¿Pueden los niños aprender a programar usando un entorno de programación basado en texto con un agente compañero?

Elizabeth K. Morales-Urrutia[1], José M. Ocaña[2], Diana Pérez-Marín[3], Celeste Pizarro[3]

[1] Ejército Ecuatoriano, Ecuador, 182020, Ambato, Ecuador
[2] Universidad Técnica de Ambato, Ecuador, 182020, Ambato, Ecuador
[3] Universidad Rey Juan Carlos, España, 28933, Madrid, España
{e.morales.2017,m.ocana.2017}@alumnos.urjc.es, {diana.perez, celeste.pizarro}@urjc.es

Resumen: Aprender a programar en Educación Primaria está generando un gran interés investigador en los últimos años. Se necesita más investigación sobre los entornos y lenguajes de programación más adecuados para los niños. Un enfoque parece estar centrado en el uso de programas basados en bloques sin posibilidades de depuración y usando lenguajes multimedia simplificados. Por otro lado, también tiene bastante interés la investigación en agentes compañeros para facilitar el aprendizaje en niños. En este artículo, la pregunta de investigación es si los niños, sin conocimiento previo de programación, podrán aprender a programar usando un entorno de programación basado en texto con un agente compañero. La hipótesis es que los niños mejorarán sus notas de forma significativa usando un entorno de aprendizaje de la programación basado en texto con un agente compañero. Para comprobar la hipótesis se ha llevado a cabo un experimento con 21 estudiantes de entre 10 y 12 años en España. Durante los tres meses que duró el experimento, los estudiantes usaron el sistema una hora por semana. Los resultados obtenidos validan la hipótesis. Los niños pueden aprender a programar usando un entorno de programación basado en texto con un agente compañero sin necesidad de tener conocimientos previos de programación.
Palabras clave: Aprendizaje de programación, Entorno de programación basado en texto, Pseudocódigo, Educación Primaria, Compañero de Aprendizaje.

Abstract: Learning how to program in Primary Education has attracted a great deal of research for the last decades. It is not clear how programming environments and languages should be adapted to children for better learning and use. One trend seems to be focused on the use of visual block-based programming environments with multimedia programming languages. There has also been research a great deal of interest in the research of learning companions to help children to learn. In this paper, the research question is whether children, withour previous programming knowledge, can learn how to program by using a text-based programming environment with a learning companion. It is our hypothesis that children will have a significant increase of their test scores when using such programming learning environment. An experiment was carried out with 21 students between 10 and 12 years old in Spain. During the experiment, for three months, students used the textual programming environment with the learning companion one-hour per week. The pre-post test results have validated the hypothesis. Children can learn how to program by using a text-based programming environment with a learning companion, even if they do not have previous programming knowledge.
Key words: Learning programming, Text-based programming environment, Pseudocode, Primary Education, Learning Companion.

1. Introducción

Enseñar a programar a niños ha generado un gran interés a nivel mundial (Adam et al. 2019; Jacobsen, 2014). Algunos autores indican que uno de los principales beneficios de enseñar programación a los niños es desarrollar su pensamiento computacional (Wing, 2006). Otros autores, como en este artículo, se

 © ADIE, Asociación para el Desarrollo de la Informática Educativa

centran más en la búsqueda de entornos y lenguajes de programación más adecuados para la enseñanza a niños teniendo en cuenta diversos factores como su edad, nivel de competencia digital o conocimientos previos en programación.

En las últimas décadas, se han explorado varios enfoques para la enseñanza de la programación en Educación Primaria, como son: programación textual, programación visual y enfoque desenchufado. La programación textual comenzó en los años sesenta con el lenguaje de programación LOGO (un dialecto de Lisp) (Feurzeig & Papert, 1967).

LOGO, inicialmente, era una lenguaje interpretado con retroalimentación inmediata en cada instrucción. Los niños programaban en LOGO para mover una tortuga. La idea clave de este enfoque es que el mejor aprendizaje no viene de dar al profesor los mejores recursos de enseñanza sino en dar a los estudiantes oportunidades para crear pensando (Papert, 1980). Según este enfoque, escribir es una buena forma de que los estudiantes piensen y por lo tanto, aprendan a programar pensando.

La programación visual se basa normalmente en el uso de entornos de programación gráficos como Scratch (Resnick et al. 2009), en el que los niños no escriben. En su lugar, arrastran y sueltan bloques con las instrucciones de programación que encajan como en un puzzle (Ouahbi et al. 2015; Muñoz et al. 2017; Serna-Agudelo et al. 2018).

La programación desenchufada se centra en realizar ejercicios, sin necesidad de dispositivos electrónicos (Zapata-Ros, 2019). Lo importante es pensar para poder solucionar estos ejercicios de programación (Brackmann et al. 2016). No obstante, la eficacia de este tipo de enfoque aún no ha sido probada (Kalelioğlu, 2015).

Por otro lado, también ha tenido un gran interés la investigación en agentes compañeros. Estos agentes compañeros se pueden definir como un agente virtual que posee un cierto nivel de inteligencia y autonomía, además de habilidades sociales para permiten establecer y mantener relaciones con los usuarios a largo plazo (Lim, 2012). Los niños tienen suficiente imaginación para transformar situaciones reales en fantasía, dándoles personalidad y vida a los objetos inanimados que aparecen en su pantalla, e interactuando con ellos como si fueran sus amigos (Lee et al. 2009), mejorando de esta forma su relación con las aplicaciones informáticas (Roa-Seïlerm, 2016).

