El desarrollo de Scratch-Jr: el aprendizaje de programación en primera infancia como nueva alfabetización

The development of Scratch-Jr: programming learning in early childhood as a new literacy

Marina Umaschi Bers
Lynch School of Education ad Human Development, Boston College
E-mail: marina.bers@bc.edu

Traducción del inglés: María Monserrat Pose
Proyecto Educación y Nuevas Tecnologías (PENT)
Facultad Latinoamericana de Ciencias Sociales (FLACSO)

Fecha de recepción: 25 de Noviembre 2023 • Aceptado: 11 de Diciembre 20231

UMASCHI BERS, M. (2023). El desarrollo de Scratch-Jr: el aprendizaje de programación en primera infancia como nueva alfabetización Virtualidad, Educación y Ciencia, 26 (14), pp. 43-62.

Resumen

Este artículo describe el trabajo realizado por la autora y su equipo de colaboradores sobre el desarrollo del lenguaje de programación gratuito, ScratchJr, para niños y niñas de 5 a 7 años. Luego, detalla su enfoque pedagógico llamado "Código como otro Lenguaje" (CAL-Coding as Another Language) que entiende la enseñanza de programación no sólo como resolución de problemas sino como una actividad expresiva que permite a los individuos manipular un sistema simbólico situado socialmente, con una gramática y sintaxis, para comunicar ideas y crear artefactos compartibles. El artículo describe ScratchJr y CAL como si fueran una plaza donde los niños aprenden jugando, inventando, creando y socializando.

Palabras clave: lenguaje de programación; pensamiento computacional; infancia; alfabetización.

Abstract

This article describes the work done by the author and her team on the development of the free programming language, ScratchJr, for children from 5 to 7 years old. Then, it details her pedagogical approach called "Code as Another Language" (CAL-Coding as Another Language), which understands the teaching of programming not only as problem solving but as an expressive activity that allows individuals to manipulate a socially situated symbolic system, with its grammar and syntax, to communicate ideas and to create shareable artefacts. The article describes ScratchJr and CAL as a playground where children learn by playing, inventing, creating and socializing.

Keywords: programming language; computational thinking; childhood; literacy.

Introducción

Las pantallas, computadoras, teléfonos, tablets y relojes son omnipresentes. Sin embargo, no todas las pantallas permiten a los usuarios realizar las mismas actividades. Mientras que algunas pantallas están diseñadas para fomentar actividades pasivas, tales como mirar publicaciones de los medios, otras invitan al juego activo. Dentro de esta última categoría, algunas pantallas alojan aplicaciones y juegos interactivos, mientras que el foco de otras está en promover la interacción social. En la primera infancia, el mundo de las pantallas está principalmente dominado por las industrias educativas y de entretenimiento ya que los niños y niñas son demasiado pequeños para involucrarse en las redes sociales.

ScratchJr es una aplicación gratuita que codiseñé con Mitch Resnick y la compañía PICO, que combina el mundo del juego y el mundo del aprendizaje, al brindar oportunidades para que las infancias se involucren como si estuvieran en una plaza infantil orientada a la programación (Bers, 2020a; 2020b; 2022). En esta plaza de programación, las infancias aprenden conceptos importantes sobre la ciencia de la computación, mientras crean sus propias animaciones e historias. La pantalla de ScratchJr se convierte en una herramienta para el aprendizaje creativo, la resolución de problemas y la expresión personal (Bers et al., 2014). La siguiente escena permite asomarse a una plaza de programación en ScratchJr:

> Anat es una estudiante de nivel inicial. Ha usado ScratchJr los últimos dos meses en el tiempo áulico dedicado al currículum "Programación como otro lenguaje" desarrollado por el grupo de investigación Dev Tech (Bers, 2019b). La semana pasada, la Sra. Feldman, la maestra de Anat, leyó un cuento sobre Marie Curie. Los niños y niñas exploraron su vida y sus descubrimientos científicos sobre radioactividad. Así, aprendieron que fue la primera mujer que ganó un premio Nobel, no una sino dos veces en dos campos científicos diferentes. Disfrutaron la historia de amor entre Marie Curie y Pierre Curie y se entristecieron con la trágica muerte de Pierre.
>
> Después de la historia, la Sra Feldman les dio a los chicos y chicas iPads y los invitó a contar la vida de Marie Curie usando ScratchJr. A Anat le encantó la idea. Decidió enfocarse en recrear el casamiento de Marie y Pierre. Decidió dibujar dos personajes: uno que parecía Marie y el otro Pierre. Los programó para moverse hacia adelante y hacia atrás, para girar y saltar mientras sonaba una música de fondo. Los invitados que eligió Anat incluían varios personajes disponibles en la biblioteca de ScratchJr, tales como pájaros, perros y flores.
>
> Anat quería que Pierre preguntase a Marie: "¿Me amas?" y que Marie respondiese: "Sí". Se dispuso a crear su programa de ScratchJr combinando los bloques para "decir", que le permiten a los personajes expresarse a través de globos de diálogo. Cuando probó su programa, Anat se percató de que Marie y Pierre hablaban al mismo tiempo, y era difícil comprender el flujo de la conversación. Empezó a experimentar con diferentes bloques hasta que descubrió lo que pensó que era una solución perfecta al problema: conectó un bloque "esperar" después de cada bloque "decir" en su programa. La función del bloque "esperar", que tiene la apariencia de la esfera de un reloj, es pausar el programa por un breve lapso. Anat ubicó el bloque "esperar" en su programa, lo que originó intervalos entre los globos de diálogo de cada personaje. Como resultado, la conversación entre Marie y Pierre se entendió.
>
> Durante el momento de la ronda, en la clase de tecnología, Anat se sintió orgullosa de compartir su proyecto con sus compañeros. Varios estudiantes quisieron saber cómo había logrado que los

> personajes hablaran entre ellos. Sonriendo de felicidad, Anat le mostró a la clase el código que había utilizado y les explicó cuidadosamente cómo había programado la cantidad de tiempo que los personajes esperarían antes de decir sus líneas. Fue un proceso que involucró la prueba y el error de forma divertida y gratificante.

Anat es una de los 35 millones de niños y niñas alrededor del mundo que aprenden a programar y a crear sus propias historias interactivas con la aplicación gratuita ScratchJr (ScratchJr-DevTech Research Group, 2020). Lanzamos ScratchJr en Julio de 2014, después de tres años de investigación sobre cómo diseñar un lenguaje de programación que fuera apropiado para la etapa del desarrollo de niños pequeños (Bers & Sullivan, 2018;2019; Bers & Resnick, 2015; Flannery et al, 2013). Nuestro objetivo fue crear una especie de plaza, pero esta vez, una de programación para que los pequeños/as pudieran expresarse por sí mismos.

ScratchJr fue lanzado, además, en un momento en donde la necesidad de una fuerza de trabajo con conocimientos tecnológicos en una economía altamente tecnologizada iba en aumento. Este incremento inspiró el desarrollo de muchas aplicaciones educativas para niños pequeños con orientación en STEM (del inglés: ciencia, tecnología, ingeniería y matemáticas), (Kafai & Burke, 2014). Según la Oficina de Estadísticas Laborales de los Estados Unidos, se proyecta que se abrirán un millón de puestos para profesiones relacionadas con la computación entre 2014 y 2024 y en menos de 10 años a partir de este momento se estima que los Estados Unidos necesitarán 1, 7 millones más de ingenieros y profesionales de la computación. Sin embargo, satisfacer las demandas del mercado laboral no fue nuestro objetivo cuando diseñamos SratchJr para niños y niñas pequeños, sino un resultado positivo inesperado.

