ScratchJr, mientras que otros se han adaptado para que sean comprensibles por el usuario.

- Posición *y*, es decir, la posición vertical del personaje. Se utiliza el criterio espacial de ScratchJr, con el escenario dividido en filas numeradas, de abajo a arriba, de 1 a 15.

- Posición *x*, es decir, su posición horizontal. Análogamente, las columnas están numeradas de izquierda a derecha de 1 a 20.
- Orientación, a la derecha o a la izquierda.
- Ángulo, comprendido entre 0 y 359 grados.
- Visibilidad, es decir, si el personaje es visible o no.
- Escala, cuyo valor se adapta a cada personaje, de forma que recoge su tamaño relativo entre el mínimo y el máximo posible (1% y 100%, respectivamente).
- Diálogo, que indica si el personaje está hablando.
- Velocidad del personaje, con valores 25, 50 y 100, correspondientes a velocidades lenta, media y rápida, respectivamente.

Los campos de atributo no sólo muestran sus valores, sino que permiten modificarlos. La interacción varía según el atributo:

- Posición *y*, posición *x*. Deben modificarse manipulando la posición del personaje directamente en el escenario o ejecutando bloques de movimiento.
- Orientación, visibilidad. Al ser campos booleanos, cambian de un valor al contrario si se hace clic directamente sobre el campo correspondiente.
- Ángulo, escala, velocidad. En la Fig. 2 puede observarse que existen dos controles a la derecha de estos campos, uno para aumentar su valor y otro para disminuirlo. Para los tres atributos, las variaciones coinciden con las que se obtendrían con el bloque correspondiente. Por ejemplo, un tic de aumento o disminución sobre el atributo de ángulo produce un giro a derecha o a izquierda de 30º, respectivamente.
- Diálogo. Sólo puede modificarse mediante la ejecución de un bloque “Decir”.

2.3. Bloque “Repetir”

Un bloque de control destacado es el bloque “Repetir”. Durante su ejecución, no se sabe cuántas iteraciones lleva realizadas. Para que se conozca el estado de ejecución del bloque “Repetir”, se hace visible un contador interno cuando comienza la ejecución del guion en el que se encuentra.

La Fig. 3 muestra los sucesivos estados de la ejecución paso a paso de un guion sencillo consistente en un paso a la derecha seguido de un bucle que repite dos veces un bloque “Crecer”.

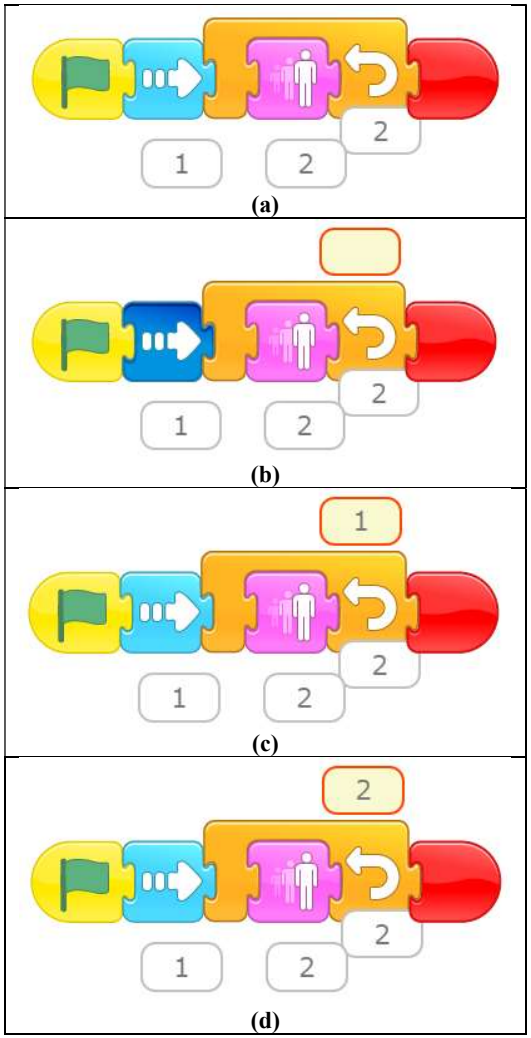

Fig. 3. Estados sucesivos de la ejecución de un bucle con dos iteraciones

Veamos su explicación paso a paso:

- Figura 3(a): muestra el guion antes de iniciar la ejecución.
- Figura 3(b): se ejecuta y resalta el primer bloque, "Mover a la derecha". Dado que el guion contiene un bloque "Repetir", también aparece el contador del bucle, sin tomar ningún valor todavía.
- Figura 3(c): se ejecuta la primera iteración del bucle, ya que engloba un solo bloque, "Aumentar".
- Figura 3(d): se ejecuta la segunda iteración del bucle.

