CB-1.142

TALLERES CON SCRATCH PARA LA ADQUISICIÓN DE APRENDIZAJES MATEMÁTICOS EN EDUCACIÓN PRIMARIA

José Antonio Rodríguez – José Antonio González-Calero – José Manuel Sáez
joseantonio.rodriguez1@alu.uclm.es – jose.gonzalezcalero@uclm.es – jmsaezlopez@edu.uned.es
UCLM, España – UCLM, España. – UNED, España.

Núcleo temático: Enseñanza y aprendizaje de la Matemática en las diferentes modalidades y niveles educativos.
Modalidad: CB
Nivel educativo: Primario (6 a 11 años)
Palabras clave: Scratch, pensamiento computacional, resolución de problemas

Resumen:
Las potenciales ventajas del uso de la programación en el aprendizaje de las matemáticas, sumado a la aparición de lenguajes de programación visuales como Scratch, han reavivado el uso de la programación con objetivos didácticos. Sin embargo, la mayoría de experiencias y estudios se centran en las etapas de Bachillerato y Educación Secundaria. Esta comunicación da cuenta de una experiencia de aula mediante la que se inicia a alumnos de 6º Educación Primaria en el desarrollo del pensamiento computacional y sus posibles beneficios a la hora de la adquisición de conceptos matemáticos. Organizado en forma de talleres, una primera fase de instrucción se encuentra enfocada en el lenguaje de programación y a la gestación de los primeros conceptos relacionados con el pensamiento computacional, y una segunda fase en la que el lenguaje computacional se utiliza como herramienta para el desarrollo de actividades relacionadas a la resolución de tareas matemáticas. La comunicación se centrará en el diseño de los talleres y en las tareas matemáticas resolubles mediante Scratch.

Introducción

Los cambios vertiginosos que están ocurriendo en nuestra sociedad actual debido a las innovaciones tecnológicas influyen directamente en todos los niveles de la educación (Cabero y Llorente, 2010). Sin embargo, en el caso de Educación Primaria, la introducción de lenguajes de programación, nunca han sido utilizados de modo extensivo, ni han perdurado en el tiempo (Resnick et al., 2009). Esto puede deberse a que los primeros lenguajes de programación resultaban poco amigables y complejos para estas edades. Ante este problema no han sido pocos los lenguajes de programación que se han intentado adaptar o simplificar con fines educativos (p.ej., *Greenfoot*, *Etoys*, *Alice* o *LOGO*), pero, sin embargo, ninguno de ellos ha conseguido afianzarse dentro de las aulas (Lewis, 2010). En este contexto, el área pedagógica del Masacchussets Institute of Tecnhology (MIT) desarrolló *Scratch*, un lenguaje de programación, orientado especialmente a alumnos de Educación Primaria y Secundaria, que permite programar de manera simplificada y más

ISBN 978-84-945722-3-4

visual (Lee, 2009). Los lenguajes de programación apropiados para estas edades deben ser fáciles de empezar a programar, pero, al mismo tiempo, posibilitar el desarrollo de proyectos complejos con el tiempo y la capacidad de tener un amplio abanico de potenciales proyectos (Resnick et al., 2009).

Aunque en la actualidad coexisten diferentes teorías en relación a la educación y las TIC, la inmensa mayoría abogan por el uso de las nuevas tecnologías y destacan la obligación de un cambio en el paradigma educativo (Wing, 2008). Este autor advierte que, a mediados del siglo XXI, el pensamiento computacional será una competencia clave para nuestro alumnado. El pensamiento computacional se define como la habilidad de resolución de problemas, diseño de sistemas y comprensión del comportamiento humano basado en conceptos y procesos informáticos (Wing, 2008). En Wing (2014) se afirma que los beneficios educativos de poder pensar computacionalmente, comenzando con el uso de abstracciones y de las capacidades de razonamiento, mejoran y refuerzan las habilidades intelectuales y, por lo tanto, pueden transferirse a cualquier dominio. En concreto, conocer cómo funcionan los bucles, los condicionales, las secuencias, el trabajo con variables y el desarrollo de algoritmos está asociado con un desarrollo en la capacidad de los estudiantes de solventar problemas y crear soluciones eficaces (Chao, 2016).

El contexto anteriormente descrito invita a pensar que *Scratch* puede ser una herramienta con potencial educativo en las aulas. Este trabajo pretende evaluar el potencial de alguna de las configuraciones en las que el lenguaje de programación *Scratch* podría aplicarse en los niveles de Educación Primaria.