En mi libro, *Coding as a Playground*, invitaba a los lectores a recordar las plazas a las que concurrían en su infancia (Bers, 2020a; 2020b). Las plazas proporcionan a los niños y niñas la oportunidad de explorar, inventar, simular, comunicarse, colaborar y resolver problemas con otros. La plaza es un espacio donde uno puede tomar distintas decisiones, aprender haciendo y experimentar por ensayo y error. Una plaza para programar extiende esas posibilidades al mundo digital. En ScratchJr, las infancias pueden crear e inventar historias y juegos, arte interactivo y grabaciones.

En ese sentido, contrastamos las plazas con los corralitos. Los corralitos son espacios seguros, pero que cercenan la libertad para experimentar, carecen de autonomía para explorar y carecen de oportunidades creativas. Las plazas no tienen límites mientras que los corralitos son limitados. Los plazas fomentan importantes aspectos del desarrollo humano que los corralitos impiden (Bers, 2008; 2012; 2022). Yo acuñé la metáfora "plazas vs. corralitos" ("playground vs. playpen") para proporcionar una lente para comprender las experiencias más apropiadas para el desarrollo infantil cuando se habla de tecnología (Bers, 2008; 2020b). Cuando se lanzan al mercado aparatos nuevos, robots, aplicaciones y juegos, volver a esta metáfora puede resultar útil para guiar nuestras elecciones. Esta metáfora nos permite ver más allá de los espejitos de colores que nos venden las nuevas tecnologías y enfocarnos en las oportunidades que proporcionan para el aprendizaje creativo. ScratchJr es una plaza de código en el que los niños y niñas pueden insuflar vida a sus proyectos mientras además exploran ideas poderosas de Ciencias de la Computación y desarrollan el pensamiento computacional (Bers & Kazakoff 2012; Bers, 2020b; 2021a; 2021b).

Lamentablemente, desde la perspectiva del desarrollo humano, muchas de las tecnologías actuales

para niños pequeños son corralitos y no plazas (Bers, 2019b; 2022b; 2021a; 2021b). Mientras que la idea más obvia es que algunas aplicaciones en las pantallas privan a los niños y niñas de la actividad física, la metáfora va más allá de eso. En una plaza, las infancias pueden visitar el arenero, la hamaca, el tobogán o simplemente corretear. De manera similar, cuando usan ScratchJr, los niños se involucran en toda clase de actividades más allá del código (Bers & Sullivan, 2018). Por ejemplo, pueden crear y modificar personajes en el editor de diseño, y grabar y jugar con sus propias voces y sonidos (Bers & Resnick, 2015). ScratchJr alienta a sus usuarios y usuarias a aprender experimentando, cometiendo errores, arreglando esos errores y resolviendo problemas.

Algunos juegos de computadora se comercializan como educativos porque desarrollan habilidades pre-académicas y enseñan formas, colores, letras, sonidos y números. Sin embargo, la pregunta es: ¿Estas tecnologías proporcionan las mismas oportunidades para la creatividad? ¿Qué sucede con la resolución de problemas, la exploración y la colaboración? Nuestro objetivo, cuando diseñamos ScratchJr fue proporcionar un lenguaje de programación accesible para que todos los niños y niñas pequeñas pudieran aprender a escribir código, pensar de nuevas maneras y usar la tecnología para la expresión personal. Como una plaza, ScratchJr no tiene límites y permite a los niños y niñas tomar las riendas de la exploración y la expresión creativa.

ScratchJr: una plaza para programar

Como muchos intentos de diseño creativo, ScratchJr comenzó con una pregunta: ¿Cómo podemos crear un lenguaje de programación apropiado al desarrollo infantil? Nos inspiramos en Scratch, pensado para chicos a partir de 8 años, diseñado por Mitch Resnick y su equipo en el MIT Media Lab, que es utilizado por millones de jóvenes alrededor del mundo. (Se puede visitar scratch.mit.edu para conocer más detalles) (Resnick, 2013).

Después de observar a mis tres hijos, que eran pequeños en ese momento, intentar usar el programa Scratch, me percaté de la necesidad de algunos cambios importantes en el diseño. Mientras ellos entendían los conceptos básicos de programación, les era difícil usar la interfaz. Los desbordaba la multiplicidad de opciones de comandos de programación que no podían leer o comprender. Con un adulto mediando la experiencia, podían usar Scratch satisfactoriamente, pero no era apropiado desde el punto de vista del desarrollo para que los niños pequeños lo usasen por sí mismos.

Esta situación me perturbaba. Los chicos y las chicas no necesitan de una persona adulta que los tenga de la mano en la plaza. Parte de la experiencia es explorar las instalaciones y navegar por las reglas sociales con independencia. Pueden llegar a necesitar ayuda con las tareas más desafiantes pero las plazas están diseñadas de manera tal que los chicos puedan jugar y experimentar por sí mismos. Descubrí la necesidad de un lenguaje de programación que les permitiese a los niños y niñas experimentar los beneficios evolutivos de una plaza, la libertad, la exploración y la misma sensación de dominio sin una persona adulta supervisando cada acción.

Junto con Mitch Resnick decidimos colaborar en el proyecto ScratchJr e invitamos a nuestros colegas, Paula Bontá y Brian Silverman de la Playful Invention Company (PICO) en Canadá, a unirse a nuestro equipo. La aventura comenzó en 2011 cuando recibimos una beca de la National Science Foundation para iniciar la investigación y el proceso de diseño (NSF 1118664). Asimismo, recibimos el apoyo de la Scratch Foundation que financió el ecosistema Scratch. Nos llevó tres años desarrollar

ScratchJr por completo y buscamos los mejores diseñadores para cada etapa del desarrollo, quienes proporcionaron su guía y su valioso feedback, incluyendo educadores, familias, directores, niños y niñas.

Comenzamos nuestro diseño y proceso de desarrollo observando cómo niños pequeños usaban Scratch, diseñado para chicos más grandes y notando sus dificultades (Leidl, Bers, & Mihm, 2017). Pasamos muchas horas en jardines de infantes y aulas de primer y segundo grado para poder comprender las limitaciones que encontraban los niños de 5 a 7 años, específicamente (Flannery et al., 2013). Por ejemplo, observamos que los niños y niñas pequeños tenían dificultades con los numerosos comandos ofrecidos por Scratch. Así, comprendimos desde el inicio la necesidad de simplificar y ofrecer una paleta de programación más limitada.

A lo largo de este proceso, los docentes nos señalaron que mientras los niños y niñas aprendían a leer y a escribir en inglés, lo hacían de izquierda a derecha, una direccionalidad que podría ser replicada en ScratchJr (Bers 2019b). En cambio, en Scratch se programa de arriba hacia abajo para imitar otros lenguajes de programación establecidos y más avanzados (Bers 2019b; Hassenfeld et al., 2020; Shanahan & Lonigan 2013, Vee, 2017).

Apoyados en estos descubrimientos, empezamos el diseño de nuestros primeros prototipos en ScratchJr. En cada etapa del desarrollo llevamos a cabo testeos de uso con niños y niñas pequeños, familias y educadores (Bers & Kazakoff, 2012; Bers, Govind, & Relkin, 2021). Si bien este abordaje llevó tiempo y resultó tedioso, aseguró la creación de un lenguaje de programación que cada uno de estos grupos encontrara útil, con la diversidad de sus necesidades y deseos. Trabajamos con numerosos docentes y niños/as en sesiones informales después de la jornada escolar, talleres de educadores, intervenciones experimentales en el aula y sesiones de juego en los hogares para determinar las mejores prácticas (Bers, 2021c; Bers, Govind, & Relkin, 2021; Bers & Sullivan, 2019; Csizmadia, Standl, & Waite, 2019). Adicionalmente, realizamos encuestas en línea y focus groups presenciales para obtener feedback. Todo ello proporcionó invaluable información para nuestro equipo de diseño.