Si avanzáramos un paso más, saldría del bucle y terminaría la ejecución del guion. El guion volvería a presentar el mismo aspecto que en la Fig. 3(a).

2.4. Modos de avance

Recordemos que los únicos controles disponibles en ScratchJr para controlar la ejecución de un programa son el icono de bandera verde para comenzar y el de hexágono rojo para terminar. La Fig. 4 muestra cinco controles adicionales, que se han situado junto a la bandera verde (y el hexágono rojo).

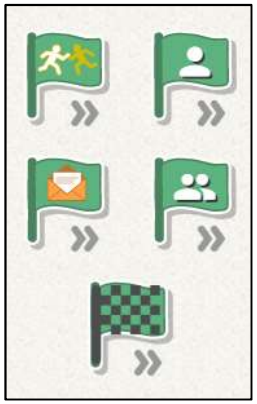

Fig. 4. Estados sucesivos de la ejecución de un bucle con dos iteraciones

Veamos la función de cada control nuevo, de arriba abajo y de izquierda a derecha en la Fig. 4:

- Avanza hasta chocar. La ejecución del programa avanza hasta que un bloque disparador detecta un evento de “Al tocar al personaje”, es decir, cuando dos personajes chocan.
- Avanza un paso. El programa del personaje activo avanza un paso, es decir, se ejecuta un bloque de cada guion activo de dicho personaje.
- Avanza hasta mensaje. La ejecución del programa avanza hasta que un bloque disparador detecta un evento de “Al recibir mensaje”.
- Avanzan todos un paso. Los programas de todos los personajes avanzan un paso.
- Avanza hasta el final. Este icono no aparece inicialmente, pero se hace visible cuando el usuario presiona alguno de los otros iconos. Su uso supone que la ejecución deja de estar controlada por el usuario.

Se mantiene el efecto visual de ScratchJr consistente en que el bloque que se acaba de ejecutar se resalta con un tono más oscuro. Esto facilita su identificación.

La introducción de los controles de avance se ha realizado respetando la semántica operacional del lenguaje, es decir, no se ha alterado la ejecución de ningún bloque de ScratchJr. Sin embargo, una curiosa irregularidad de ScratchJr es que algunos bloques se ejecutan pero nunca se resaltan sus bloques, dando la impresión de que no consumen tiempo de ejecución. Es el caso del bloque ”Repetir”, cuyo control no se tiene en cuenta en la planificación de la ejecución. Por tanto, durante un avance paso a paso, el bloque no se resalta ni su ejecución cuenta como un paso.

2.5. Pantalla de rastreo simultáneo

Las facilidades presentadas en los subapartados anteriores se han integrado en la pantalla principal de ScratchJr (Fig. 1), modificando ligeramente la ubicación de algunos elementos y añadiendo otros nuevos. También se ha añadido un icono que permite acceder a una pantalla en la que se muestran simultáneamente los programas de todos los personajes. De esta forma, se puede tener una visión global del avance de la ejecución del mismo.

La Fig. 5 muestra esta pantalla para un programa con tres personajes. Obsérvese que se mantienen todas las funciones de la pantalla principal, salvo la edición de programas. Ante la ejecución del programa, se muestra su avance en todos los guiones. En caso de que no puedan mostrarse todos los personajes en una sola pantalla (lo cual depende del tamaño de la misma), se activa una barra de desplazamiento

El usuario puede hacer activo a cualquier personaje con clicar en su zona de programa. Como consecuencia, sus guiones se muestran con colores no difuminados y se presentan los valores de sus atributos. En la figura, está seleccionado el personaje de color azul.

El icono X de la esquina superior izquierda permite volver a la pantalla principal y la flecha de la parte inferior permite que la ejecución salte a la página siguiente. Por tanto, proporciona otra función de control de la ejecución. Si hubiera una página anterior, aparecería una flecha parecida en la esquina inferior izquierda, pero orientada a la izquierda.

2.6. Limitaciones

Los avances permitidos por el rastreador son limitados, ya que sólo permite avanzar paso a paso o hasta que ocurre un evento (análogamente a los puntos de ruptura de los depuradores para lenguajes de programación textuales).