En este artículo, el enfoque que se seguirá será el primero descrito: programación textual. En nuestro trabajo previo (referencia eliminada por revisión a ciegas), hemos desarrollado un agente compañero que dialoga con los niños para poder enseñarles programación usando pseudocódigo. En particular, el agente les propone ejercicios de programación en los que deben usar determinados conceptos básicos como entrada, salida, condicionales o bucles, y los niños deben escribir el programa. El agente lo compila y les indica si han llegado a la solución correcta, o bien se inicia el proceso de depuración para poder llegar a la solución correcta.

Hasta ahora, siempre hemos usado el agente como continuación al aprendizaje previo de programación con Scratch. La pregunta de investigación, en este artículo, es si los niños, sin conocimiento previo de programación, aprenderán a programar con un entorno de programación textual usando pseudocódigo con un agente compañero. La hipótesis es que sí serán capaces debido al potencial pedagógico de la programación textual basada en la teoría del construccionismo de Papert (1980) y al potencial emocional de tener un agente compañero que les ayuda a llegar a la solución correcta, permitiendo a los estudiantes equivocarse, entender cuál es su error y corregirlo para llegar a su solución.
Se ha realizado un experimento con 21 estudiantes de 10 a 12 años estudiando Educación Primaria en España sin conocimientos previos de programación. En España no es obligatoria la enseñanza de programación hasta Educación Secundaria, aunque ya hay publicados estudios solicitando la necesidad de incorporar una asignatura de informática en edades más tempranas (Velázquez-Iturbide, 2018a; Velázquez-Iturbide, 2018b).

En su lugar, en la actualidad, la programación se enseña como una actividad transversal en el resto de asignaturas o bien como una actividad extraescolar. Para el experimento, se solicitó la colaboración a un colegio voluntario que accedió a dedicar parte de la asignatura de Plástica para programación durante tres

meses, una hora por semana. Al inicio del experimento todos los niños completaron un cuestionario inicial de conocimiento (pre-test) y al final del experimento volvieron a completar el mismo cuestionario (post-test) para poder identificar el aprendizaje en el tiempo que habían tenido en su capacidad de programación de los conceptos enseñados por Alcody. El análisis de los datos revela una mejora significativa en las notas del post-test respecto al pre-test, validando la hipótesis. También se obtuvieron datos cualitativos de la observación directa de los estudiantes y entrevistas con sus profesores.

El artículo está organizado en siete secciones: la Sección 2 revisa el trabajo relacionado; la Sección 3 presenta el entorno de programación textual con el agente compañero Alcody; la Sección 4 describe el experimento realizado y los resultados obtenidos; y, finalmente la Sección 5 cierra el artículo con las principales conclusiones y líneas de trabajo futuro.

2. Trabajo relacionado

2.1. Enseñanza de la programación

Existe un interés mundial en enseñar programación en edades tempranas (Heintz et al. 2016). En Australia, Inglaterra, Estonia, Finlancia, Suecia, Korea del Sur y Macedonia la enseñanza de programación es obligatoria en Educación Primaria (Balanskat & Engelhardt, 2015; Murphy et al. 2017; García-Peñalvo et al. 2017; Llorens-Largo et al. 2017). En España, la enseñanza de programación en Educación Primaria no es obligatoria pero se espera que se pueda introducir una asignatura de Informática en el currículo de los niveles pre-universitarios como solicitan varias sociedades científicas españolas (Velázquez-Iturbide, 2018a; Velázquez-Iturbide, 2018b). En todo caso, se reconoce ya la importancia de esta enseñanza, como actividad extraescolar o transversal a las asignaturas ya existentes (de forma voluntaria).

Los principales enfoques que se están usando para enseñar programación a niños son la programación visual con Scratch (Resnick et al. 2009). Scratch es un lenguaje multimedia que permite a los niños crear programas de forma sencilla arrastrando y soltando bloques con las instrucciones sin necesidad de escribirlos. La Figura 1 muestra un ejemplo de pantalla de Scratch.

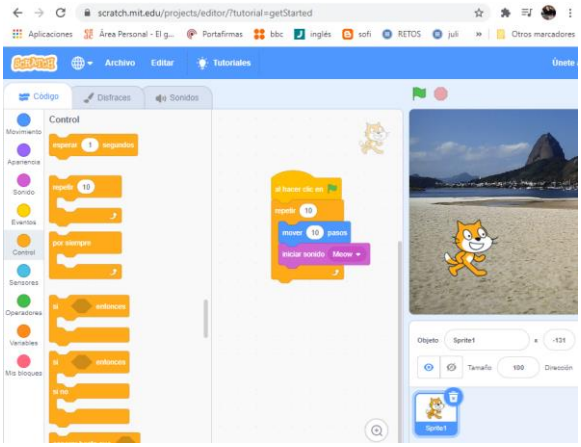

Figura 1. Ejemplo de pantallazo de Scratch

Como se puede observar en la Figura 1, en esta ocasión el niño ha puesto al personaje principal del gato (se pueden añadir otros como por ejemplo una pelota, y cada objeto tendría su programa) con un programa simple de movimiento en el que se mueve por la playa (se pueden añadir muchos fondos distintos) mientras die “miau”.