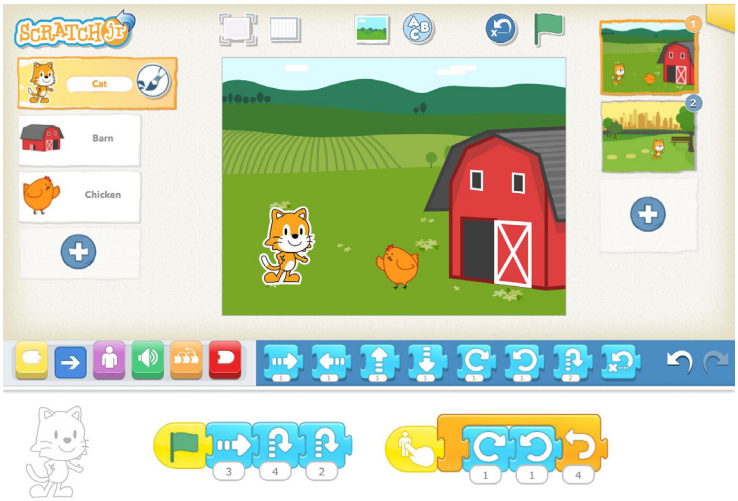
Después de una campaña de inicio, en julio de 2014, lanzamos la actual versión de ScratchJr como aplicación nativa para tablet, seguida por una versión para Android en marzo 2015. Al año siguiente, expandimos la compatibilidad a dispositivos a Chromebook e hicimos que ScratchJr estuviese disponible en variados dispositivos alrededor del mundo.

Dedicamos gran cantidad de tiempo a trabajar en colaboración con diseñadores gráficos, de manera tal que la interfaz transmitiese la misma invitación al juego que una plaza en la vida real. La paleta de colores establece un tono compatible con el juego, la gráfica es brillante y fantasiosa; y las acciones programables son divertidas. Los niños y niñas, al sentirse atraídos por la animación, pueden explorar el espectro de conceptos de programación sistemáticamente o toqueteando. También pueden acoplar los bloques gráficos de programación para hacer que sus personajes se muevan, salten, dancen y canten; modificar los personajes en el editor de diseño, crear fondos coloridos y únicos, agregar sus propias voces y sonidos e incluso sacar fotos de sí mismos para insertar en sus historias.

ScratchJr tiene una biblioteca de proyectos del usuario, un editor principal de proyecto, herramientas para seleccionar y dibujar personajes y gráficas para los fondos (figura 1) . En el centro del editor del proyecto está la página de la historia, la escena en construcción. Se pueden agregar nuevos personajes, texto y escenarios clickeando los botones grandes marcados con íconos: la silueta

de un gato, el ABC, una cadena montañosa. Las páginas que aparecen en miniaturas en el lado derecho pueden ser creadas y ejecutadas en una secuencia como una historia en varias escenas.

Figura 1. La interfaz de ScratchJr.



Fuente: elaboración propia

La paleta de instrucciones (azul) de programación se encuentra en el centro del editor. Los niños y niñas despliegan una categoría de instrucción por vez al clickear los indicadores de la izquierda. Los bloques de instrucciones de la paleta se activan al arrastrarlos al área inferior de trabajo. Los bloques encastrados crean programas que se leen y se activan de izquierda a derecha. La bandera verde (Play) y los botones rojos de "Stop" respectivamente hacen que la animación programada inicie y se interrumpa.

Los bloques de programación se organizan en seis categorías representadas por diferentes colores: bloques amarillos de desencadenamiento, bloques celestes de movimiento, bloques violetas de apariencias, bloques verdes de sonido, bloques naranjas de control de flujos y bloques rojos de finalización. Esos bloques de programación unidos como un gran rompecabezas permiten a los chicos controlar las acciones de sus personajes en la pantalla. Por ejemplo, la siguiente imagen muestra un guión de programación de un personaje saltando dos veces y luego creciendo y encogiéndose.

Figura 2. Un guión de programación en ScratchJr.

Fuente: elaboración propia

(Este guión comenzará cuando el usuario presione la bandera verde. Cuando el guión corra, el personaje correspondiente va a saltar dos veces, después crecerá dos veces y luego se encogerá dos veces hasta recuperar su tamaño original).

El diseño de la forma de los bloques impide errores sintácticos. Las partes con la forma de piezas de rompecabezas tienen propiedades visuales que se corresponden con sus propiedades sintácticas. Por ejemplo, el bloque "repetir para siempre" sólo puede aparecer al final del programa. Como nada debería seguir el comando "repetir para siempre", el lado derecho de ese bloque es redondeado para que no se pueda unir a ninguna otra pieza.

Un guión de programación corre como una secuencia de izquierda a derecha en lugar del tradicional formato de arriba hacia abajo de la mayoría de los lenguajes de programación, incluyendo Scratch. Esta elección refuerza la conciencia de la dirección de la escritura y la alfabetización en el idioma inglés (Bers 2019b; 2022). Mientras corre el guión de un personaje, la aplicación resalta cada bloque que se ejecuta, como si estuviera representando las instrucciones dadas a un personaje en el escenario.

Cuando la aplicación abre la pantalla del proyecto, los bloques de movimiento se muestran dentro de la paleta de bloques en la mitad de la pantalla. Los niños y niñas pueden arrastrar tantos bloques de movimiento como deseen desde la paleta hacia el área de programación de abajo y luego conectarlos para crear guiones. Para programar con bloques de otras categorías, podrían tocar uno de los botones clasificados por color en el lado izquierdo de la paleta. Por ejemplo, si tocan el botón violeta, los bloques de movimiento de la paleta se reemplazan con bloques de apariencia. De esta manera, tienen acceso a más de veinticinco bloques de programación, sin sentirse sobrecargados por las opciones disponibles en pantalla. El texto que muestra el nombre de cada bloque puede revelarse tocando sobre cada uno de ellos, lo que favorece el reconocimiento de las palabras (Bers, Govind, & Relkin, 2021).

Los bloques de programación proponen un espectro de conceptos de secuencias simples de movimiento a estructuras de control. Al realizar un proyecto con ScratchJr., se ponen en juego conceptos centrales de la Ciencia de la Computación, disponibles en trabajos previos (Bers, 2020a; 2020b). Además, los niños y niñas pueden usar ScratchJr para otras actividades que excedan la programación (Bers, 2022; Bers & Sullivan, 2018). Pueden editar personajes existentes o crear los propios, y pueden usar su propia voz grabada o sonidos y sus propias fotos en sus proyectos. Luego pueden incorporar esos materiales en variados medios en sus proyectos para personalizarlos y expresarse en forma creativa.

ScratchJr cuenta con un set gráfico básico en comparación a la variedad disponible en Scratch. Esta decisión obedece a nuestro lema "menos es más", a fin de simplificar la dificultad de los niños y niñas en navegar por tan vasto despliegue de opciones. Además, los alienta a crear sus propias gráficas que pueden relacionarse con temas específicos del aula. Pueden editar las imágenes incluidas o dibujar las propias en un editor de vectores gráficos escalables que viene incorporado.