Antecedentes

En 1967, un equipo de investigadores pioneros en la inteligencia artificial y la robótica, entre los cuales destacaba el educador Seymour Papert, encontraron la necesidad de desarrollar un lenguaje de programación con el objetivo de ofrecer una metodología novedosa, especialmente en relación con la enseñanza de las matemáticas. Desde su punto de vista, en esta materia imperaba una enseñanza basada en la aplicación de métodos formales, generalmente tediosos, poco motivadores y mecánicos (Feurzeig, Papert y Lawler, 2011). Papert (1980) deseaba aplicar las teorías constructivistas de Jean Piaget en el ámbito de las matemáticas para que fuera el propio alumno el que impartiendo órdenes a una pequeña tortuga, solucionara problemas matemáticos. De esta forma nació *LOGO*, lenguaje de programación muy cercano al pseudocódigo, que fue muy conocido en la época. Con *LOGO* se establecieron las bases de diferentes proyectos que pretendían instrumentalizar la programación en la enseñanza de las matemáticas. existen numerosos estudios que destacan las ventajas del uso de este lenguaje de programación en las clases de matemáticas (p.ej., Howe et al., 1982; o, Layman y Hall, 1988). Bar-On (1986) afirmaba que la utilización de *LOGO* "da la oportunidad de explorar el enfoque algorítmico de las operaciones matemáticas, la visualización matemática de objetos, así como facilitar la comprensión y las ideas matemáticas" (p. 400). Autores como Pea y Kurland (1984) detectaron que enseñar a programar con Logo a los alumnos mejora su capacidad a la hora de resolver problemas, mientras que otros, aún reconociendo este elemento positivo, alertaron del riesgo de no utilizar la programación correctamente y de caer en automatismos y ejercicios con escaso sentido didáctico (p.ej., Layman y Hall, 1988). Además, más allá del ámbito de las matemáticas, se encontraron evidencias de que el trabajo con *LOGO* podría generar otros beneficios educativos y de comportamiento. Por ejemplo, Feurzeig y

ISBN 978-84-945722-3-4

Papert (1969) detectó una mejora evidente en las tasas de lectura comprensiva en estudiantes de enseñanza secundaria, así como cambios de comportamiento tales como mayor grado de confianza en sí mismos, actitudes sociales más positivas o ampliación de los intereses intelectuales. A pesar de la existencia de investigaciones que evidencian los posibles efectos positivos del uso de *LOGO*, conviene destacar que la gran mayoría de los estudios se centraron en alumnos con una edad superior a doce años
Sobre *Scratch*, heredero de *LOGO*, se han desarrollado recientemente diferentes estudios que señalan diversos beneficios en el desarrollo del pensamiento matemático como resultado del trabajo con este lenguaje de programación (p. ej., Calao et al., 2015). Otro estudio reciente subraya que la utilización de *Scratch* en sesiones de matemáticas permite que los profesores adapten su propio estilo de enseñanza y experiencia a las necesidades de sus alumnos para comunicar no solo ideas computacionales sino también matemáticas (Benton, Hoyles, Kalas, y Noss, 2017). En la misma línea, Calder (2010) vuelve a poner en el foco de atención en la herramienta al señalar que su utilización obliga a los alumnos a utilizar de forma implícita conocimientos matemáticos en la creación de sus proyectos en *Scratch*, demostrando ser un medio por el cual los programas son fácilmente compuestos y modificados, fomentando así el uso de habilidades críticas, meta-cognitivas y reflexivas siempre relacionadas con las matemáticas. En este sentido, se ha de señalar que, a pesar de que los estudios anteriores presentan elementos de confianza a la hora de valorar el potencial de *Scratch* en la asignatura de matemáticas, son muy escasos los trabajos que identifican mejoras en áreas concretas de las matemáticas.

Objetivos

El objetivo principal de este estudio es evaluar la influencia del uso de la herramienta *Scratch* en alumnos de 6º de Primaria en la adquisición de elementos del pensamiento computacional y la adquisición de conceptos matemáticos. En relación con el segundo de los objetivos, el estudio se centra en dos aprendizajes concretos que vienen definidos en el currículo mediante estándares de aprendizaje evaluables (Decreto 54/2014 de Castilla-La Mancha, región española donde se realiza el estudio). Dichos estándares vienen definidos como: *Calcula el mínimo común múltiplo y el máximo común divisor* y *Describe posiciones y movimientos por medio de coordenadas, distancias, ángulos, giros…*