A nivel pedagógico, Scratch está basado en la teoría del construccionismo de Papert (1980). Según esta teoría, el aprendizaje es más efectivo si se interactúa con un objeto que facilite pensar (en palabras de Papert: “an object to think with”). En este caso, el programa que van construyendo los niños como bloques que encajan como piezas de un puzzle sería el objeto con el que van pensando los niños para aprender a programar.

Según la teoría del construccionismo, los estudiantes no podrán aprender solo escuchando a su profesor o copiando su solución. La idea es evitar proporcionar una solución correcta. En su lugar, se debe animar a los estudiantes a intentar llegar a su solución según su forma de pensar. Los beneficios de este enfoque con herramientas de programación visual se han publicado en varios experiencias en el campo de la programación con niños (Papavlasopoulou et al. 2019; Zainuddin et al. 2020).

No obstante, este enfoque no se limita a herramientas de programación visual y también se puede aplicar con herramientas de programación textual. La investigación sobre usar lenguajes de programación basados en el uso de bloques o basados en texto se ha explorado en los últimos años con detalle tanto en proyectos como TACCLE 3 (García-Peñalvo, 2016; García-Peñalvo & Mendes, 2018), revisiones (Howland et al. 2009; Lye & Koh, 2014), y casos de estudio (Martínez-Valdés et al. 2017; Vico, 2017).

Por último, cada vez hay más publicaciones referidas al uso de programación desenchufada (Brackmann et al. 2016; Zapata-Ros, 2018), aunque aún se sigue necesitando más investigación en estos enfoques (Kalelioğlu, 2015).

2.2. Agentes compañeros

Los agentes compañeros se pueden definir como un agente virtual que posee cierto nivel de inteligencia y autonomía, además de habilidades sociales que les permiten establecer y mantener relaciones a largo plazo con los usuarios (Lim, 2012).

El diseño de los agentes es muy importante, y debe ser atractivo para los estudiantes, ya que en caso contrario podría tener efectos negativos en su motivación y en el proceso de enseñanza-aprendizaje (Heidig & Clarebout, 2011).

Varios proyectos europeos se han centrado en estudiar los efectos de usar agentes compañeros en educación como los proyectos LIREC y EMOTE . Algunos ejemplos de agentes desarrollados son Huggable, un osito de peluche para cuidado pediátrico (Lee et al. 2009) o Emily, un agente para hablar con los niños (Adam et al. 2010). En general, estos agentes se basan en el uso de tres elementos principales para conseguir una interacción agradable con los estudiantes: personalización, adaptación a las habilidades de los niños (la interacción debe causar cierto reto al estudiante pero sin ser demasiado complicado) y tener un componente social (el agente debe tener cierta empatía y habilidades sociales para conectar con los niños).

Hay tres estrategias principales que los agentes compañeros pueden adoptar cuando estén interactuando con los niños en un entorno de aprendizaje (Roa-Seïlerm, 2016): simpatía para guiar a los niños hacia un estado positivo, alegría para celebrar su progreso e interrogativa para encontrar la causa de su estado emocional. Además de establecer las estrategias de interacción con los estudiantes, a la hora de diseñar un agente se debe implementar una estrategia de interacción conversacional con fases de inicio y final del diálogo, gestión de turnos, espera activa y como manejar las interrupciones (López-Mencía, 2011). Se podrían integrar estrategias como mostrar asombro cuando el usuario proporciona nueva información, mostrar confusión si el usuario introduce información contradictoria, introducir interjecciones afirmativas (como ajá) para mostrar que se está escuchando al usuario, y preguntar en el caso de que el usuario esté tardando mucho en hablar cuando es su turno.

La Figura 2 muestra un ejemplo de agente compañero. En este caso, se trata de “Little Mozart” del proyecto LIREC que se ejecuta en una aplicación para móviles y permite a los niños aprender a componer música.

Figura 2. Ejemplo de agente compañero: “Little Mozart” del proyecto LIREC

Como se puede observar en la Figura 2, en este caso el agente adopta una forma humana lo que supone que la situación sea bastante realista. Sin embargo, no es aconsejable, diseñar agentes tan realistas si la calidad del diálogo entre los estudiantes y los agentes no va a cumplir las expectativas que pueden crearse en los usuarios. En caso de que el diálogo no pueda cumplir las expectativas de cercanía a una conversación humano-humano, se podrían usar agentes en forma de animales o robots (Roa-Seülerm, 2016).