El rastreador permite controlar el avance de la ejecución, sin alterar el comportamiento global del programa, salvo en tres casos. Primero, mediante el avance repetido de un paso de un solo personaje. En esta situación, se altera el comportamiento del programa porque la ejecución de los demás personajes no avanza. Segundo, cuando varios personajes se mueven a distintas velocidades. En esta situación, la intervención humana interfiere ya que no permite mostrar el efecto de las distintas velocidades. Tercero, desde la pantalla con todos los guiones puede saltarse a la página anterior o siguiente. Esto produce un cambio en el flujo de la ejecución independiente del código del programa.

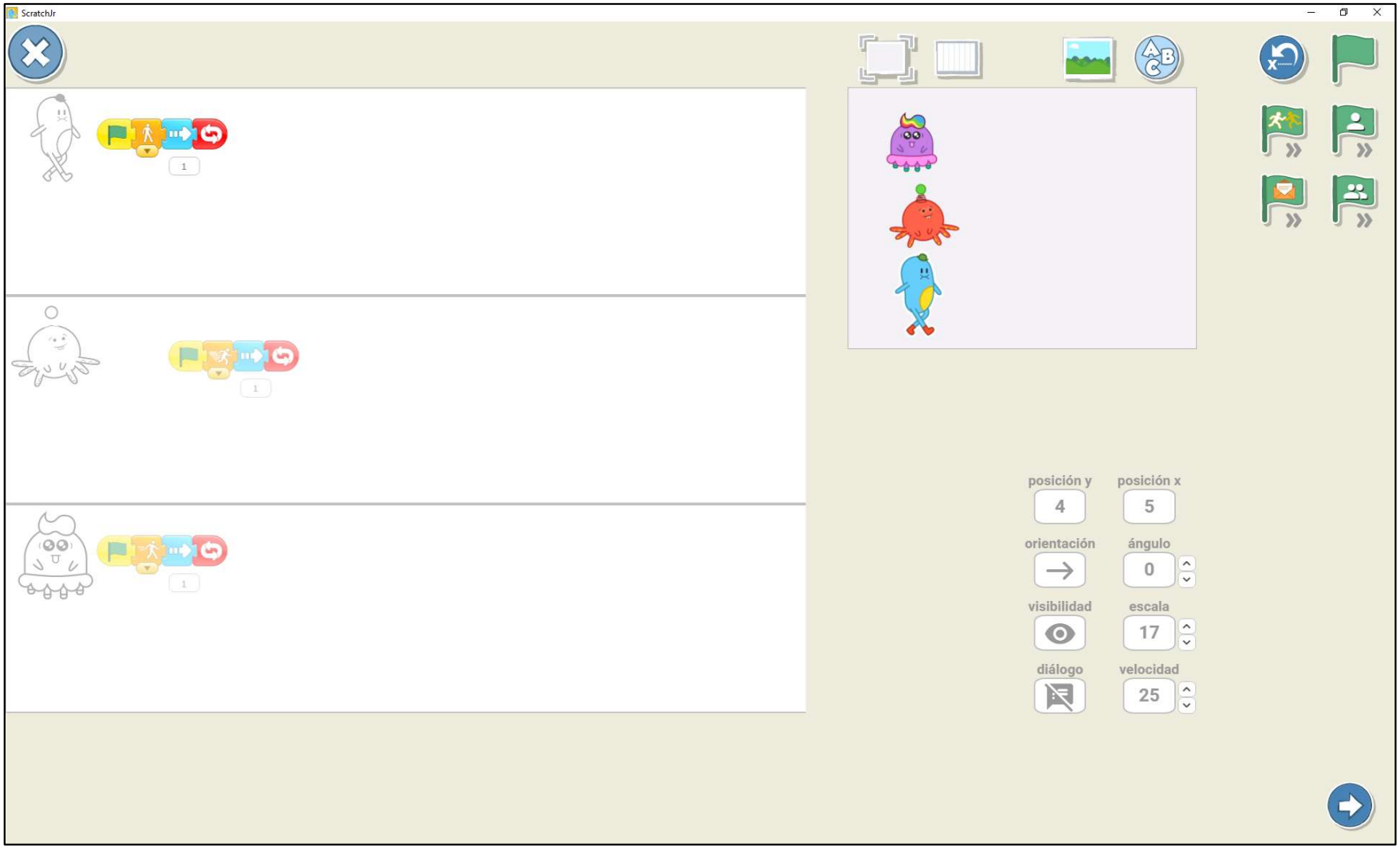

Fig. 5. Pantalla en la que se muestran simultáneamente todos los guiones antes de iniciar la ejecución

