Hemos leído

Cómo conseguir que la programación sea accesible a estudiantes con discapacidades visuales: examen de la bibliografía[1]

Making programming accessible to learners with visual impairments: a literature review

A. Hadwen-Bennett, S. Sentance, C. Morrison

Resumen

Aprender a programar puede ser difícil, y, en el caso de estudiantes con deficiencias visuales (DV), se presentan numerosos obstáculos adicionales al proceso de aprendizaje. Muchos entornos de programación modernos resultan inaccesibles a este tipo de estudiantes, puesto que es difícil o imposible interactuar con el entorno en cuestión mediante un lector de pantalla. Tras un examen de la bibliografía, se han podido detectar varias estrategias cuyo propósito es conseguir que los estudiantes con deficiencia visual tengan acceso a la enseñanza de la programación. Dichas estrategias se pueden dividir, a rasgos generales, en las categorías siguientes: retroalimentación auditiva y táctil, accesibilidad de los lenguajes basados en texto, accesibilidad de los lenguajes basados en bloques, y objetos físicos. Una cuestión que encontramos con frecuencia en la bibliografía es la dificultad a la que se enfrentan los estudiantes con deficiencias visuales a la hora de lograr una comprensión de la estructura general de un código informático. Gran parte de las investigaciones realizadas hasta la fecha en esta materia se centran en la evaluación de aquellas intervenciones que se dirigen a estudiantes de Educación Secundaria y universitaria con deficiencias visuales, prestándose una atención relativamente escasa

1 Publicado en la revista *International Journal of Computer Science Education in Schools, vol. 2, n.º 2, abril 2018* [formato PDF] con una licencia Creative Commons CC BY 4.0. Traducido por José Luis de Aragón Mari con permiso de los autores. DOI: 10.21585/ijcses.v2i2.25.

Hadwen-Bennett, A., Sentance, S., y Morrison, C. (2019). Cómo conseguir que la programación sea accesible a estudiantes con discapacidades visuales: examen de la bibliografía. *Integración: Revista digital sobre discapacidad visual, 74*, 127-150.

a los procesos de aprendizaje que afectan a los alumnos con deficiencias visuales. Además, la mayor parte de los trabajos de investigación se refieren a los lenguajes basados en texto, pese a que la mayoría de los cursos de introducción a la programación para estudiantes de Educación Primaria utilizan lenguajes basados en bloques. Por lo tanto, existe una necesidad urgente de profundizar en la investigación sobre estrategias potenciales que permitan iniciar a los alumnos de Educación Primaria con deficiencias visuales en la programación, así como en los procesos de aprendizaje relacionados con esta cuestión.

Palabras clave

Deficiencias visuales. Enseñanza de la programación. Programación física. Necesidades especiales.

Abstract

Programming can be challenging to learn, and for visually impaired (VI) learners, there are numerous additional barriers to the learning process. Many modern programming environments are inaccessible to VI learners, being difficult or impossible to interface with using a screen reader. A review of the literature has identified a number of strategies that have been employed in the quest to make learning to program accessible to VI learners. These can be broadly divided into the following categories; auditory and haptic feedback, making text-based languages (TBLs) accessible, making block-based languages (BBLs) accessible and physical artefacts. A common theme among the literature is the difficulty VI learners have in gaining an understanding of the overall structure of their code. Much of the research carried out in this space to date focuses on the evaluation of interventions aimed at VI high-school and undergraduate students, with limited attention given to the learning processes of VI learners. Additionally, the majority of the research deals with TBLs, this is despite the fact that most introductory programming courses for primary learners use BBLs. Therefore, further research is urgently needed to investigate potential strategies for introducing VI children in primary education to programming and the learning processes involved.

Key words

Visual impairments. Teaching to program. Physical programming. Special needs.

Hadwen-Bennett, A., Sentance, S., y Morrison, C. (2019). Cómo conseguir que la programación sea accesible a estudiantes con discapacidades visuales: examen de la bibliografía. *Integración: Revista digital sobre discapacidad visual, 74*, 127-150.