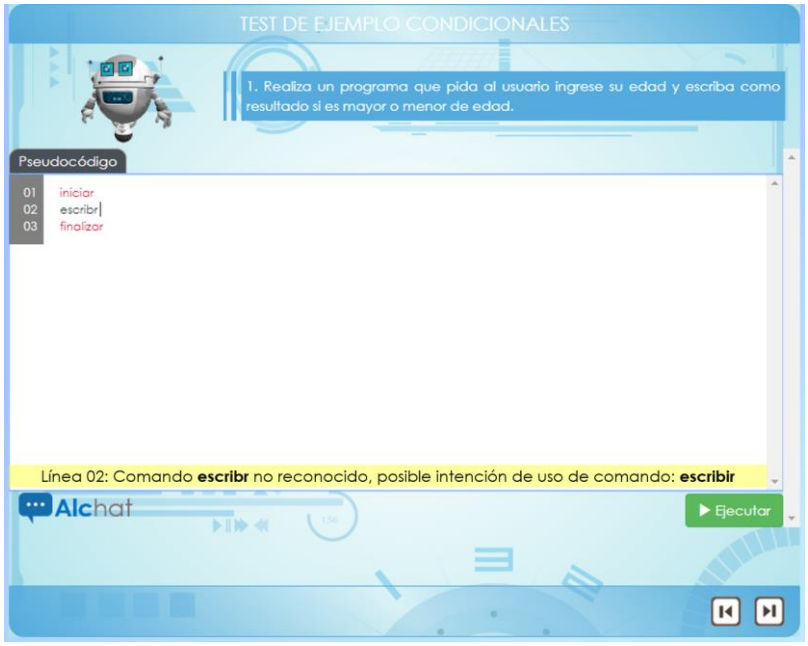

Figura 3. Ejemplo de pantalla principal de Alcody

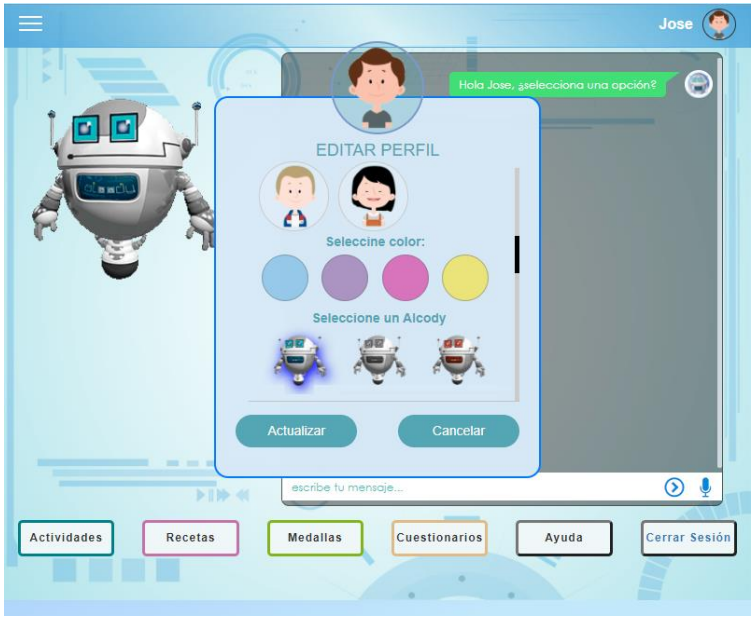

Figura 4. Opciones de Alcody

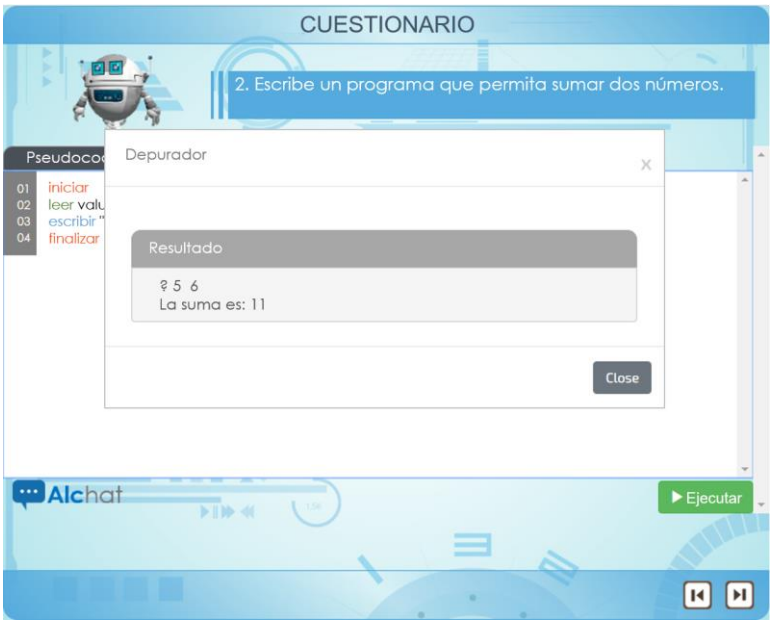

Figura 5. Ejemplo de resultado de ejecución de programa correcto

Figura 6. Ejemplo de depuración de programa incorrecto

Figura 7. Ejemplo de diálogo de Alcody para el acompañamiento emocional con recomendaciones

