suma+ 91
Artículos
Julio 2019
pp. 15-22

Pensamiento computacional y aprendizaje de las matemáticas a través de Scratch

Víctor López
Edelmira Badillo
Cristina Simarro
Digna Couso

Presentamos una breve secuencia formativa realizada en la asignatura de Aprendizaje de las Matemáticas con futuros maestros del grado de Educación Primaria. Se propone a los estudiantes diseñar mediante el lenguaje de programación por bloques Scratch un pequeño videojuego del Pong, donde una pelota se mueve aleatoriamente y rebota por toda la pantalla, y mediante una pala se debe evitar que la pelota se cuele por la parte inferior de la pantalla. El diseño de este videojuego lleva a los estudiantes a enfrentarse en pequeños grupos a distintos retos, tanto matemáticos como computacionales, a partir de los cuales surgen diferentes estrategias de resolución. A través de la discusión de las estrategias llevadas a cabo por los estudiantes pretendemos identificar algunas de las oportunidades didácticas que ofrece Scratch en la enseñanza y aprendizaje de la matemática escolar.

Palabras clave: Scratch, Programación, Pensamiento computacional, Geometría, Resolución de problemas.

Computational thinking and mathematics learning using Scratch

We present a brief instructional sequence in Mathematics Education course for pre-service primary school teachers. Students have to design a simple Pong videogame using programming language Scratch, in which a ball moves randomly around the screen, and players have to move a spade to avoid the ball slip down. The design process brings students to face both mathematic and computational challenges, and different strategies arise. By discussing those strategies, we aim to identify some of the learning opportunities offered by Scratch for mathematics teaching and learning in school.

Keywords: Scratch, Programming, Computational thinking, Geometry, Problem solving.

En los últimos años ha habido una eclosión de herramientas educativas vinculadas a las actividades de programación, robótica y otras actividades que se engloban dentro del paraguas llamado «pensamiento computacional». Esta idea aparece actualmente en muchos de los nuevos currículos y planes educativos. Por ejemplo, la idea de «usar las matemáticas y el pensamiento computacional» se considera una de las ocho prácticas científicas clave del K12 Next Generation Science Education Standards (NRC, 2012), y también es uno de los elementos centrales del reciente paradigma educativo STEM o STEAM (Simarro, López, Cornellà, Peracaula, Niell y Estebanell, 2016). Las actividades de programación y robótica se multiplican tanto en contextos formales (en asignaturas de programación, vinculada a la enseñanza-aprendizaje de otros contenidos curriculares, a través de proyectos STEAM) como no formales (extraescolares, clubs de programación, concursos y ligas de diseño de robots), y su presencia se observa en todas las etapas educativas (desde las *beebots* preescolares hasta las competiciones universitarias de robots).

Detrás de este enfoque educativo subyace la idea de que aprender a pensar computacionalmente (y aprender pensando computacionalmente) es clave para la resolución de problemas

Artículo recibido en *Suma* en mayo de 2018 y aceptado en diciembre de 2018

y para el diseño de todo tipo de soluciones (Wing, 2006), usando estrategias como la división de problemas complejos en módulos de tamaño inferior, la secuenciación de procesos largos y complejos en *pasos*, la organización de datos reconociendo patrones lógicos, la generalización de casos concretos para llegar a situaciones abstractas y generalizables, el uso de algoritmos para automatizar soluciones, o la validación y revisión constante de las soluciones propuestas (Selby y Woollard, 2014). Por un lado, este enfoque construccionista está en clara sintonía con el enfoque indagativo y modelizador del ámbito científico (Wagh, Cook-Whitt y Wilensky, 2017), pero por el otro, no se trata de habilidades excluyentes para los profesionales STEM, sino de competencias transversales para la ciudadanía (Zapata-Ros, 2015). En este sentido, autores como Brennan y Resnick (2012) han puesto el énfasis en que aprender a pensar computacionalmente no pasa solamente por aprender conceptos computacionales (secuencias, bucles, paralelismos, eventos, condicionales, operadores y datos), sino además por participar de las prácticas computacionales (incrementar e iterar, testear y depurar, reutilizar y combinar, abstraer y modularizar) y de las perspectivas computacionales (expresar, conectar y cuestionar). Estas prácticas y perspectivas, además, ayudan a entender la próxima naturaleza de las prácticas profesionales científicas, matemáticas e ingenieriles, cada vez más inmersas en un mundo digital y donde la modelización computacional ha redefinido la propia naturaleza de estas disciplinas (Weintrop y otros, 2016).