1. Introducción

La introducción de la informática en el plan de estudios nacional inglés en 2014 trajo consigo la exigencia de que los estudiantes de Educación Primaria aprendieran los conceptos básicos de la programación a partir de los 5 años de edad (Department for Education, 2014). La programación puede ser difícil de aprender, y los estudiantes con discapacidades visuales se enfrentan a muchas barreras adicionales durante el proceso de aprendizaje. Muchos entornos de programación modernos resultan inaccesibles para los estudiantes con discapacidades visuales, ya que es difícil o imposible interactuar con el entorno en cuestión utilizando un lector de pantalla (Baker[, Milne y Ladner], 2015; Stefik et al., 2011),[2] y las interfaces de usuario a menudo emplean representaciones con un alto componente gráfico (Ludi, 2013). Kane y Bigham (2014) establecieron los siguientes criterios con respecto al desarrollo de entornos que faciliten el aprendizaje de la programación a niños con deficiencias visuales:

- «Los instrumentos de programación deben ser accesibles para el estudiante, y han de funcionar con la tecnología asistencial que este utilice».

- «El estudiante debe realizar tareas de programación que mantengan su interés y que originen una respuesta (*feedback*) alentadora». (Kane y Bigham, 2014; 257).

Este examen bibliográfico se propone ofrecer una visión general y crítica sobre las distintas estrategias que se han venido utilizando para conseguir que el aprendizaje de la programación resulte accesible para estudiantes con deficiencias visuales. Además, se indicarán y comentarán aquellos ámbitos en los que se requiere continuar las investigaciones.

2. Metodología

Esta reseña examina documentos procedentes de revistas arbitradas (evaluadas por expertos) que se publicaron en el periodo comprendido entre el año 2000 y noviembre de 2017. Se tuvo acceso a estos estudios mediante la búsqueda en bases de datos de investigación, así como recurriendo al rastreo de citas. Se efectuaron búsquedas

2 En la versión original en inglés, la cita aparece como «Stefik et al., 2011», aun a pesar de ser la primera vez que se menciona. Habida cuenta de que en las referencias bibliográficas aparecen dos obras de 2011 encabezadas por Stefik y otros autores, se ha optado por no desarrollar esta cita para evitar posibles errores [N. del ed.].

en las siguientes bases de datos: ACM Digital Library, Taylor and Francis, IEEE, Eric y Wiley Online Library.

Inicialmente, se utilizaron los términos de búsqueda «persona con discapacidad visual», «programación» y «educación», y, a continuación, se emprendieron búsquedas adicionales en las que se hizo uso de términos de búsqueda alternativos con significados similares. En la Tabla 1 se ofrece un resumen de dichos términos.

Tabla 1. Resumen de términos de búsqueda

Término de búsqueda	Alternativas
Persona con discapacidad visual	Invidente, deficiencias visuales
Programación	Codificación, desarrollo de programas
Educación	Aprendizaje, estudiantes, colegio

Una vez elaborada una lista breve de artículos, se siguieron los siguientes criterios para decidir si un determinado artículo debiera incluirse o no en el examen bibliográfico:

- Se incluyeron aquellos artículos que tenían un enfoque pedagógico, aunque también se recuperó un número reducido de artículos que aportaban información contextual.

- Se incluyeron aquellos artículos que aparecían en publicaciones académicas arbitradas, es decir, sometidas a revisión por pares.

- Se incluyeron artículos publicados a partir de 2000. La única excepción fue un artículo que se cita con frecuencia y que, por lo tanto, proporciona información contextual.

Un examen más detallado de la bibliografía puso de manifiesto cuatro temas principales: cómo hacer accesibles los lenguajes basados en texto, cómo hacer accesibles los lenguajes basados en bloques, objetos físicos y, también, retroalimentación (*feedback*) auditiva y táctil. Cada uno de estos temas se estudia en los subapartados siguientes. En el Apéndice se ofrece un panorama general de la bibliografía, con referencias cruzadas relativas a cada uno de los temas.