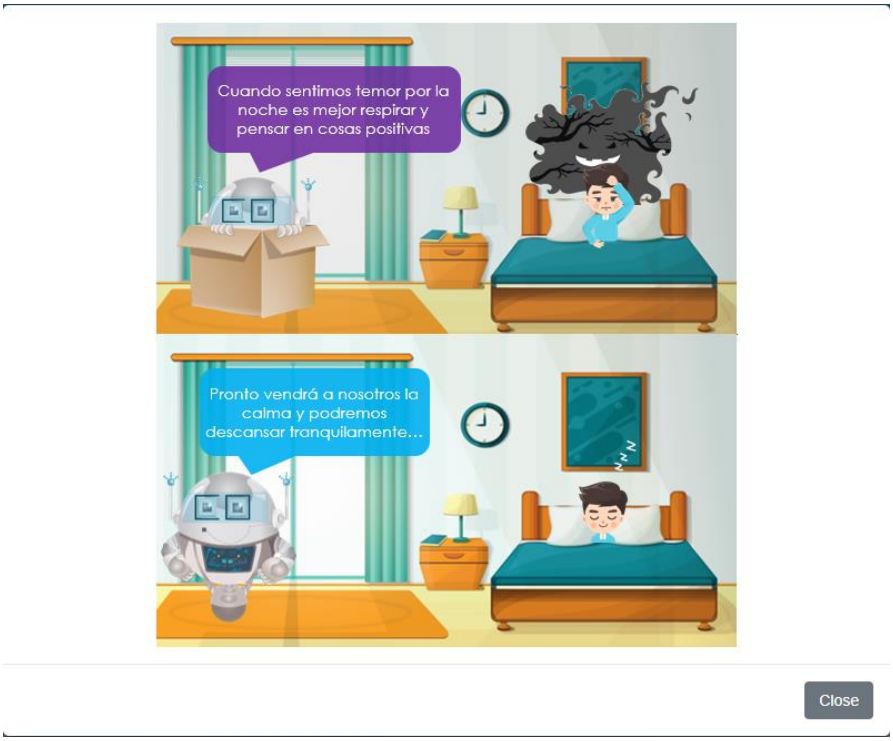

Figura 8. Ejemplo de recomendación

3. Alcody

En esta sección, se presenta Alcody que es un agente compañero para enseñar programación a niños cursando Educación Primaria co-diseñado con ellos (referencia eliminada para revisión a ciegas). La Figura 3 muestra un pantallazo de la página principal de Alcody.

Como puede observarse en la Figura 3, Alcody está representado por un robot. El color del robot y del fondo pueden ser seleccionados por el estudiante según sus gustos personales (ver Figura 4). Alcody siempre comienza el diálogo con el estudiante pidiéndole que resuelva algún problema de programación relacionado con algún concepto básico como entrada/salida, condicionales o bucles. En este caso, el programa cubre tanto entrada/salida (tiene que pedir una edad por teclado y mostrar el resultado en pantalla) como condicionales (tiene que comparar el número introducido por teclado con 18 para saber si es mayor que 18 – mayor de edad, o si es menor de 18 – menor de edad).

El estudiante debe escribir el pseudocódigo en el área de texto y pulsar el botón "Ejecutar". Si el programa es correcto, se compila y muestra el resultado (ver Figura 5). En caso contrario, se inicia el proceso de depuración para ir ayudando a corregir los errores detectados hasta llegar a la solución correcta (ver Figura 6) siguiendo un enfoque construccionista de Papert. El objeto para pensar sería el programa y la interacción con este programa, su construcción a la hora de programar, lo que permite al niño desarrollar la competencia de programar. Además, se considera fundamental el acompañamiento afectivo de Alcody al estudiante. Como se puede ver en las Figura 7 y 8, el diálogo de Alcody no se limita a conceptos de programación también pregunta al niño por su estado de ánimo y si detecta, por ejemplo, que el niño le dice que tiene pesadillas, le muestra consejos e intenta que vuelva a un estado de ánimo más positivo para poder seguir concentrado en la tarea de programar.

4 Metodología

4.1 Objetivo e hipótesis

El objetivo de este estudio es investigar si los niños, sin conocimiento previo de programación, podrán aprender a programar usando un entorno de programación basado en texto con un agente compañero como Alcody. La hipótesis es que los niños mejorarán sus notas de forma significativa usando un entorno de aprendizaje de la programación basado en texto con el agente compañero.

4.2 Participantes y contexto

Un total de 21 estudiantes de entre 10 y 12 años participaron en el experimento (10 niños y 11 niñas). Todos estaban matriculados en 6º de Educación Primaria. No habían tenido clases de programación con anterioridad pero sí iban de forma regular a la clase de informática (con conexión a Internet) para usar los ordenadores en otras asignaturas como inglés. En el aula de informática había un ordenador por niño.

El experimento se llevó a cabo en un colegio público de Madrid en España. La razón por la que se escogió este colegio es porque al no ser obligatoria la enseñanza de la programación en Educación Primaria en la Comunidad de Madrid en España se dependía de la voluntad del Director y de los profesores para usar tiempo de otras asignaturas. En el caso de este colegio, la Directora aceptó el uso de Alcody puesto que creía que podía ser beneficioso para los estudiantes y habló con el profesor de Plástica que aceptó ceder parte de su horario.

4.3 Material experimental y currículum

Cada niño fue asignado un usuario y contraseña para poder acceder a Alcody. Con sus credenciales individuales podían acceder a Alcody en su ordenador. Como currículum, Alcody tiene tres tutoriales que abarcan los conceptos tradicionales de un curso de introducción a la programación: entrada/salida, condicionales y bucles. El primer tutorial comienza con explicación del pseudocódigo y qué es Alcody, el concepto de programa, programación, variable y entrada/salida y en los

siguientes tutoriales se revisan el resto de conceptos y cómo programarlos en Alcody.