El lenguaje de programación Scratch

Dentro del amplio abanico de herramientas que promueven el desarrollo del pensamiento computacional, la plataforma de programación Scratch es sin duda la que más acogida y éxito está teniendo en la escuela (Maloney y otros, 2010). Heredero del lenguaje de programación LOGO, se trata de un sencillo lenguaje de programación por bloques desarrollado por el MIT (Massachusetts Institute of Technology) que puede ser usado tanto *online* como *offline*. Permite a los estudiantes a partir de 6 años crear y compartir una gran variedad de creaciones (animaciones y narraciones, presentaciones, imágenes interactivas, simulaciones, juegos, etc.), mediante la construcción de bloques de programación a partir de piezas, como si se tratara de un sencillo rompecabezas. Cada pieza constituye una línea de código, con los que el usuario puede definir variables, cambios en la apariencia de los personajes, funciones condicionales («si», «solo si», «siempre que», etc.), funciones lógicas y operadores matemáticos («y», «o», «+», «mayor que», etc.), así como los *inputs* que hagan ejecutar cada elemento del programa («al pulsar una tecla», «al mover el ratón», etc.). Los distintos colores (figura 1) representan diferentes tipos de funciones: azul para el movimiento de los objetos, morado para la apariencia, marrón para los eventos, verde para los operadores matemáticos y amarillo para los elementos de control. Su interfaz de programación permite ver a la vez el código que se crea (las piezas en bloques) y un escenario con personajes que interactúan según las reglas que se le determina con este código. Cada usuario puede guardar y compartir sus pequeños proyectos, pero también reusar los proyectos de otros usuarios.

Figura 1. Principales piezas de código que, debidamente conectadas y secuenciadas, permiten generar pequeños programas

Scratch, pensamiento computacional y aprendizaje de las matemáticas

Una de las áreas del conocimiento donde este lenguaje de programación ofrece mayor interés es la de matemáticas. Las primeras investigaciones

en torno al papel de Scratch en la enseñanza y aprendizaje de las matemáticas resaltan su potencial en la construcción de significado de las ideas matemáticas. Benton, Hoyles, Kalas y Noss (2017) proponen un enfoque metodológico *ScratchMaths*, que resulta ser facilitador tanto para profesorado como alumnado en primaria para construir el concepto de algoritmo y de giro total de 360º en un polígono. A su vez, Calao, Moreno-León, Correa y Robles (2015) encuentran una mejora significativa del pensamiento matemático en grupos experimentales de estudiantes de sexto de primaria (11-12 años) que han recibido formación en Scratch cuando se evalúa sus habilidades de modelización matemática, razonamiento, formulación y resolución de problemas y comparación y ejecución de procesos y algoritmos. No es casual que el currículo de matemáticas para primaria en Cataluña, por ejemplo, incluya dentro de la dimensión «resolución de problemas», que la programación y la robótica educativa requieren estrategias que favorecen las habilidades de resolución de problemas. De igual manera, se resalta la importancia del papel de los recursos, como por ejemplo los materiales manipulativos, visuales y las TIC, puesto que favorecen la experimentación, el razonamiento y la comprensión de las ideas matemáticas, así como su transferencia para la interpretación de fenómenos del mundo (Burgués y Sarramona, 2013).

Un ejemplo de reto matemático y computacional: diseñar un mini-juego de Pong

Una actividad que ejemplifica este potencial educativo de Scratch para promover el pensamiento computacional en el área de matemáticas es el diseño del mini-juego llamado Pong. Es una actividad inspirada en la que propone Córdova (2013) con estudiantes de primaria, y consiste en definir los elementos básicos del juego y sus interacciones usando el lenguaje Scratch. En este juego, se espera que la pelota pueda moverse libremente por toda la superficie de la pantalla, y rebotar al tocar los laterales y el extremo superior (figura 2). El jugador, controlando el movimiento horizontal de la pala, debe interceptar la pelota siempre que se desplaza hacia el extremo inferior de la pantalla, hacerla rebotar y evitar así que el balón se cuele por la parte inferior de la pantalla. Este juego para un jugador es, de hecho, la versión previa para hacer un juego de Ping-Pong para dos jugadores, uno de los primeros videojuegos de primera generación que apareció en las máquinas recreativas de los años setenta del siglo pasado (David, 2004).