Conclusiones

Hemos presentado un rastreador de programas ScratchJr, cuyo objetivo es facilitar a los profesores la comprensión de la ejecución de los programas ScratchJr y facilitar su depuración. Aunque ScratchJr se utiliza principalmente en Educación Infantil, la herramienta también puede resultar útil a profesores de otras etapas educativas que aprendan ScratchJr como primer lenguaje de programación (Paredes-Velasco & Velázquez-Iturbide, 2022).

Como trabajos futuros se prevé evaluar su utilidad percibida y aceptación por parte de profesores en formación o en activo (Paredes-Velasco & Velázquez-Iturbide, 2022). También se prevé realizar otras mejoras. Primero, pueden añadirse más facilidades que permitan un control más fino de la ejecución de los programas. Para ello, podrían añadirse otros tipos de avance o permitir la selección de otros bloques como puntos de ruptura. Esta ampliación choca con el diseño minimalista de ScratchJr, ya que un aumento de la funcionalidad puede redundar en menos usabilidad (Velázquez-Iturbide, Pérez-Carrasco & Debdi, 2015), pero puede ser aceptable si ayuda a los profesores a comprender mejor la programación. Segundo, podría modificarse ligeramente la semántica operacional del propio lenguaje para conseguir homogeneidad, de forma la ejecución de cualquier bloque siempre se vea como un paso de la ejecución del programa.

Agradecimientos

Este trabajo se ha financiado con dos proyectos-puente de la Universidad Rey Juan Carlos (M2614 y M3035) y el proyecto PROGRAMA del Ministerio de Ciencia e Innovación (ref. PID2022-137849OB-I00).

Referencias Bibliográficas

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the ACM*, 60(6), 72-80. https://doi.org/10.1145/3015455.

Bers, M. U., & Resnick, M. (2016). The Official ScratchJr Book: Help Your Kids Learn to Code. No Start Press.

Carpensen, M. E., Gal-Ezer, J., McGettrick, A., & Nardelli, E. (2018). Informatics for All. The Strategy. ACM Europe & Informatics Europe. https://cutt.ly/zYh1Ze0.

du Boulay, B., O'Shea, T., & Monk, J. (1981). The black box inside the glass box: Presenting computing concepts to novices. *International Journal of Man-Machine Studies*, 14(3), 237-249. https://doi.org/10.1016/S0020-7373(81)80056-9.

Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2020). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, 29, 12-28. https://doi.org/10.1002/cae.22255.

Flannery, L. P., Kazakoff, E. R., Bontá, P., Silverman, B., Bers, M. U., & Resnick, M. (2013). Designing ScratchJr: Support for early childhood learning through computer programming. *Proceedings of the 12th International Conference on Interaction Design and Children, IDC '13* (pp. 1-10), ACM DL. https://goi.org/10.1145/2485760.2485785.

Gomes; A., & Mendes, A. J. (2007). Learning to program - difficulties and solutions. *Proceedings of the International Conference on Engineering Education, ICEE 2007*, Academic Press.

González González, C. S. (2019). Estado del arte en la enseñanza del pensamiento computacional y la programación en la etapa infantil. *Education in the Knowledge Society*, 20. https://doi.org/10.14201/eks2019_20_a17.

Koehler, M. J., & Mishra, P. (2009). What is technological pedagogical content knowledge? *Contemporary Issues in Technology and Teacher Education*, 9(1), 60-70.

Lockwood, J., & Mooney, A. (2018). Computational thinking in secondary education: Where does it fit? A systematic literary review. *International Journal of Computer Science Education in Schools*, 2(1). https://doi.org/10.21585/ijcses.v2i1.26.

Luxton-Reilly, A. Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). Introductory programming: A systematic literature review. *ITiCSE'18 Companion* (pp. 55-106), ACM DL, DOI 10.1145/3293881.3295779.

Macrides, E., Miliou, O., & Angeli., C. (2022). Programming in early childhood education: A systematic review. *International Journal of Child-Computer Interaction*, 32, 100396. https://doi.org/10.1016/j.ijcci.2021.100396.