Diseño experimental

Para la realización del experimento se plantea un estudio cuasi-experimental divido en dos fases: una inicial, basada en una serie de talleres de instrucción en Scratch y una segunda fase donde se completan talleres matemáticos relacionados con los estándares de aprendizajes señalados en la sección anterior (véase Figura 2). Tanto el grupo de control como el grupo experimental completa un pre-test antes de la primera fase con el objeto de evaluar su nivel previo en pensamiento computacional. Igualmente antes de iniciar la segunda fase ambos grupos realizan un pre-test para mensurar el nivel previo de los estudiantes en relación con los estándares de aprendizaje seleccionados. Durante la segunda fase del experimento el grupo experimental utiliza *Scratch* como herramienta de trabajo para la resolución de las tareas planteadas en los talleres a lo largo de cuatro sesiones mientras, por otra parte, el grupo de control completa los mismos talleres aunque sin la posibilidad de emplear *Scratch* como herramienta.

ISBN 978-84-945722-3-4

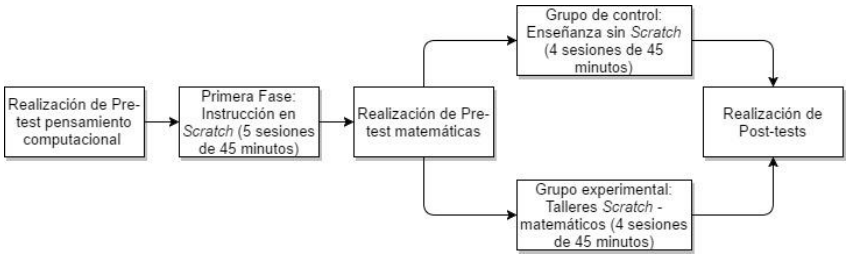

Figura 2: Organización de fases

Al finalizar la segunda fase de la experimentación, los participantes de ambos grupos completaron dos post-tests, uno orientado al pensamiento computacional y otro al grado de adquisición de los estándares de aprendizaje de la materia de Matemáticas. La comparación entre pre-tests y post-tests posibilitará tanto comprobar el grado de adquisición de elementos relacionados al pensamiento computacional como el grado de adquisición de los conocimientos matemáticos en ambos grupos.

Participantes

En el estudio toman parte 48 estudiantes de dos grupos naturales de sexto curso de Educación Primaria de un colegio público urbano de la región de Castilla-La Mancha (España). El grupo experimental y el grupo de control están formado por 24 estudiantes (10 alumnos y 14 alumnas) y 23 estudiantes (9 alumnos y 14 alumnas), respectivamente.

Instrumentos

Debido a la escasez de estudios que aborden el pensamiento computacional en edades tempranas fue necesario el desarrollo de un cuestionario *ad hoc* para evaluar el nivel previo de pensamiento computacional de los estudiantes. El diseño de los ítems se basó en las recomendaciones de Román-González, Pérez-González y Jiménez-Fernández (2015). El cuestionario consta de diez ítems que comprenden cuatro elementos del pensamiento computacional. A continuación se muestra un ejemplo de ítem para cada una de estas cuatro componentes:

- *Secuencias*. Sucesión de instrucciones que se siguen para completar una meta (véase Figura 2 para un ejemplo de ítem ligado a este elemento del pensamiento computacional).
- *Iteración o bucle*. Estructura de control que permite repetir una o más secuencias múltiples veces (Figura 3).
- *Manejo de eventos*. Instrucciones que permiten interactuar con los objetos del entorno (Figura 4).
- *Condicionales*. Instrucciones que permiten realizar, o no, una acción (Figura 5).

ISBN 978-84-945722-3-4

Figura 3: Ejemplo de ítem sobre secuencias.

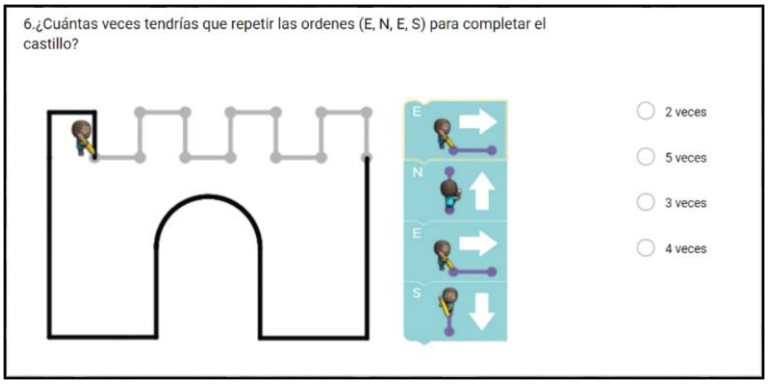

Figura 4: Ejemplo de ítem sobre bucles.