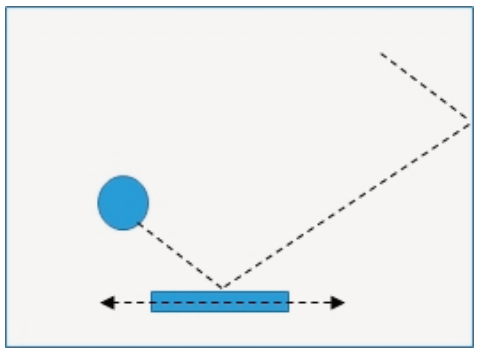

Figura 2. Representación de los elementos del juego Pong y de sus movimientos

Esta actividad no solo ha sido ampliamente usada en Scratch (en la plataforma hay decenas de programas hechos por estudiantes de todo el mundo), sino que se enmarca en la Competencia 1 del listado de competencias básicas del ámbito matemático (Burgués y Sarramona, 2013), que se concreta en la capacidad de traducir un problema en lenguaje matemático o una representación matemática utilizando variables, símbolos, diagramas y modelos adecuados.

Esta actividad ha sido realizada en el curso 2017-18 en la asignatura Aprendizaje de las Matemáticas, dentro del grado de Educación Primaria de la UAB. En ella han participado un total de 80 estudiantes, organizados en pequeños grupos de 3-4 miembros. La actividad se plantea en forma de taller, de 2 horas y 30 minutos de duración, y se enmarca en una secuencia formativa más amplia en la que los estudiantes posteriormente deben exponer las ideas clave desarrolladas durante el taller. Para guiar el trabajo en pequeños grupos, a lo largo del taller se plantea a los estudiantes tres retos:

— El reto de definir y ejecutar el movimiento del balón y de la pala.
— El reto de definir la interacción entre el balón y la pala.
— El reto de refinar o ampliar el juego para mejorar su funcionamiento e incorporar comportamientos que añadan realismo y motivación.

A través de los retos que se plantean al alumnado participante, a continuación, presentamos y discutimos cuáles son las estrategias de resolución que emergen en los diferentes grupos de trabajo, así como las oportunidades didácticas que identificamos tanto para desarrollar el pensamiento matemático como el computacional.

El reto de definir y ejecutar el movimiento del balón y de la pala

El taller, de hecho, empieza planteando a los estudiantes qué elementos debe tener el juego para funcionar. En una actividad de exploración de ideas previas seguida de una puesta en común en la pizarra digital (figura 3), los estudiantes identifican los objetos físicos; las magnitudes que definen el comportamiento de estos objetos físicos; las reglas que determinan la interacción entre los personajes; así como otros elementos que permitirían mejorar el juego, haciéndolo más realista o más emocionante para un jugador.

Una vez realizada la discusión inicial, se propone a los estudiantes el reto de definir y ejecutar el movimiento de los dos principales objetos que intervienen en el juego: el balón y la pala. Para ello, se les plantea la siguiente demanda:

a) El lenguaje Scratch tiene un conjunto de bloques de programación de color marrón llamado «Eventos», que son necesarios para ejecutar cada bloque de piezas. ¿De qué maneras se puede hacer ejecutar el movimiento de la pelota y el movimiento de la pala? ¿Qué diferencias existen y por qué?

b) El lenguaje Scratch tiene un conjunto de bloques de programación de color azul, que permiten definir el movimiento de los personajes. ¿De qué maneras se puede definir el movimiento de los objetos con Scratch y cuál es más conveniente para definir el movimiento de la pala y el movimiento de la pelota?

A partir de esta doble demanda (definir cada movimiento mediante bloques de programación y a su vez definir la manera en cómo se ejecuta cada bloque), afloran múltiples discusiones. Los estudiantes se percatan de las diferencias entre el movimiento del balón y de la pala: el balón se mueve en todas direcciones, mientras que la pala solo lo hace en el eje horizontal. Esto les lleva a preguntarse qué tipo de coordenadas ayudan a describir mejor cada movimiento, apareciendo así la distinción entre describir el movimiento mediante los ejes X e Y (coordenadas cartesianas) o mediante pasos en una dirección y giros respecto esta dirección (coordenadas polares).