3. Visión general de la bibliografía

3.1. Cómo hacer accesibles los lenguajes basados en texto

3.1.1. Accesibilidad de los entornos de programación

Una encuesta realizada entre programadores experimentados en el campo de la discapacidad visual pone en evidencia que muchos entornos de programación o bien no son completamente compatibles con los lectores de pantalla, o bien ofrecen una experiencia de navegación complicada, al utilizar, exclusivamente, señales auditivas, lo que hace que resulten inaccesibles para muchos programadores con deficiencias visuales (Albusays y Ludi, 2016). Por ejemplo, Eclipse presenta diversas ventanas con pestañas a las que se puede acceder mediante atajos de teclado. Sin embargo, el procedimiento consiguiente consume mucho tiempo cuando se depende de indicadores auditivos (Cheong, 2010). Además, los entornos operativos BricxCC y Robot C, que han sido diseñados para la programación de robots Lego Mindstorms, no son plenamente compatibles con JAWS (un lector de pantalla muy utilizado) (Ludi, 2013). Si bien Visual Studio (2010) es técnicamente accesible, no produce sonido alguno que indique al usuario cuándo se produce el desplazamiento de una pestaña a otra (Stefik et al., 2011).

Un planteamiento adoptado para solucionar el problema de la inaccesibilidad de los entornos de programación consiste en la utilización conjunta de un editor de textos estándar y de un lector de pantalla (Bigham[, Aller, Brudvik, Leung, Yazzolino y Ladner], 2008; Cheong, 2010; Kane y Bigham, 2014). Este enfoque tiene un inconveniente: la pérdida de las herramientas de depuración que se utilizan comúnmente en la mayoría de los entornos de programación actuales. También se han desarrollado herramientas para mejorar la accesibilidad de los entornos de programación. Así, por ejemplo, Wicked Audio Debugger (WAD) se elaboró para actuar junto con el popular entorno de programación Visual Studio, ayudando a los programadores con discapacidad visual durante el proceso de depuración (Stefik[, Alexander, Patterson y Brown], 2007).

Una estrategia alternativa consiste en el desarrollo de entornos de programación accesibles. Ejemplo de esto es JavaSpeak, que se desarrolló como una herramienta de ayuda para el aprendizaje de la programación en Java con destino a estudiantes universitarios con deficiencias visuales (Francioni y Smith, 2002; Smith[, Francioni y Matzek], 2000). Se basa en el concepto EmacSpeak (Raman, 1996), que presenta una

interfaz de voz dirigida a programadores experimentados. A diferencia de EmacSpeak, JavaSpeak se ha diseñado para estudiantes universitarios que están aprendiendo a programar, y permite a dichos estudiantes experimentar el código con el que trabajan con diferentes niveles de detalle y precisión. Si bien se ha descrito el proceso de desarrollo del entorno JavaSpeak, no existe constancia de que se haya evaluado la utilización de esta herramienta.

De forma más reciente, se ha desarrollado el entorno de programación JBrick para hacer accesible la programación de robots Lego Mindstorms (Ludi, 2013). El lenguaje NXC (Not eXactly C) se ha utilizado, junto con el entorno de programación BricxCC, en programas de extensión que permitan a estudiantes con deficiencias visuales programar robots Lego Mindstorms (Dorsey[, Chung y Howard], 2014; Ludi y Reichlmayr, 2011). Sin embargo, el entorno de programación BricxCC no es completamente compatible con JAWS (un lector de pantalla muy difundido). JBrick se desarrolló como alternativa a BricxCC, y es compatible con lectores de pantalla y teclados en braille de uso común, permitiendo la localización sencilla de códigos al consultar el número de línea, y proporcionando retroalimentación visual y auditiva (Ludi[, Ellis y Jordan], 2014).