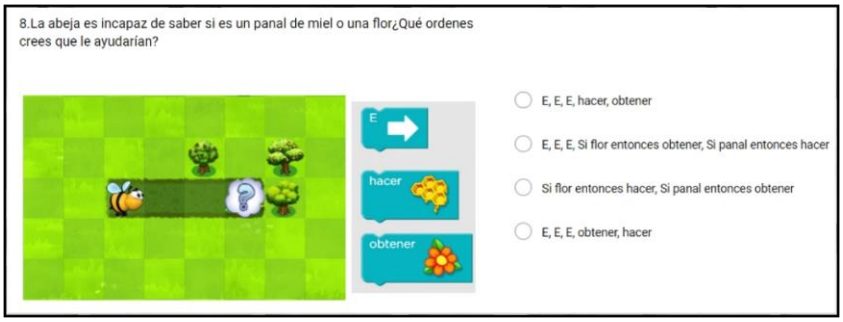

Figura 5: Ejemplo de ítem sobre eventos.

Figura 6: Ejemplo de ítem sobre condicionales.

El cuestionario post sobre pensamiento computacional es una versión del pre-test con el mismo número de ítems y en el que los ítems son isomorfos a los del pre-test, es decir, son

ISBN 978-84-945722-3-4

variaciones de los ítems utilizados en el cuestionario previo con el objeto de abordar los mismos objetivos. Tanto el pre- como el post-test en pensamiento computacional fueron diseñados como cuestionario *online.*.
El segundo de los instrumentos de evaluación se relación con la evaluación de los estándares de aprendizajes seleccionados de la materia de Matemáticas. Al igual que en el caso del pensamiento computacional, se diseñaron dos versiones del cuestionario (pre-test y post-test) con la misma estructura y características. El cuestionario consta de ocho ítems, cuatro ligadas a cada estándar de aprendizaje. A continuación se incluye un ejemplo de un ítems utilizado para el estándar de aprendizaje *Calcula el mínimo común múltiplo y el máximo común divisor* (Figura 6).

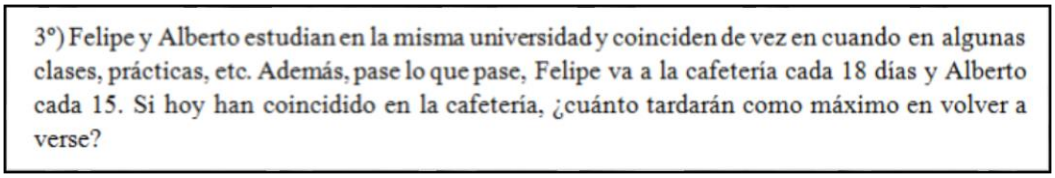
3°) Felipe y Alberto estudian en la misma universidad y coinciden de vez en cuando en algunas clases, prácticas, etc. Además, pase lo que pase, Felipe va a la cafetería cada 18 días y Alberto cada 15. Si hoy han coincidido en la cafetería, ¿cuánto tardarán como máximo en volver a verse?

Figura 7: Ejemplo de ítem ligado a estándar de aprendizaje

Metodología
La primera fase de la experimentación se centra exclusivamente en una instrucción básica en *Scratch* que posibilite al estudiante su uso como herramienta en la resolución de tareas matemáticas. Esta fase consta de cuatro sesiones de 45 minutos, cada una de las cuales se centra en uno de los siguientes elementos: secuencia, bucles, tratamiento de eventos y condicionales. El diseño de las sesiones está basado en las recomendaciones de Brennan, Balch y Chung (2014). Las sesiones se dividen en dos partes : una inicial con actividades grupales donde se introduce la componente del pensamiento computacional en cuestión en tareas donde no se involucra *Scratch* y, una segunda parte, donde se traslada ese elemento al lenguaje de programación de *Scratch*. Este diseño se basa en las recomendaciones de Benton et al. (2017), quienes sostienen que actividades en las que el alumno actúa un objeto con el que se programa, constituyen una base adecuada para posteriormente identificar las acciones en el lenguaje de programación. En la segunda fase del estudio, ambos grupos completan cuatro talleres de 45 minutos dedicados a la resolución de problemas matemáticos. El grupo experimental, a diferencia del grupo de control, emplea *Scratch* en todas las sesiones como herramienta para la resolución de los problemas planteados