ScratchJr tiene una funcionalidad llamada "la grilla" que se superpone a la escena de la animación. Se puede activar y desactivar, ya que es muy útil durante la programación pero no lo es al presentar el proyecto. La grilla fue diseñada para ayudar a niños y niñas a comprender las unidades de medida para cada bloque de programación. Sirve para guiar las unidades de medida para el movimiento lineal.

Por ejemplo, un personaje programado para "moverse hacia la derecha 10" se desliza 10 celdas de la grilla en lugar de 10 píxeles u otra medida arbitraria. La grilla es similar al cuadrante superior derecho del Sistema de Coordenadas Cartesianas, con unidades de medida discretas en lugar de continuas (Clements & Battista, 2000). Sus ejes numerados favorecen el recuento y proporcionan una marca para seguir la cuenta. Estas decisiones se tomaron para fomentar el aprendizaje y el desarrollo.

Además, algunas decisiones de diseño se tomaron en función del proceso de alfabetización (Bers, 2019b; Hassenfeld et al., 2020). La habilidad para crear cuatro "páginas" independientes e integrar texto y habla en un proyecto permite que niños y niñas creen su propio libro de historias con un principio, desarrollo y fin. Al crear estos proyectos, piensan en términos de si esto sucede, entonces sucede esto otro. Al programar con ScratchJr, comienzan a comprender los componentes básicos de una historia, al mismo tiempo que refuerzan las habilidades de secuenciación.

Trabajando con ScratchJr, descubren ideas poderosas y desarrollan habilidades que pueden aplicarse en diferentes ámbitos, tales como secuenciación, estimación, predicción, composición y descomposición (Bers & Kazakoff 2012). ¿Cuántos? o ¿Cuán lejos? son preguntas que suelen escucharse cuando los chicos y chicas usan ScratchJr. Es más, los docentes con experiencia los invitan a predecir qué sucederá cuando hagan correr cada iteración del programa, y a pensar sobre si los cambios que han hecho producirán el resultado esperado. ScratchJr posibilita feedback inmediato a la precisión de sus estimaciones y predicciones. Eso constituye uno de los aspectos valiosos del lenguaje de programación: el pensamiento computacional puede ser testeado y recibir feedback. (Bers & Sullivan, 2019; Relkin, & Bers, 2019; Grover & Pea, 2013).

Muchos de los aspectos en el diseño de ScratchJr posibilitan la resolución de problemas de bajo nivel cognitivo, pero a la vez habilitando procesos de mayor grado de complejidad; como resolución de problemas de un guión que produce resultados inesperados (Bers, 2021b; 2022; Bers et al, 2014; Bers, Govind, & Relkin, 2021; Grover & Pea, 2013). Estas decisiones del diseño desafían a las/los usuarios posibilitándoles operar con diferentes niveles cognitivos en ese proceso.

En la plaza de programación, los chicos y chicas pueden jugar por su cuenta o pueden participar en actividades organizadas (Bers 2012; 2020a; 2022). Para facilitar la enseñanza de ScratchJr, a través de los años, el grupo de investigación de DevTech ha desarrollado más de veinte unidades curriculares integrando la enseñanza de las ciencias de la computación con el desarrollo del pensamiento computacional para estudiantes principiantes, intermedios y avanzados (Bers et al, 2014).

Por ejemplo, los estudiantes principiantes exploran algoritmos a través de secuenciamientos lineales simples mientras los estudiantes intermedios pueden explorar loops de secuencias usando bloques repetidos, y los estudiantes avanzados pueden ocuparse de secuencias paralelas de dos o más programas simultáneos (Bers, 2020a; 2022; Kazakoff & Bers, 2011; 2014). Sin embargo, en cada nivel de experticia, introducimos nuevos bloques de ScratchJr. Por ejemplo, en un currículum inicial los y las estudiantes usan bloques de movimiento azules para programar un personaje para bailar o moverse alrededor del escenario. En un currículum intermedio, podrían combinar los bloques azules, con los que ya se encuentran familiarizados, con nuevos bloques verdes para la grabación de sonidos y bloques amarillos para comenzar a pulsar para crear personajes interactivos que hablen cuando sean tocados. Finalmente, en un currículum avanzado, combinan todo su conocimiento previo y agregan bloques de mensajes naranjas para programar personajes que puedan interactuar realísticamente entre

ellos, representando un juego, una conversación o una historia.

Además, lo distintivo de nuestra propuesta es la integración de la programación con otras disciplinas como Matemática, Arte, Ciencias Sociales y Alfabetización (Bers, 2022). Si bien la tecnología es importante, será la pedagogía que se use para enseñar con y sobre ScratchJr la que determinará si la experiencia se asemeja a la de una plaza o la de un corralito.

La programación como otro lenguaje

Muchas teorías pedagógicas consideran la programación como dentro o relacionada con las disciplinas STEM. De allí que la programación es considerada como una actividad de resolución de problemas que involucra el pensamiento con la abstracción y la lógica, al mismo tiempo que desarrolla el pensamiento computacional (Wing, 2006;2011; Grover & Pea, 2013).

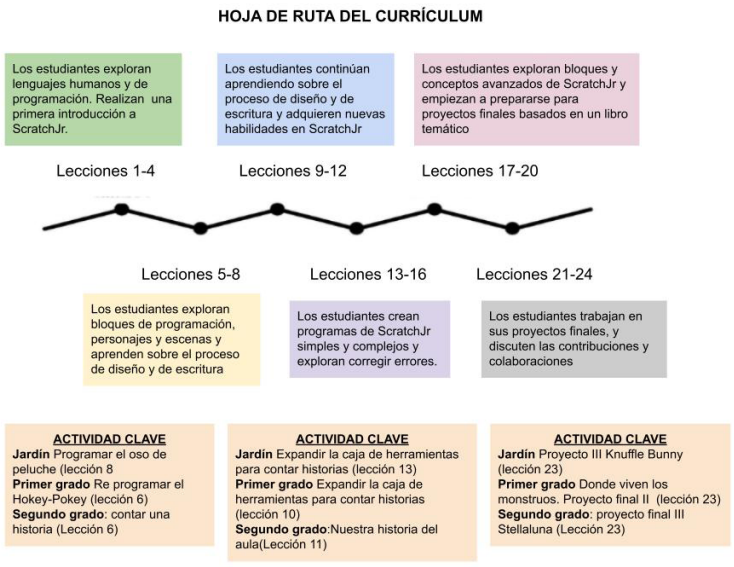
El enfoque pedagógico, al que denomino "Código como otro Lenguaje" (CAL-Coding as Another Language) entiende la programación no sólo como resolución de problemas, sino como una actividad expresiva que permite a los individuos manipular un sistema simbólico situado socialmente, con una gramática y sintaxis, para comunicar ideas y crear artefactos compartibles (Bers, 2019b; 2020b; 2021a; 2022; Bers, Govind, & Relkin, 2021). Desde esta perspectiva, los lenguajes de programación se consideran emparentados con los lenguajes escritos como herramientas de expresión.

La pedagogía CAL apoya la exploración de las similitudes y las diferencias entre los lenguajes naturales y artificiales para la creación de proyectos computacionales, a la vez que toma prestadas estrategias de enseñanza de la alfabetización tradicional para crear en la plaza de programación (Bers, 2019b; Hassenfeld & Bers, 2020). Sin embargo reconoce las diferencias significativas entre los lenguajes de programación y los naturales para propósitos expresivos (Fedorenko et al, 2019; Bers, 2019b; Bers, Govind, & Relkin, 2021).