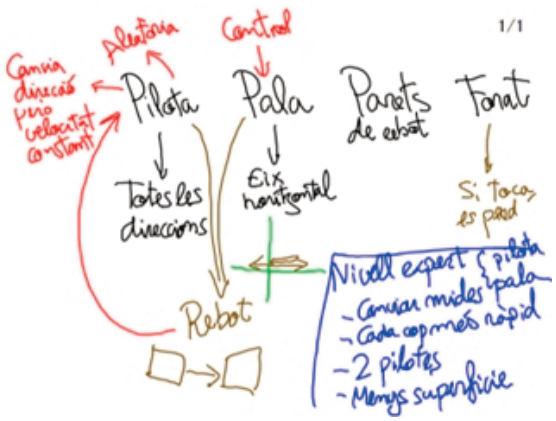

Figura 3. Puesta en común en la pizarra de los elementos que debe contener el juego

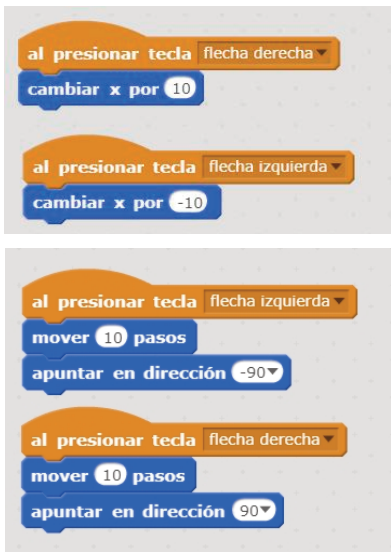

Figura 4. Dos propuestas diferentes para definir el movimiento de la pala, mediante coordenadas cartesianas (arriba) y polares (abajo)

En la figura 4 se observan las dos propuestas para definir el movimiento de la pala, ambas pensadas para que el jugador controle este movimiento con las flechas laterales del teclado. Mientras un grupo aboga por definir el desplazamiento lateral sumando y restando valores a la posición X de la pala, el segundo grupo aboga por girar la pala 180º y hacerla avanzar siempre hacia la dirección en la que apunta. A su vez, este disenso lleva a una nueva cuestión: ¿cuál es la mejor solución desde el punto de vista computacional? Esto permite plantear una importante perspectiva computacional: ¿todo código en un programa informático debe tender al máximo de simplicidad?

Además de definir el movimiento de la pala, la discusión se vuelve aún más intensa cuando se enfrentan al reto de definir el movimiento del balón. A diferencia de lo que ocurre con la pala, el balón debe moverse *solo* por la pantalla, sin ir controlando su movimiento con el teclado. Algunos estudiantes, por ejemplo, proponen un patrón de movimiento a partir de una secuencia de órdenes que simulen un movimiento caótico por toda la pantalla (figura 5, izquierda), pero pronto se dan cuenta que en cada partida el balón se moverá igual. Después de múltiples versiones, la mejor solución propone definir una dirección aleatoria inicial, que en cada partida sea distinta y que permita al balón desplazarse libremente y de forma indefinida.

Igual como ocurre con el movimiento de la pala, desde el punto de vista computacional, los estudiantes discuten sobre cómo combinar las diferentes piezas de código. Por ejemplo, se dan cuenta que si la pieza «apuntar en dirección» se introduce dentro de la pieza de control «por siempre» el movimiento de la pelota no es el esperado. En otras palabras: que las piezas de código no cumplen la propiedad conmutativa.

El reto de definir la interacción entre el balón y la pala

Una vez definidos estos dos movimientos (figuras 4 y 5), los estudiantes pronto se dan cuenta que con esto no basta, y que hay que definir la interacción entre balón y pala. Para ello, se les plantea la siguiente demanda:

> Ahora que ya están definidos los 2 movimientos, debemos definir la interacción entre los dos personajes: la pelota debe rebotar sobre la pala, y en caso de no hacerlo, perder el juego.
>
> c) El lenguaje Scratch tiene un conjunto de bloques de programación de color amarillo llamado «Control», que sirven para estructurar, secuenciar las órdenes que recibe cada personaje. Se pueden definir repeticiones (finitas o infinitas), y también condiciones del estilo «si … entonces ….». ¿De qué maneras se puede definir la interacción que se da en el rebote? Ten en cuenta que Scratch no tiene una opción del estilo «si un personaje toca a otro, que rebote», sino que hay que definir matemáticamente el rebote para poder introducir un algoritmo concreto en el lenguaje Scratch.
>
> d) Igual que has hecho en el apartado anterior, utilizando las piezas «Control», prueba a definir qué ocurre cuando el balón se cuela por debajo (no conseguimos hacerla rebotar). Al igual que antes, Scratch no tiene una opción del estilo «si un personaje toca un lateral, que pierda», sino que hay que definirla. Prueba a traducir esta expresión matemáticamente, para luego traducirla de nuevo al lenguaje Scratch.

La construcción de un algoritmo que defina el rebote de la pelota sobre la pala supone posiblemente el mayor reto matemático a lo largo

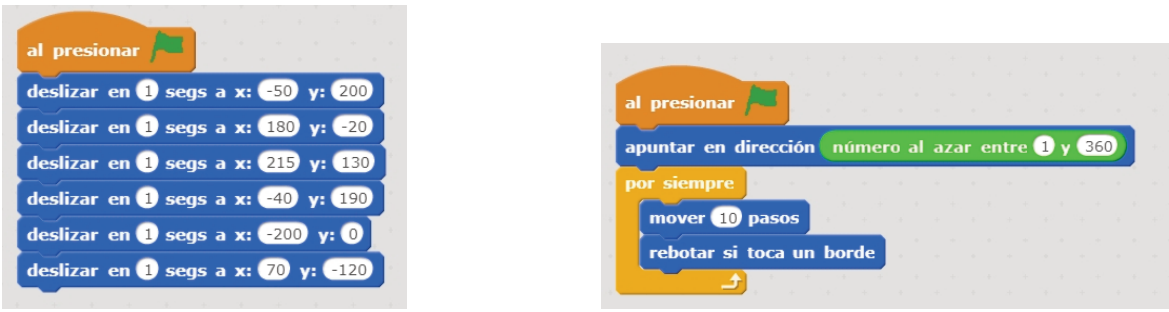

Figura 5. Dos propuestas diferentes para definir el movimiento del balón, mediante definición de desplazamientos entre puntos en el plano (izquierda) y mediante una dirección inicial aleatoria y una repetición de pasos (derecha).

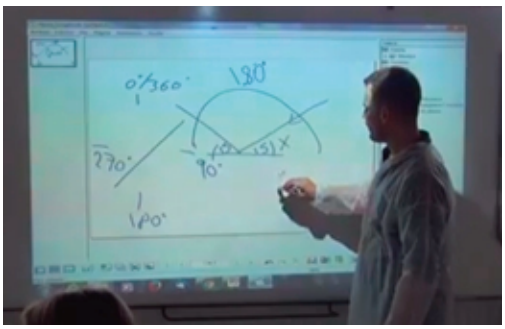

Figura 6. Un estudiante trata de resolver matemáticamente la definición del rebote (izquierda), y posteriormente traducir esta relación en el lenguaje computacional de Scratch (derecha)

del taller. Puesto que todos los grupos de estudiantes han usado las coordenadas polares para definir el movimiento del balón (figura 5, derecha), de forma intuitiva, el rebote es concebido inicialmente como un cambio en el signo del ángulo. Es común durante la discusión en pequeños grupos oir expresiones como: «cuando rebota, pasa de apuntar a 40º a –40º, ya que es ángulo inverso». Esto tendría sentido si no fuera porque en el lenguaje Scratch, todo ángulo se define a partir del eje vertical y, por lo tanto, el algoritmo necesario para definir el rebote horizontal requiere de una relación entre ángulos suplementarios. Es decir, se requiere considerar que los ángulos de entrada y de salida del rebote sumen 180º, lo que puede traducirse en lenguaje Scratch mediante un programa como el representado en la figura 6 (derecha).

En la discusión sobre qué algoritmo permite relacionar estos ángulos surgen diferentes estrategias. Algunos grupos de estudiantes siguen una estrategia de ensayo y error, mientras que otros buscan relaciones a través de simetrías en el plano.