Martínez-Valdés, J. A., Velázquez-Iturbide, J. Á., & Hijón-Neira, R. (2017). A (relatively) unsatisfactory experience of use of Scratch in CS1. *Proceedings of 5th International Conference on Technological Ecosystems for Enhancing Multiculturality, TEEM'17* (6 pp.), ACM DL. https://doi.org/10.1145/3144826.3145356.

Paredes-Velasco, M., & Velázquez-Iturbide, J. Á. (2022). Una asignatura para la formación del profesorado en programación mediante lenguajes basados en bloques. *Actas de las JENUI 2022*, 7, 337-344. https://aenui.org/actas/pdf/JENUI_2022.pdf.

Pareja-Flores, C., Urquiza-Fuentes, J., & Velázquez Iturbide, J. Á. (2007). WinHIPE: An IDE for functional programming based on rewriting and visualization. *ACM SIGPLAN Notices*, 42(3), 14-23. https://doi.org/10.1145/1273039.1273042.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60–67. https://doi.org/10.1145/1592761.1592779.

Robins, A. B. (2019). Novice programmers and introductory programming. *The Cambridge Handbook of Computing Education Research*, Fincher, S. A., & Robins, A. V., Eds. Cambridge University Press, pp. 327-376.

Seppälä, O., Ball, T., Barik, T., Becker, B. A., Denny, P., Duran, R., Sorva, J., & Velázquez-Iturbide, J. Á. (2019). Notional machines for Scratch and Python. *Notional Machines and Programming Language Semantics in Education*, Guzdial, M., Krishnamurthi, S.,

Sorva, J., & Vahrenhold, J., Eds., Dagstuhl Reports, Dagstuhl Publishing, 9(7), 21. https://doi.org/10.4230/DagRep.9.7.1.

Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education*, 13(2), artículo 8. https://doi.org/10.1145/2483710.2483713.

Velázquez-Iturbide, J. Á. (2021). Towards the design of notional machines for simple block-based languages. *Proceedings of the International Symposium on Computers in Education, SIIE'21* (6 pp.), Balderas, A., Mendes, A., & Dodero, J. M., Eds. IEEE Xplore. https://doi.org/10.1109/SIIE53363.2021.9583645.

Velázquez-Iturbide, J. Á. (2023). Designing exercises for block-based languages: The case of ScratchJr. *Proceedings TEEM 2022: Tenth International Conference on Technological Ecosystems for Enhancing Multiculturality*, García-Peñalvo, F. J., & García-Holgado, A., Eds. Lecture Notes in Educational Technology, Springer, pp. 50-59. https://doi.org/10.1007/978-981-99-0942-1_5.

Velázquez-Iturbide, J. Á., Llorens-Largo, F., López-Álvarez, D., & Marqués-Andrés, M. (2023). *Informe CODDII/SCIE sobre formación del profesorado y didáctica de la informática en etapas preuniversitarias*, Sociedad Científica Informática de España. https://onx.la/c025d.

Velázquez-Iturbide, J. Á., Pérez-Carrasco, A., & Debdi, O. (2015). Experiences in usability evaluation of educational programming tools. *STEM Education: Concepts, Methodologies, Tools, and Applications*, González-González, C., Ed. IGI Global, pp. 461-480. https://doi.org/10.4018/978-1-4666-7363-2.ch025.

Nota curricular

J. Ángel Velázquez Iturbide es Catedrático de Universidad de la Universidad Rey Juan Carlos y director del Laboratorio de Tecnologías de la Información en la Educación (LITE). Su principal área de investigación es la didáctica y el software educativo para la educación en programación y en algoritmia, con un énfasis actual en la programación basada en bloques.
El Dr. Velázquez es miembro senior de IEEE Computer Society, IEEE Education Society, ACM y ACM SIGCSE. También es vicepresidente de la Asociación para el Desarrollo de la Informática Educativa (ADIE), vocal para educación preuniversitaria de la Sociedad Científica Informática de España (SCIE) y miembro del *Steering Committee* de la coalición *Informatics for All*.

David de Vicente Peña es Graduado en Ingeniería Informática (2023) por la Universidad Rey Juan Carlos, Madrid, España. Realizó su trabajo fin de grado con títutlo "Prototipo de un rastreador/depurador de ScratchJr".

Eva López Puente es Graduada en Ingeniería Informática por la Universidad Rey Juan Carlos, Madrid, España. Actualmente se encuentra matriculada en la Escuela Internacional de Doctorado de la Universidad, realizando su tesis doctoral sobre el desarrollo de aplicaciones educativas colaborativas para el lenguaje ScratchJr.