El enfoque CAL hace foco en las prácticas compartidas (Hassenfeld et al, 2020; Hassenfeld & Bers, 2020): la creación de proyectos, ya sea a través de programación o de redacción; el proceso de diseño creativo involucrado en construir estos proyectos, la necesidad de revisarlos y repararlos a cada paso del camino; y la forma de compartir los productos finales con otras personas como una forma de expresar nuestra individualidad, nuestros intereses, pasiones e identidades (Hassenfeld & Bers, 2020). Para volver más concretas, para educadores y estudiantes, estas conexiones entre la programación y la alfabetización, desarrollamos el currículum CAL, que es gratuito y ha sido traducido al español (Bers, 2021a; Fedorenko et al, 2019). Ver https://sites.tufts.edu/codingasanotherlanguage/curricula/scratchjr/

El currículum CAL-ScratchJr, dividido en unidades para Jardín de infantes, primero y segundo grado, usa la aplicación de programación ScratchJr como un medio para explorar las similitudes entre la programación y la alfabetización (Bers, 2019a). Cada currículum de cada nivel por grado consiste en veinticuatro lecciones de 45 minutos, que involucran a los estudiantes para que desarrollen habilidades para el pensamiento computacional, la computación, la resolución de problemas y la colaboración mientras aprenden cómo crear sus propios proyectos interactivos con ScratchJr. La figura 3 muestra la hoja de ruta del currículum.

Figura 3: Hoja de ruta del currículum



Fuente: elaboración propia

Cada lección CAL- ScratchJr se compone de diversas actividades con y sin dispositivos. Incluyen canciones, círculos de tecnología, actividades ScratchJr estructuradas y desestructuradas y libros (tanto de ficción como de no ficción) (Bers & Sullivan, 2018). Por ejemplo, los libros de historias ficcionales incluyen *Donde viven los monstruos* de Maurice Sendak o *Había una anciana que se tragó una mosca* de Simms Taback; libros de no ficción que cuentan la historia de mujeres pioneras en ciencias de la computación como Ada Lovelace or Grace Hopper. Se alienta a los docentes a sustituir cualquiera de estos libros con sus favoritos, siempre que tengan una secuencia clara de eventos. Los niños y niñas crean sus propios finales para sus libros y aprenden cómo recrear las historias en modos creativos usando sus propias animaciones programadas en ScratchJr.

El currículum CAL-ScratchJr además apoya habilidades relacionadas con Matemática Fundamental, Lectura y Prácticas del Lenguaje, que se enseñan comúnmente en las aulas de primera infancia. Sin embargo, se trata de un currículum de ciencias de la computación organizado en torno al contenido y a la secuenciación de siete ideas poderosas de la Ciencias de la Computación que son apropiadas para la edad y alientan el pensamiento computacional (Bers, 2018; 2019b; Grover & Pea, 2013): sistemas de hardware/software, algoritmos, modularidad, estructuras de control, representación, depuración de programas, y proceso de diseño detallados en la Tabla 1, en la siguiente página.

El currículum CAL fue diseñado para ser flexible. La duración se puede ajustar para tornar las lecciones más largas o cortas en función de las necesidades curriculares de las diferentes escuelas. Cada lección sigue una estructura similar: una actividad de inicio que involucra un juego de baja implicación tecnológica para introducir ideas computacionales en forma lúdica, actividades de programación para reforzar habilidades, desafíos estructurados para practicar, exploraciones creativas para experimentar y expandir habilidades, juegos sin pantallas ni dispositivos para promover la interacción social y el movimiento, actividades de lectura y escritura y círculos de tecnología para compartir y reflexionar (Sullivan & Bers, 2015; 2017; 2018;). El currículum se compone de actividades individuales, en

grupos pequeños y con toda la clase.

Tabla 1: Las siete ideas poderosas, conceptos asociados y ejemplos del currículum AL-ScratchJr

Idea poderosa	Conceptos asociados	Ejemplos del currículum CAL-ScratchJr
Algoritmos	Secuenciación/orden, Organización lógica	Aprenden a programar el gatito ScratchJr en una secuencia específica para bailar el "Hokey Pokey"
Modularidad	Dividir tareas más grandes en partes más pequeñas, instrucciones.	Dividen y descomponen la canción " Si ud está loco y lo sabe" en componentes más pequeños que se pueden programar para que el gatito de ScratchJr la actúe.
Estructuras de control.	Reconocer patrones y repetición, causa y efecto	Aprenden a usar repeticiones en loop.
Representación	Representación simbólica, modelos	Aprende que cada bloque del programa se traduce en una acción única para el gatito ScratchJr.
Hardware/Software	Los dispositivos inteligentes no son mágicos, han sido diseñados por humanos.	Aprenden cómo operar una tablet.
Proceso de diseño	Resolución de problemas, perseverancia, edición / revisión	Crean un proyecto final "Alboroto salvaje" donde planifican, programan y revisan con la colaboración y el feedback de pares.
Depuración de errores	Identificación de problemas, resolución de problemas, perseverancia	Identifican problemas en sus programas y realizan lluvia de ideas con soluciones

Fuente: elaboración propia

Mientras el contenido está organizado en base a las siete ideas poderosas de las Ciencias de la Computación, se tienden conexiones explícitas en cada unidad con la alfabetización en la primera infancia (ej: el proceso de escritura, rememorar, sintetizar y secuenciar, uso de lenguaje descriptivo e ilustrativo, reconocimiento de recurso de estilo como la repetición y la anticipación y el uso de estrategias de lectura tales como la predicción, el resumen y la evaluación) (Bers, 2018; 2019b; 2020a; Papert, 1980; Vee, 2017; Unahalekhaka & Govind, 2021)

A lo largo de las lecciones CAL, cada aspecto curricular se utiliza para vincular con otro (Strawhacker & Bers, 2019). Por ejemplo, cuando los niños y niñas se enfrentan al pensamiento algorítmico, también exploran secuenciamiento y narración de historias. En el proceso de diseño se tienden conexiones activas con el proceso de escritura, y al depurar todos los programas que no funcionan, niños y niñas repasan estrategias que se asemejan a la edición sistemática de su escritura.

En la plaza, los niños y niñas no sólo desarrollan habilidades cognitivas sino además, socio-emocionales. Es posible que les resulte difícil crear proyectos inteligentes o creativos y el proceso puede resultar frustrante. Igual que en la plaza, donde los niños y niñas necesitan aprender a jugar en las barras del pasamanos sin encapricharse, los chicos que usan la plaza de programación ScratchJr necesitan aprender a manejar su propia frustración - un paso importante hacia el desarrollo de habilidades socioemocionales como la confianza en sus propias capacidades, el sentimiento de dominio y de autorregulación (Bers, 2021a; 2022). Los chicos y chicas aprenden a colaborar, negociar

conflictos, gestionar el tiempo y las emociones, usar el lenguaje para expresar sus necesidades, etc. Por lo tanto, guiados por el marco del Desarrollo de Tecnología Positiva (Positive Technological Developmwent PTD) (Bers, 2012), diseñamos CAL-ScratchJr para fomentar una multiplicidad de dimensiones del desarrollo humano: desarrollo personal, social, emocional y moral (Bers, 2022).

El marco PTD indaga en el desarrollo del niño que está creciendo en la actual era digital, acerca de sus modos para obtener información para el diseño de tecnologías, las intervenciones ricas en tecnología y los materiales curriculares, a fin de que contribuyan a que los niños y niñas se conviertan en agentes activos de su propio desarrollo y contribuyan a la sociedad. El Desarrollo de Tecnología Positiva, se enfoca en seis comportamientos positivos: comunicación, colaboración, generación de comunidad, creación de contenido, creatividad y decisiones de comportamiento.