En paralelo a la definición de qué le sucede al balón al tocar a la pala, los estudiantes deben definir también qué sucede cuando este se cuela por debajo de la posición de la pala. Algunos grupos de estudiantes optan por describir esta condición usando de nuevo los ejes de coordenadas. En este caso, para usar la relación de valor de Y inferior a –150, ya que define el semiplano que vendría a ser el borde inferior de la pantalla (figura 7, izquierda). Otros grupos optan por incorporar al nuevo juego un nuevo personaje, en este caso una simple barra horizontal que actúe de borde inferior, y definen la opción «tocando borde», igual que anteriormente se ha hecho con el «tocando pala». A su vez, la manera de representar en Scratch que se pierde la partida también es diferente en cada grupo. Entre una amplia variedad de formatos originales para representar en el juego que se ha perdido, los estudiantes proponen que el balón desparezca, cambie de aspecto, o bien aparezca un texto o sonido que indique que se ha perdido.

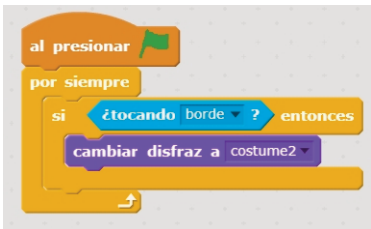

Figura 7. Dos maneras de definir qué sucede cuando el balón se cuela por debajo: definiendo la posición Y (arriba) o definiendo tocar otro personaje (abajo)

A lo largo de este reto, además, son múltiples las discusiones, dudas y cuestiones sobre cómo las distintas piezas deben ordenarse unas dentro de otras, secuenciarse, usarse en paralelo, etc. Sin duda, este contexto de discusión y construcción de conocimiento es otra evidencia de la generación de interesantes oportunidades de aprendizaje, tanto de ideas computacionales, como conectadas también al desarrollo del proceso de

Figura 8. Dos propuestas distintas de ampliación del juego: añadiendo dificultad creciente (izquierda) o realismo (derecha)

razonamiento y pensamiento conjetural del currículo de matemáticas por competencias. Por ejemplo, los estudiantes no solo deben usar los controles de tipo «si … entonces…» y «por siempre…», sino que deben decidir cómo se combinan entre ellos, ya que no es lo mismo un «por siempre si» que un «si por siempre».

El reto de refinar y ampliar el juego

El desarrollo del trabajo en pequeños grupos genera diferentes ritmos de trabajo, que en gran medida vienen dados por la experiencia previa de cada grupo en el lenguaje Scratch, el pensamiento computacional y los entornos de programación en general. Así, a aquellos grupos que han logrado los dos retos anteriormente presentados, es decir, que ya disponen de una versión funcional del videojuego de Pong, se les proponen diferentes retos avanzados. Estos pueden ser en torno a ampliar el juego para hacerlo más emocionante, ya sea añadiendo vidas, puntos, obstáculos, etc. Por ejemplo, uno de los grupos se le ocurre que el balón se mueva más rápido cada vez que rebota, para así hacer que el juego sea acelerado y con dificultad creciente. Para ello, definen la variable «velocidad» de manera que el balón ya no se mueve a 10 pasos por unidad de tiempo, sino a «velocidad» pasos por unidad de tiempo, y a su vez determinan que tras cada rebote la velocidad incrementa en un punto (figura 8, izquierda). Otros grupos optan por modelizar comportamientos físicos que añadan realismo al rebote. Por ejemplo, un estudiante propone añadir realismo reproduciendo un rebote que no sea perfecto, sino con pequeñas desviaciones aleatorias de $\pm 10°$, que simulan posibles irregularidades de la superficie de la pala (figura 8, derecha).

Conclusiones

Este taller realizado con estudiantes para maestros del grado de Educación Primaria ofrece múltiples oportunidades de aprendizaje, que permiten identificar el potencial del lenguaje de programación Scratch para la enseñanza y aprendizaje de las matemáticas escolares.

	Pensamiento matemático	Pensamiento computacional
El reto de definir y ejecutar el movimiento del balón y de la pala	• Coordenadas polares y cartesianas, y su idoneidad para describir cada movimiento • Valores fijos y aleatorios dentro de un rango	• Eventos automáticos o controlados • Secuenciación de órdenes • Bucles de repetición • Optimización del código
El reto de definir la interacción entre el balón y la pala	• Ángulos suplementarios • Simetrías • Semiplanos definidos por inecuaciones	• Bucles de repetición • Condicionales • Tiempo de ejecución de un comando
El reto de refinar y ampliar el juego	Modelización de fenómenos más realistas o más emocionantes, etc.	Depuración del código para hacer el programa más eficiente, reparar elementos que no funcionan, etc.