Consideraciones.
En la fecha de escritura del presente trabajo está próxima a su conclusión la segunda fase del estudio, la referente a los talleres de contenido matemático. En consecuencia, aún no es posible una evaluación cuantitativa del impacto que la instrucción y el uso de *Scratch* en la resolución de problemas matemáticos pueda tener tanto en el desarrollo del pensamiento computacional como en el adquisición de aprendizajes matemáticos. Sin embargo, sí es posible una valoración previa acerca del diseño e implementación de las diferentes sesiones. En este sentido, es reseñable el hecho de cómo una instrucción de escasa duración en el lenguaje de programación, capacita a los estudiantes para usar *Scratch* de manera autónoma en la resolución de tareas matemáticas. La posibilidad de emplear *Scratch* en la

ISBN 978-84-945722-3-4

etapa de Educación Primaria sin la necesidad de invertir un elevado número de horas en la formación en el propio lenguaje, como se ha atestiguado en este estudio, es un elemento altamente positivo que permite considerar este lenguaje de programación como una herramienta con potencial para ser empleado de manera intensiva en estas edades.

Referencias bibliográficas

Bar-On, E. (1986). A programming approach to mathematics. Computers & Education, 10(4), 393-401.

Benton, L., Hoyles, C. Kalas, I., y Noss, R. (2017) Bridging Primary Programming and Mathematics: some findings of design research in England. Digital Experiences in Mathematics Education. Advance online publication. doi: 10.1007/s40751-017-0028-x.

Brennan, K., Balch, C., & Chung, M. (2014). Creative computing. Recuperado de: http://scratched.gse.harvard.edu/guide/

Cabero, J. y Llorente, M.C. (2010). Comunidades virtuales para el aprendizaje. Edutec. Revista Electrónica de Tecnología Educativa, 34, 1-10.

Calao, L. A., Moreno-León, J., Correa, H. E., y Robles, G. (2015). Developing mathematical thinking with Scratch: An experiment with 6th grade students. En G. Conole, T. Klobučar , C. Rensing, J. Konert, y É. Lavoué (Eds). Design for Teaching and Learning in a Networked World, 10th European Conference on Technology Enhanced Learning, EC-TEL 2015 (pp. 17-27). Toledo, Spain: Springer. doi: 10.1007/978-3-319-24258-3_2

Calder, N. (2010). Using Scratch: An Integrated Problem-Solving Approach to Mathematical Thinking. Australian Primary Mathematics Classroom, 15(4), 9-14.

Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. Computers & Education, 95, 202-215.

Feurzeig, W. (1969). Programming-Languages as a Conceptual Framework for Teaching Mathematics. Final Report on the First Fifteen Months of the LOGO Project. Cambridge, MA: BBN.

Feurzeig, W., Papert, S. A., y Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. Interactive Learning Environments, 19(5), 487-501.

Howe, J. A. M., Ross, P. M., Johnson, K. R., Plane, F., y Inglis, R. (1982). Teaching mathematics through programming in the classroom. Computers & Education, 6(1), 85-91.

Layman, J., y Hall, W. (1988). Logo: A cause for concern. Computers & Education, 12(1), 107-112.

Lee Jr, J. (2009). *Scratch programming for teens.* Boston: Cengage Learning.

Lewis, C. M. (2010). How programming environment shapes perception, learning and goals: Logo vs. Scratch. En Proceedings of the 41st ACM technical symposium on Computer science education (pp. 346-350). ACM.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Nueva York: Basic Books.

Pea, R. D. y Kurland, D. M. (1984). On the cognitive effects of learning computer programming. New ideas in psychology, 2(2), 137-168.

ISBN 978-84-945722-3-4

Resnick, M., Maloney, J., Monroy-Hernández, A. M., Rusk, N., Eastmond, E., Brennan, K., … Kafay, Y. (2009). Scratch: programming for all. Communications of the ACM, 52(1), 60-67. doi: 10.1145/1592761.1592779

Román-González, M., Pérez-González, J. C., y Jiménez-Fernández, C. (2015). Test de Pensamiento Computacional: diseño y psicometría general. En III Congreso Internacional sobre Aprendizaje, Innovación y Competitividad (CINAIC 2015), Octubre 14-16, 2015, Madrid, ESPAÑA. doi: 10.13140/RG.2.1.3056.5521

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), 3717-3725. doi: 10.1098/rsta.2008.0118

Wing, J. M. (2014, 10 de enero). Computational thinking benefits society [Blog post]. Recuperado de: http://socialissues.cs.toronto.edu/2014/01/computational-thinking/

ISBN 978-84-945722-3-4