El marco PTD se inspira en una vieja pregunta:"¿Cómo deberíamos vivir?", y guía el currículum CAL-ScratchJr no sólo para promover el dominio de las Ciencias de la Computación y las habilidades de programación sino también experiencias que permitan a los niños y niñas desarrollar un sentido de identidad, valores y propósito (Bers, 2012; 2022). En este objetivo, PTD se alinea con el Marco para el Aprendizaje para el Siglo 21, que enfatiza la integración de habilidades técnicas con una comprensión de cuestiones éticas y sociales relacionadas con el uso de nuevas tecnologías (Partnership for 21st Century Skills, 2007) y con los Estándares Estudiantiles de la Sociedad Internacional de Tecnología en Educación, que identifica la ciudadanía digital, la colaboración y la comunicación como habilidades tecnológicas fundamentales junto a habilidades relacionadas con la tecnología computacional y la solución de problemas.

Dentro del marco PTD, CAL-ScratchJr brinda oportunidades para el desarrollo socioemocional en el contexto de un ambiente de aprendizaje basado en el juego; una plaza de programación en donde tiene lugar la exploración intencional de valores éticos y morales y la promoción intencional de comportamientos positivos y fortalecimiento del carácter (Bers, 2021a; 2022).

Los niños y niñas pueden lograr progresos importantes, tanto trabajando individualmente como en grupos, y suelen compartir sus proyectos en comunidad con orgullo. Desarrollan una valoración especial por sus pares y docentes por brindarles la ayuda y el acompañamiento que necesitan para aprender programación exitosamente. Entienden la determinación, persistencia y paciencia necesarias para completar su trabajo. Aprenden a enfrentar sus propias dificultades y fallas en forma directa y se involucran activamente en resolver problemas cuando un proyecto no es exactamente lo que desearon. También aprenden a tomarse el tiempo que necesitan para aprender, incluso cuando se trata de un proceso lento. Aprenden a tolerarse cuando su proceso es diferente del de otro estudiante o cuando las cosas no salen bien todo el tiempo. Entienden que la programación involucra un constante proceso de iteración y revisión en el que se necesita flexibilidad y apertura mental. El currículum CAL ayuda a crear un ambiente para practicar valores humanos aprendiendo a usar un lenguaje de programación; para entender que nuestras acciones, como las acciones de cualquiera que crea, tienen consecuencias.

Estudios piloto del currículum CAL-ScratchJr han demostrado la habilidad de los estudiantes para aprender programación y desarrollar habilidades de pensamiento computacional (de Ruiter & Bers, 2021). Además, el análisis de proyectos con ScratchJr a lo largo de 24 lecciones muestran los diferentes bloques usados por niños y niñas y cómo la complejidad del proyecto varía (Unahalekhaka &

Bers, 2021a; 2021b). Por otra parte, las y los maestros que participaron en programas de capacitación profesional con ScratchJr acrecentaron tanto su confianza en materia tecnológica como sus habilidades para la programación, además de mejorar su habilidad para incorporar la programación en sus aulas (Bers, 2008; 2020a; 2022; Bers, Govind, & Relkin, 2021; Portelance, Strawhacker, & Bers 2015).

Descubrimos que el clima del aula era importante para alentar el éxito estudiantil frente a un desafío. Las aulas que fomentaban una actitud indulgente de cara al error y la perseverancia frente a las dificultades, eran más propicias para las plazas de programación que aquellas en las que la instrucción dejaba poco lugar a la creatividad y al ensayo y error (Portelance & Bers, 2015; Strawhacker, Lee & Bers, 2017). Mientras los niños y niñas programan diferentes proyectos, se dan cuenta gradualmente de su habilidad para encontrar soluciones intentando muchas veces, usando diferentes estrategias o pidiendo ayuda (Bers, 2010). Asimismo, en nuestra investigación, encontramos que las aulas colaborativas donde docentes y estudiantes aprenden juntos, en lugar de seguir instrucciones estrictas de un educador, obtienen mejores resultados de aprendizaje y una mayor comprensión de los conceptos de programación en ScratchJr por parte de niños y niñas (DevTech Research Group, 2020; Strawhacker, & Bers, 2019; Strawhacker, Govind, & Bers, 2022; Bers, 2021b; 2021c; 2022). Estas resultan lecciones importantes, que los niños y niñas incorporarán más allá de la plaza de programación o el aula.

Como conclusión, vamos a pensar una última vez en la plaza. La plaza está llena de niños y personas adultas: vemos padres, madres, abuelos y cuidadores, niños y niñas más grandes con sus hermanos más pequeños; individuos de todas las edades y habilidades. El juego y el aprendizaje sucede en un ambiente multigeneracional. Cada individuo está ocupado con su propia tarea y encuentra sus propias formas de entretenerse: sentarse en un banco en un costado, observar, correr, empujar la hamaca, etc. La plaza proporciona actividades y espacios para cada individuo. Esta imagen de diferentes generaciones y personas de diversos ámbitos y profesiones compartiendo una experiencia, es la que inspira la plaza de programación. Por eso diseñamos actividades y un currículum para apoyar a personas adultas y niños involucrándose en programación creativa juntos, a lo largo de días en familia con ScratchJr, y noches de programación (Bers & Sullivan, 2018; Govind, Relkin, & Bers, 2020). En nuestra investigación, encontramos que personas adultas y niños toman roles diferentes. Por ejemplo, observamos a chicos y chicas involucrarse en roles de planeamiento mientras las personas adultas adquirían roles de tutoría. Esto no resulta sorprendente: es un espejo de otras áreas de la vida y el desarrollo. Así como el Movimiento de Alfabetismo Familiar ha demostrado cómo intervenciones de lectura compartida y programas de lectura en el hogar puede aumentar el desarrollo lingüístico y cognitivo (National Early Literacy Panel, 2008), la programación en familia puede impactar potencialmente de la misma forma en el pensamiento computacional y las habilidades de programación de niños y niñas (Roque, Lin, & Liuzzi 2014; Roque, 2016). Los chicos y las chicas no aprenden únicamente en el aula sino en cada ambiente en los que viven, que proporcionan infinitas oportunidades para el desarrollo del pensamiento computacional y las habilidades de programación.

Conclusión

Los primeros años del desarrollo son un emocionante y desafiante período de crecimiento. Cuando eligen pantallas, los investigadores/as, educadores/as y diseñadores/as de políticas educativas deben considerar cuidadosamente cómo los niños y niñas las utilizarán: ¿Serán consumidores de contenidos

creados por otros o crearán su propio contenido? ¿Llevarán a cabo una exploración lúdica o tareas repetitivas? ¿Usarán una plaza o un corralito?

Con más de veinte años de investigación, mi recomendación es que necesitamos más plazas. Plazas de programación como ScratchJr ofrecen oportunidades únicas no solo para el desarrollo del pensamiento computacional y habilidades de programación sino, además, para la expresión personal creativa y el crecimiento en múltiples áreas de desarrollo.

La investigación ha demostrado que el período más crítico para el aprendizaje y el desarrollo es la primera infancia (Kazakoff & Bers, 2011; 2014). La primera infancia es un tiempo de oportunidades para despertar la curiosidad natural infantil. Sin embargo, se deben seleccionar cuidadosamente las herramientas y tecnologías para niños y niñas en este período crítico teniendo en cuenta que aún no leen ni escriben, carecen de importantes funciones ejecutivas como la atención, la memoria de trabajo, la regulación emocional, están recién aprendiendo a trabajar con otros y están ansiosos por explorar el mundo tocando, haciendo y rompiendo (Bers, 2021a; 2021b).