Los tutoriales están integrados dentro de Alcody para que los niños no solo los puedan hacer al comenzar a usar Alcody sino en cualquier momento en el que les surja una duda. La Figura 9 muestra un fragmento del tutorial de la clase inicial en la que Alcody está dando la bienvenida al estudiante y le empieza a enseñar el pseudocódigo con el que se pueden comunicar. Para hacer más amenos los tutoriales no están solo en texto, también tienen una grabación en voz infantil de su texto y decorados con ilustraciones que sirven también como refuerzo al contenido textual.

Además de los tutoriales, Alcody tiene grabados ejercicios de programación como los enunciados mostrados en las Figuras 3, 5 y 6, creados por profesores de programación y guardados en la base de datos de Alcody. Los ejercicios se van mostrando según el concepto que se está enseñando en clase y según el estudiante va completando los conceptos anteriores.

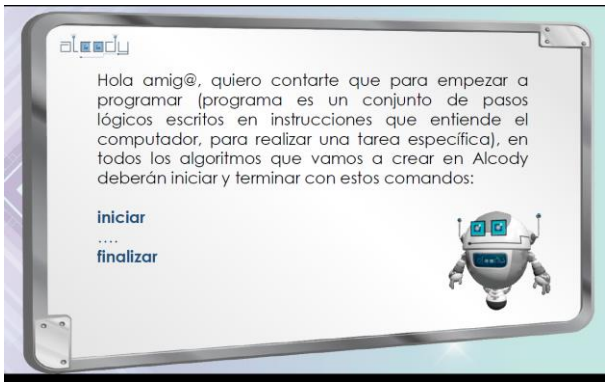

Figura 9. Fragmento del tutorial de la clase inicial de Alcody

4.4 Procedimiento

Se siguió un procedimiento pre-post test con un solo grupo experimental. Para poder identificar si los estudiantes tenían algún tipo de conocimiento previo de programación, el primer día de clase, a principios de febrero, se les pidió a todos que completasen de forma individual y con su código de alumno un pre-test con las siguientes cinco preguntas de programación:

P1 (entrada/salida). Escribe un programa que muestre "Hola" en la pantalla.
P2 (entrada/salida + variables). Escribe un programa que muestre el resultado de sumar 2 más 2.
P3. (entrada/salida + variables + condicionales) Escribe un programa que compare dos números dados e indique cuál es el mayor.
P4. (entrada/salida + variables + condicionales) Escribe un programa que pregunte una palabra en español y muestre su traducción en inglés. En la pantalla deben aparecer ambas palabras.
P5. (entrada/salida + variables + condicionales + bucles) Escribe un programa que cuente del 1 al 5.

En el pre-test se indició a los estudiantes que podían usar cualquier pseudocódigo o lenguaje de programación para resolver estos ejercicios de programación. El tiempo para realizar el test fue de una hora. La Tabla 1 recoge la rúbrica de evaluación, cada pregunta fue evaluada de 0 (nota mínima) a 1 (nota máxima) siguiendo la misma rúbrica. La nota mínima para el test fue 0 y la máxima 5, siendo necesario como mínimo 2,5 puntos para poder considerar que se había aprobado el test.

Después del pre-test, los estudiantes pudieron usar Alcody durante cuatro meses desde cualquier ordenador conectado a Internet con sus credenciales. Además, una vez al mes se hace un repaso presencial en su clase con un profesor experto en programación y en Alcody. En el aula también estaba el profesor de Plástica de los estudiantes para vigilar su comportamiento respecto al uso de los ordenadores y en general, en el aula. El mismo pre-test se usó como post-test como última actividad que realizaron los estudiantes, en las mismas circunstancias que el pre-test y durante el mismo tiempo.

Tabla 1. Rúbrica para el test de programación

Puntos	**Evaluación**
0	Respuesta en blanco o equivocada
0,25	Se usa una instrucción correcta
0,5	Todas las instrucciones usadas son correctas pero sin parámetros
0,75	Todas las instrucciones usadas son correctas con parámetros pero algunas no son necesarias
1	Código perfecto

Tabla 2. Parámetros estadísticos del experimento

	M	SD
Pre	1.69	1.07
Post	2.42	1.45

4.5 Resultados

La Tabla 2 recoge los parámetros estadísticos con la media y la desviación típica de los pre- y post- tests realizados a los 21 estudiantes que usaron Alcody durante los 4 meses que duró el experimento. Como se puede observar hay una mejora en las notas de los estudiantes después de usar Alcody, como se desprende del uso de una t-student para muestra apareadas, con un p-valor significativo ($t=2.509$, $p=0.021$). Inicialmente, debido a su falta de conocimientos de programación la media es realmente baja en la escala de 0 (mínima puntuación) a 5 (máxima puntuación) explicada en el apartado 4.4 no llegan a aprobar, y después de usar Alcody la media sube de forma significativa hasta 2.42 casi rozando el aprobado que se estableció en 2.5

Desde el punto de vista cualitativo, cuando se preguntó a los estudiantes respecto a su grado de satisfacción con el uso de Alcody en entrevistas realizadas a los estudiantes según iban usando el sistema, las opiniones recibidas fueron muy positivas con frases como:

• "Me encanta Alcody. No sabía qué era eso de programar y sonaba muy difícil, y gracias a Alcody me parece interesante y quiero aprender más"

• "Recordar las instrucciones y encontrar el orden correcto para que funcione el programa sería muy complicado sin la ayuda de Alcody. Gracias a Alcody me doy cuenta de dónde me he equivocado y me alegra mucho ver cuando funciona el programa."