Tabla 1. Principales ideas, razonamientos y perspectivas relacionadas con el pensamiento matemático y el pensamiento computacional que se movilizan para cada uno de los retos

Para resolver los retos que se les plantean, los estudiantes deben movilizar tanto ideas y razonamientos del pensamiento matemático como del pensamiento computacional (tabla 1). De hecho, esta coordinación entre lo matemático y lo computacional es especialmente interesante desde la perspectiva del establecimiento de conexiones intramatemáticas y extramatemáticas en el aula de matemáticas (Gamboa, Badillo y Ribeiro, 2015).

Agradecimientos

Este trabajo ha sido posible gracias al proyecto PECOFIM (ARMIF2016 00031) y a los grupos de investigación educativa ACELEC (2017S GR1399) y GIPEAM (2017SGR101).

Referencias bibliográficas

Benton, L., C. Hoyles, I. Kalas y R. Noss (2017), «Bridging Primary Programming and Mathematics: some findings of design research in England», *Digital Experiences in Mathematics Education*, vol. 3, n.º 2, 115-138.

Brennan, K., y M. Resnick (2012), «New frameworks for studying and assessing the development of computational thinking, *Proceedings of the 2012 annual meeting of the American Educational Research Association*, Vancouver, 1-25.

Burgués, C., y J. Sarramona (coord.) (2013), *Competències bàsiques de l'àmbit matemàtic. Identificació i desplegament a l'educació primària*, Generalitat de Catalunya. Departament d'Educació. Direcció General d'Educació Infantil i Primària, Barcelona.

Calao, L. A., J. Moreno-León, H. E. Correa y G. Robles (2015), *Developing mathematical thinking with Scratch. Design for teaching and learning in a networked world,* Springer International Publishing, 17-27.

Cordova, P. (2013), *Crear jocs amb Scracht*, <https://crearjocs.blogspot.com.es/2013/07/pong.html>.

Ellis, D. (2004), *A Brief History of Video Games. Official Price Guide to Classic Video Games*, Random House, 3-5.

Gamboa, G., E. Badillo y M. Ribeiro (2015), «El horizonte matemático en el conocimiento para la enseñanza del profesor: Geometría y medida en educación primaria», *PNA*, 10(1), 1-24.

Maloney, J., M. Resnick, N. Rusk, B. Silverman y E. Eastmond (2010), «The scratch programming language and environament», *ACM Trans. Comput. Educ.,* 10(4), Article 16.

NRC (2012), *A framework for K-12 Science Education. Practices, Crosscutting Concepts and Core Ideas*, The National Academies Press, Washington, D.C.

Selby, C., y J. Woollard (2014), *Refining an understanding of computational thinking,* University of Southampton.

Simarro, C., V. López, P. Cornellà, M. Peracaula, M. Niell y M. Estebanell (2016), «Més enllà de la programació i la robòtica educativa: el pensament computacional en l'ensenyament STEAM a infantil i primària. Ciències» *Revista del Professorat de Ciències d'Infantil, Primària i Secundària*, 32, 38-46.

Wagh, A., K. Cook-Whitt y U. Wilensky (2017), «Bridging Inquiry-Based Science and Constructionism: Exploring the Alignment Between Students Tinkering with Code of Computational Models and Goals of Inquiry», *Journal of Research in Science Teaching*, 54, 615-641.

Weintrop, D., E. Beheshti, M. Horn, K. Orton, K. Jona, L. Trouille y U. Wilensky (2016), «Defining Computational Thinking for Mathematics and Science Classrooms», *Journal of Science Education and Technology*, 25(1), 127-147.

Wing, J. M. (2006), «Computational Thinking», *Communications of the ACM*, 49(3), 33-35.

Zapata-Ros, M. (2015), «Pensamiento computacional: Una nueva alfabetización digital», *Revista de Educación a Distancia*, 46, 1-47.

Víctor López
<victor.lopez@uab.cat>

Edelmira Badillo
<edelmira.badillo@uab.cat>

Digna Couso
<digna.couso@uab.cat>

Cristina Simarro
<cristina.simarro.rodriguez@uab.cat>

Universitat Autònoma de Barcelona