En mi libro Coding as a Playground (Bers, 2020a), rememoro una experiencia que tuve con una maestra de primera infancia que me preguntó si debía permitirle a su hija de 6 años usar ScratchJr por su cuenta y con cuánta frecuencia. Me hicieron esta pregunta muchas veces. Yo le contesté: "¿La dejarías leer un libro? ¿Con cuánta frecuencia? ¿La dejarías escribir una historia? ¿Con cuánta frecuencia?" Me contestó: "Depende. Depende de qué libro y depende cuanto quisiera escribir. No la dejaría escribir mientras estamos compartiendo la mesa en familia, y ciertamente no la dejaría leer algunos libros para adultos que tengo en casa. Podrían asustarla." Lógicas similares se aplican al uso de pantallas. Depende.

Así como las tabletas y celulares se están apropiando del paisaje social del mundo adulto, están adquiriendo también un rol cada vez más prominente en la vida de las infancias. En mi caso personal, como cualquier madre o educadora, siento algunas incomodidades con este hecho. No hay reemplazo para las interacciones cara a cara ni para la manipulación del mundo que nos rodea a través de objetos tangibles, Es más, me preocupa que las pantallas sean a menudo utilizadas para llenar la ausencia de adultos y pares. Por supuesto, hay muchas maneras de usarlas, y algunas son positivas, como las plazas de programación. Sin embargo, la pregunta no es si los niños y niñas deberían estar frente a las pantallas o no. La pregunta es qué están haciendo con esas pantallas.

Las plazas de programación como ScratchJr, pueden ofrecer muchas oportunidades para el aprendizaje y el crecimiento personal, la exploración y la creatividad, el dominio de nuevas habilidades y nuevas maneras de aprender. No siempre llevamos a los niños y niñas a la plaza. Hay otros lugares que visitar y otras habilidades que desarrollar. Pero cuando efectivamente vamos a la plaza, queremos que sea un espacio apropiado para el desarrollo.

El enfoque de la plaza de programación que se propone aquí, mueve la discusión más allá de la visión tradicional de la programación como una habilidad técnica. La programación es una forma de alfabetización. Como tal, invita a nuevas maneras de pensar y tiene la habilidad de producir un artefacto separado de su creador, con su propio significado. Hay un productor con una intención, con una pasión, con un deseo de comunicar algo. La programación, como la escritura, es un medio de expresión humana. A través de este proceso expresivo, aprendemos a pensar, sentir y comunicar de nuevas maneras.

En la plaza de programación, las niñas y niños pequeños crean sus propios proyectos para comunicar ideas y expresar quiénes son. Se vuelven productores, y no meros consumidores de productos tecnológicos. Necesitan herramientas apropiadas para el desarrollo como ScratchJr. Con esta herramienta, se involucran en la solución de problemas y la narración de historias, desarrollan habilidades de secuenciación y pensamiento algorítmico. Navegan a lo largo del proceso de diseño desde una idea inicial hasta un producto que puede ser compartido con otros, y se vuelven orgullosos de su trabajo. En la plaza de programación, el aprendizaje es divertido. Los niños y niñas pueden ser ellos mismos y pueden explorar en forma lúdica nuevos conceptos e ideas, así como desarrollar nuevas habilidades. Así como los niños y niñas deben aprender a levantarse por sí mismos luego de caerse en la plaza, tienen que aprender a hacer lo mismo en el aula y en la programación: pueden caerse y empezar todo de nuevo otra vez.

En esta plaza de código, los niños y niñas encuentran ideas poderosas de la Ciencias de la Computación que son útiles no solo para futuros programadores e ingenieros sino para todas las personas. La programación es una manera de alfabetizarse en el siglo XXI, igual que la lectura y la escritura. Necesita empezar a desarrollarse tempranamente. En la actualidad, las personas que pueden producir en las tecnologías digitales, y no sólo consumir, serán artífices de su propio destino. La alfabetización es una manera de empoderar a los seres humanos. Las personas que saben leer y escribir pueden hacer escuchar sus voces. Aquellas que no, quedan sin participación. ¿Se aplicará también a las que no puedan programar, a aquellas que no puedan pensar computacionalmente?

Es nuestra responsabilidad introducir a los niños y niñas a la programación y al pensamiento computacional cuando son pequeños. Sabemos que, como una forma de alfabetización, la programación abrirá puertas, muchas de las cuales no pueden ser anticipadas del todo en la actualidad. También sabemos que estos jóvenes programadores son aún niños. Por lo tanto, no es suficiente copiar pedagogías, modelos de Educación en Ciencias de la Computación o lenguajes de programación diseñados para niños más grandes, que no son apropiados para el desarrollo de los más pequeños.

Esta propuesta ofrece tecnologías y currículas específicamente diseñadas para niños y niñas pequeños, que consideran sus necesidades cognitivas, sociales y emocionales. Nuestro objetivo es integrar la programación y el pensamiento computacional a través de prácticas educativas preexistentes en primera infancia, tales como la alfabetización, de manera tal que niños y niñas puedan desarrollar las nuevas alfabetizaciones del siglo XXI y aprender a expresarse tanto con lenguajes naturales como artificiales.

Referencias bibliográficas

BERS, M. U. (2008). Blocks to robots learning with technology in the early childhood classroom. Nueva York, NY: Teachers College Press.

BERS, M. U. (2012). Designing digital experiences for positive youth development: From playpen to playground. Cary, NC: Oxford.

BERS, M. U. (2018). Coding and computational thinking in early childhood: the impact of ScratchJr in Europe. European Journal of STEM Education, 3(3), 08. doi:10.20897/ejsteme/3868

BERS, M. U. (2019a). Coding as another language In C. Donohue (Ed.), Exploring key issues in early childhood

and technology: Evolving perspectives and innovative approaches (pp. 63–70). Nueva York, NY: Routledge.

BERS, M. U. (2019b). Coding as another language: a pedagogical approach for teaching computer science in early childhood. Journal of Computers in Education, 6(4), 499-528.

BERS, M. (2020a). Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom, Second Edition. Nueva York, NY: Routledge Press.

BERS, M. (2020b). Playgrounds and Microworlds: Learning to Code in Early Childhood in Designing Constructionist Futures: The Art, Theory and Practice of Learning Designs editado por Nathan Holbert, Matthew Berland and Yasmin B Kafai.

BERS, M. (2021a). Coding, robotics and socio-emotional learning: developing a palette of virtues PIXEL-BIT. Revista de Medios y Educación, 62, 309-322.

BERS, M. (2021b). From Computational Thinking to Computational Doing. In M. U. Bers (Ed.), Teaching Computational Thinking and Coding to Young Children. IGI Global. http://doi:10.4018/978-1-7998-7308-2

BERS, M. (Ed.). (2021c). Teaching Computational Thinking and Coding to Young Children. IGI Global. http://doi:10.4018/978-1-7998-7308-2

BERS, M. U. (2022). Beyond Coding: How Children Learn Human Values through Programming. Cambridge, MA: MIT Press.

BERS, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum, Computers & Education, 72, 145-157.

BERS, M., Govind, M., & Relkin, E. (2021) Coding as Another Language: Computational Thinking, Robotics, and Literacy in First and Second Grade. in Ottenbreit-Leftwich, A., & Yadav, A. (2021). Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions. ACM and the Robin Hood Learning + Technology Fund, Nueva York, NY

BERS, M. & Kazakoff, E. (2012).Developmental technologies: Technology and human development. In Lerner, R.M., Easterbrooks, M.A., Mistry, J., & Weiner, I.B. (Eds.) Handbook of Psychology, Developmental Psychology (pp. 639-657).