• "Alcody no solo te enseña a programar, te pregunta cómo te sientes y eso me gusta mucho, es guay."

Finalmente, cuando se preguntó al profesor de Plástica su opinión respecto a lo que estaba viendo en clase con los estudiantes usando Alcody, su comentario fue que le llamaba la atención el grado de interés y la involucración tan alta que tenían los estudiantes con el sistema, intentando siempre conseguir que los programas funcionaran y respondiendo las preguntas de Alcody.

5 Conclusiones y trabajo futuro

Estudiantes de Educación Primaria, incluso sin conocimiento previo de programación, pueden aprender a programar usando un entorno de programación basado en texto con un agente compañero como Alcody. En un experimento realizado con 21 niños de entre 10-12 años que usaron Alcody durante 4 meses se registró una mejora significativa de sus notas en un pre-postest de programación con 5 ejercicios involucrando conceptos como entrada/salida, condicionales y bucles.

Es importante destacar que una limitación fuerte de este estudio es que no fue posible tener un grupo control que no usara Alcody ya que era una colegio de línea uno, con un solo grupo por curso. También estuvo muy limitado el tiempo de uso de Alcody en clase, ya que las sesiones presenciales fueron únicamente una al mes de una hora de duración, y no se pudo registrar con exactitud el tiempo que usaron Alcody fuera de clase.

La ayuda con los tutoriales y el acompañamiento emocional de Alcody fueron identificados por los estudiantes como dos de los aspectos que más preferían del agente como complemento a la enseñanza de la programación, además de ayudarles a pensar siguiendo un enfoque construccionista.

Como trabajo futuro se quiere probar Alcody durante más tiempo en clase con grupo control y test y más estudiantes, y seguir avanzando en las funcionalidades de depuración, acompañamiento emocional y gamificación.

6. Agradecimientos

Investigación financiada por los proyectos TIN 2015-66731-C2-1-R (MINECO) y P2018/TCS-4307 (e-Madrid). El proyecto e-Madrid también está financiado por los fondos FSE y FEDER. También quisiéramos agradecer a los colegios, estudiantes y profesores su colaboración sin la cual no hubiera sido posible este trabajo.

7. Referencias

Adam, C. L. Cavedon, L. Padgham, Hello Emily, how are you today? Personalised dialogue in a toy to engage children, presented at the proceedings of the workshop on companionable dialogue systems, association for computational linguistics, USA: ACL, 2010, pp. 19–24.

Adam, M. Daoud, P. Frison. 2019. Direct Manipulation versus Text-based Programming. An experiment report. ITICSE 2019, DOI: 10.1145/3304221.3319738

Balanskat, A. and K. Engelhardt, "Computing our future. Computer programming and coding Priorities, school curricula and initiatives across Europe," European Schoolnet, Brussels, Belgium, 2015.

Brackmann, C., D. Barone, A. Casali, R. Boucinha, S. Muñoz-Hernandez. 2016. Computational thinking: Panorama of the Americas. In Computers in Education (SIIE), 2016 International Symposium on (pp. 1-6). IEEE.

Feurzeig, W., S. Papert. 1967. The logo programming language. ODP-Open Directory Project.

García-Peñalvo, F.J. "A brief introduction to TACCLE 3 - Coding European Project," in 2016 International Symposium on Computers in Education (SIIE 16), F. J. García-Peñalvo and J. A. Mendes, Eds., USA: IEEE, 2016. doi: 10.1109/SIIE.2016.7751876.

García-Peñalvo, F.J. and J. A. Mendes, "Exploring the computational thinking effects in pre-university education," Computers in Human Behavior, vol. 80, pp. 407-411, 2018. doi: 10.1016/j.chb.2017.12.005.

Lye, S.Y. and J. H. L. Koh, "Review on teaching and learning of computational thinking through programming: What is next for K-12?," Computers in Human Behavior, Review vol. 41, pp. 51-61, 2014. doi: 10.1016/j.chb.2014.09.012.

Howland, K., J. Good and K. Nicholson, "Language-based Support for Computational Thinking," in 2009 Ieee Symposium on Visual Languages and Human-Centric Computing, Proceedings, R. DeLine, M. Minas and M. Erwig, Eds. Symposium on Visual Languages and Human Centric Computing-VL HCC, pp. 147-150, 2009. doi: 10.1109/vlhcc.2009.5295278.

Martínez-Valdés, J.A., J. Á. Velázquez-Iturbide and R. Hijón-Neira, "A (Relatively) Unsatisfactory Experience of Use of Scratch in CS1," in Fifth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'17) (Cádiz, Spain, October 18-20, 2017) J. M. Dodero, M. S. Ibarra Sáiz and I. Ruiz Rube, Eds. p. Article 8, New York, NY, USA: ACM, 2017. doi: 10.1145/3144826.3145356.

Vico, F. "ToolboX: Una estrategia transversal para la enseñanza de la programación en entornos educativos," ReVisión, vol. 10, no. 2, pp. 53-68, 2017.

García-Peñalvo, F.J., F. Llorens Largo, X. Molero Prieto and E. Vendrell Vidal, "Educación en Informática sub 18 (EI<18)," ReVisión, vol. 10, no. 2, pp. 13-18, 2017.

Heidig, S. and G. Clarebout, “Do pedagogical agents make a difference to student motivation and learning?”, Educ. Res. Rev. 6, 27–54, 2011.

Heintz, F., L. Mannila, T. Färnqvist, “A review of models for introducing computational thinking, computer science and computing in K-12 education”, IEEE Frontiers in Education Conference, 2016, pp. 1-9.

Jacobsen, H. 2014. Five-years-olds learn coding in schools to prepare for future labour market. EurActiv.com - EU News & policy debates, across languages.

Kalelioğlu, F. 2015. A new way of teaching programming skills to children: Code. org. Computers in Human Behavior, 52, 200-210.

Lee, J.K. W.D. Stiehl, R.L. Toscano, and C. Breazeal, "Semi-autonomous robot avatar as a medium for family communication and education", Advanced Robotics, 23(14), 1925–1949, 2009.

Lee, J.K., W.D. Stiehl, R.L. Toscano, and C. Breazeal, "Semi-autonomous robot avatar as a medium for family communication and education", Advanced Robotics, 23(14), 1925–1949, 2009.

Lim, M.Y. "Memory models for intelligent social companions". Human-computer interaction: The agency perspective (pp. 241–262). Berlin, Heidelberg: Springer-Verlag, 2012.

Llorens-Largo, F., F. J. García-Peñalvo, X. Molero Prieto and E. Vendrell Vidal, "La enseñanza de la informática, la programación y el pensamiento computacional en los estudios preuniversitarios," Education in the Knowledge Society, vol. 18, no. 2, pp. 7-17, 2017. doi: 10.14201/eks2017182717.

Lopez Mencia, B. Agentes animados personificados en sistemas interactivos: diseño y evaluación.

Doctoral thesis, Universidad Politecnica de Madrid, 2011.

Muñoz, R., T. Barcelos, R. Villarroel, I. Frango. 2017. Using Scratch to Support Programming Fundamentals. Journal on Computational Thinking (JCThink), 68.

Murphy, E., T. Crick and J. H. Davenport, "An Analysis of Introductory Programming Courses at UK Universities," The Art, Science, and Engineering of Programming, vol. 1, no. 2, p. Article 18, 2017. doi: 10.22152/programming-journal.org/2017/1/18.

Ouahbi, I., F. Kaddari, H. Darhmaoui, A. Elachqar, S. Lahmine, S. 2015. Learning basic programming concepts by creating games with Scratch programming environment. Procedia-Social and Behavioral Sciences, 191, 1479-1482.

Papavlasopoulou, S., M. N. Giannakos, and L. Jaccheri, "Exploring children's learning experience in constructionism-based coding activities through design-based research", Comput. Hum. Behav., DOI: 10.1016/j.chb.2019.01.008, 2019.

Papert, S. 1980. Mindstorms: Children, computers, and powerful ideas. New York, NY: Basic Books.

Resnick, M., J. Maloney, A. Monroy-Hernandez, N. Rusk, E. Eastmond, K. Brennan, K. 2009. Scratch: Programming for all. Communications of the ACM, 12, 52(11), 60-67.

Roa Seïlerm, N. "Designing Interaction Strategies for Companions interacting with children", Emot., Tech., and Design Emot. and Techn., pages 129-168, Academic Press, http://dx.doi.org/10.1016/B978-0-12-801872-9.00007-7, 2016.

Serna-Agudelo, B., E. Recalde-España, G. Adolfo - Beltrán, C. Cañón – Recalde. 2018. El Scratch como estrategia didáctica para desarrollar la exploración del medio en la educación inicial Fase I y II . Revista Inclusión y Desarrollo, 5 (2) 2018, 19-33.

Velázquez-Iturbide et al., "Informe del Grupo de Trabajo SCIE/CODDII sobre la enseñanza preuniversitaria de la informática," Sociedad Científica Informática de España, Conferencia de Decanos y Directores de Ingeniería Informática, España, 2018b.

Velázquez-Iturbide, J.A., "Report of the Spanish Computing Scientific Society on Computing Education in Pre-University Stages," in Proceedings TEEM'18. Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality (Salamanca, Spain, October 24th-26th, 2018), F. J. García-Peñalvo, Ed. pp. 2-7, New York, NY, USA: ACM, 2018a. doi: 10.1145/3284179.3284180.

Wing, J.M. 2006. Computational thinking. Communications of the ACM, 49(3), 33–35.

Zainuddin, Z., M. Shujahat, H. Haruna, and S. Kai Wah Chu, "The role of gamified e-quizzes on student learning and engagement: An interactive gamification solution for a formative assessment system", Comput. Educ., 145, 2020.

Zapata-Ros, M. "Computational Thinking Unplugged," Education in the Knowledge Society, vol. 20, art. 18, 2019. doi: 10.14201/eks2019_20_a18.