BERS, M. U., & Sullivan, A. (2018). ScratchJr coding cards: creative coding activities. No Starch Press.

BERS, M. U. & Sullivan, A. (2019). Computer science education in early childhood: The case of ScratchJr. Journal of Information Technology Education: Innovations in Practice, 18, 113-138.

BERS, M.U. & Resnick, M. (2015). The Official ScratchJr Book. San Francisco, CA: No Starch Press

CENTURY, J., FERRIS, K. A., y ZUO, H. (2020). Finding time for computer science in the elementary school day: a quasi-experimental study of a transdisciplinary problem-based learning approach. International Journal of STEM Education, 7(1).

CHALL, J. S. (1983). Literacy: Trends and explanations. Educational Researcher, 12(9), 3-8.

CLEMENTS, D. H., y BATTISTA, M. T. (2000). Designing effective software. Handbook of research design in mathematics and science education, 761-776.63

DEVTECH RESEARCH GROUP (2020). Coding as Another Language (CAL): Teaching programming as a literacy of the 21st century. https://sites.tufts.edu/codingasanotherlanguage/

FEDORENKO, E., IVANOVA, A., Dhamala, R. y BERS, M.U. (2019). The Language of Programming: A Cognitive Perspective. Trends in Cognitive Sciences, 23(7), 525-528. doi:10.1016/j.tics.2019.04.010

FLANNERY, L.P., KAZAKOFF, E.R., BONTÁ, P., SILVERMAN, B., BERS, M.U., y 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). Nueva York, NY: ACM. doi:10.1145/2485760.2485785

GOVIND, M. y BERS, M. U. (2020). Family Coding Days: Engaging Children and Parents in Creative Coding and Robotics. Connected Learning Summit, Cambridge, MA.66

GROVER, S., y PEA, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational researcher, 42(1), 38-43.

HASSENFELD, Z. R. y BERS, M. U. (2020). Debugging the Writing Process: Lessons from a Comparison of Students' Coding and Writing Practices. The Reading Teacher, 73(6), 735-746. doi:10.1002/trtr.1885

HASSENFELD, Z. R., GOVIND, M., DE RUITER, L. E., y BERS, M. U. (2020). If You Can Program, You Can Write: Learning Introductory Programming Across Literacy Levels. Journal of Information Technology Education: Research, 19, 65-85. doi: 10.28945/4509

KAFAI, Y. B., y BURKE, Q. (2014). Connected code: Why children need to learn programming. Cambridge, MA: The MIT Press.

KAZAKOFF, E.R., y BERS, M.U. (2011, April). The Impact of Computer Programming on Sequencing Ability in Early Childhood .Paper presented at American Educational Research Association Conference (AERA), New Orlean, Louisiana.

KAZAKOFF, E. R., y BERS, M. U. (2014). Put your robot in, Put your robot out: Sequencing through programming robots in early childhood. Journal of Educational Computing Research, 50(4), 553-573.

KAZAKOFF, E. R., SULLIVAn, A., y BERS, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. EarlyChildhood Education Journal, 41(4), 245-255.68

LEIDL, K. D., BERS, M. U., y MIHm, C. (2017). Programming with ScratchJr: a review of the first year of user analytics. In Conference Proceedings of International Conference on Computational Thinking Education. Wanchai, Hong Kong.

NAEYC (2009). Developmentally Appropriate Practice in Early Childhood Programs Serving Children From Birth Through Age 8. www.naeyc.org/files/naeyc/file/positions/PSDAP.pdf.

PARTNERSHIP FOR 21ST CENTURY SKILLS. (2007). The intellectual and policy foundations of the 21st century skills framework. http://www.youngspirit.org/docs/21stcentury.pdf

PAPERT, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas. Nueva York, NY: Basic Books, Inc.

PORTELANCE, D.J., y BERS, M.U. (2015). Code and Tell: Assessing young children's learning of computational thinking using peer video interviews with ScratchJr: Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15). Medford, MA, June 21-25. Nueva York, NY: ACM.

PORTELANCE, D.J., STRAWHACKER, A., y BERS, M.U. (2015). Constructing the ScratchJr programming language in the early childhood classroom, International Journal of Technology and Design Education, 1-16.

doi:10.1007/s10798-015-9325-0

RELKIN, E. y BERS, M. U. (2020). Exploring the Relationship Between Coding, Computational Thinking and Problem Solving in Early Elementary School Students. Annual Meeting of the American Educational Research Association (AERA), San Francisco, CA

RELKIN, E. y BERS, M. U. (2021). Factors Influencing Learning of Computational Thinking Skills in Young Children. Virtual Annual Meeting of the American Educational Research Association (AERA).

RESNICK, M. (2013, May 8). Learn to Code, Code to Learn. EdSurge. https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn

ROQUE, R. (2016). Family Creative Learning: Designing Structures to Engage Kids and Parents as Computational Creators. In Peppler, K., Kafai, Y., & Halverson, E. (Eds.) Makeology in K-12, Higher, and Informal Education. Nueva York, NY: Routledge.

ROQUE, R., LIN, K., y LIUZZI, R. (2014). Engaging Parents as Creative Learning Partners in Computing, Exploring the Material Conditions of Learning, 2, 687-688.

SCRATCHJR – DEVTECH RESEARCH GROUP. (2020). https://sites.tufts.edu/devtech/research/scratchjr/

SHANAHAN, T., y LONIGAN, C. J. (Eds.). (2013). Early childhood literacy: The national early literacy panel and beyond. Towson: MD: Paul H. Brookes Publishing Company.

STRAWHACKER, A., y BERS, M. U. (2015). "I want my robot to look for food": Comparing Kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces. International Journal of Technology and Design Education, 25(3), 293-319. doi:10.1007/s10798-014-9287-7

STRAWHACKER, A., GOVIND, M., y BERS, M. U., (2022, April 21-26). Understanding the Experiences of Early Childhood Professionals' Navigation of Remote Teaching and Learning With Technology [Paper Presentation]. American Educational Research Association (AERA) Annual Meeting. San Diego, CA. https://www.aera.net/Publications/Online-Paper-Repository/AERA-Online-Paper-Repository

STRAWHACKER, A., LEE, M., y BERS, M. U. (2018). Teaching tools, teachers' rules: exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. International Journal of Technology and Design Education. doi:10.1007/s10798-017-9400-9

STRAWHACKER, A., LEE, M., CAINE, C., y BERS, M.U. (2015). ScratchJr Demo: A coding language for Kindergarten: Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15). Medford, MA, June 21-25. Nueva York, NY: ACM.

STRAWHACKER, A., PORTELANCE, D., LEE, M., y BERS, M.U. (2015). Designing Tools for Developing Minds: The role of child development in educational technology: Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15). Medford, MA, June 21-25. Nueva York, NY: ACM.

UNAHALEKHAKA, A. y GOVIND, M. (2021). Examining Young Children's Computational Artifacts. In M. Bers (Ed.) Teaching Computational Thinking and Coding to Young Children (pp.265-294). IGI Global. doi: 10.4018/978-1-7998-7308-2

UNAHALEKHAKA, A., y BERS, M. (2021a). Evaluating Young Children's Creative Coding: Rubric Development and Testing for ScratchJr Projects. [Under Review]

VEE, A. (2017). Coding literacy: How computer programming is changing writing. Cambridge, MA: The MIT

Press.

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

WING, J. M. (2011). Research notebook: Computational thinking—What and why. The link Magazine, 6, 20-23.73