3.1.2. Accesibilidad de los lenguajes de programación

Otra cuestión de importancia es la elección de un lenguaje de programación. Muchos lenguajes de uso frecuente, como C y Java, recurren muy a menudo a caracteres no alfanuméricos, como corchetes y llaves, cuyo manejo puede resultar laborioso cuando se utiliza un lector de pantalla. Además, la sintaxis compleja de muchos lenguajes puede aumentar la frecuencia de errores tipográficos y la dificultad de las tareas de depuración. Son preferibles aquellos lenguajes, como Ruby, que utilizan principalmente texto y establecen límites al número de símbolos no alfanuméricos, ya que es menos probable que originen problemas al manejarse lectores de pantalla (Kane y Bigham, 2014). En su estudio, Kane y Bigham también se interesaron por Python, ya que cumple la mayoría de los criterios antes indicados. Sin embargo, también utiliza espacios en blanco, lo que puede originar confusiones cuando se maneja con uno de tales lectores. Durante la realización del estudio, que tuvo lugar en el transcurso de una semana, con la participación de 12 estudiantes con deficiencias visuales, Kane y Bigham observaron que los estudiantes eran capaces de elaborar programas en Ruby, aunque los errores de pronunciación de algunos términos por parte del lector de pantalla provocaron pequeños problemas.

Existen lenguajes basados en texto que se han diseñado de forma específica pensando en usuarios con discapacidad visual, como, por ejemplo, APL (Audio Programming Language), desarrollado por estudiantes con deficiencias visuales para estudiantes con deficiencias visuales (Sánchez y Aguayo, 2006). APL presenta un conjunto reducido de comandos a los que se puede acceder, y que pueden seleccionarse, mediante una lista de comandos circular, sin que haya necesidad de memorizar dichos comandos. El resultado de un pequeño estudio sobre la utilización de APL indica que este lenguaje permite al estudiante comprender y aplicar conceptos de programación.

En 2011, Stefik et al. realizaron un estudio exploratorio para evaluar el entorno de programación accesible Sodbeans, junto con el lenguaje de programación Hop, desarrollado por ellos mismos. Sodbeans está dirigido a estudiantes de enseñanza media y bachillerato, y utiliza avisos acústicos para facilitar la navegación. También cuenta con un depurador auditivo para el lenguaje de programación Hop. Los resultados de la evaluación revelaron un aumento en la confianza en las propias capacidades por parte de los estudiantes tras su participación en un taller de trabajo de programación en el que se utilizaron tanto Sodbeans como Hop.

El lenguaje de programación Hop ha experimentado un desarrollo ulterior, hasta convertirse en Quorum, un lenguaje diseñado para el público en general, pero que sigue siendo accesible a estudiantes con deficiencias visuales (Stefik et al., 2011). El desarrollo de Quorum se sustenta en estudios empíricos que investigan el carácter intuitivo de la sintaxis de distintos lenguajes y las tasas de precisión alcanzadas por los programadores neófitos que se acercan a los mismos (Stefik y Siebert, 2013).

3.1.3. Navegación por el código

Un tema común, recurrente en la bibliografía, es la dificultad que tienen los estudiantes con deficiencias visuales a la hora de enfrentarse a la navegación por un código y comprender la estructura general del mismo mientras utilizan un lector de pantalla (Bigham et al., 2008; Kane y Bigham, 2014; Ludi et al., 2014). A menudo, esto puede llevar a que los estudiantes inserten líneas de código en posiciones equivocadas. Se pueden tomar medidas para atenuar estas dificultades. Con objeto de conseguir una mejor comprensión de su ubicación en el código, se puede animar al estudiante a desplazar el cursor de texto para escuchar una lectura en voz alta de los caracteres. Además, el estudiante puede disponer de ejemplos de código en braille que le ayuden a desarrollar una comprensión de la estructura general del código.

Las dificultades inherentes a la navegación por un código, así como a la comprensión de su estructura, se tuvieron en cuenta durante el desarrollo de StructJumper, un *plug-in* para el entorno de programación Eclipse que permite a los usuarios con deficiencias visuales navegar por un programa escrito en Java (Baker et al., 2015). StructJumper genera un árbol compuesto de estructuras anidadas, incluidas en el programa, que hace posible que el usuario salte con facilidad de una estructura a otra dentro del código. Los participantes en una evaluación a escala reducida de StructJumper concluyeron que el *plug-in* les ayudaba a acelerar su navegación por el código.

3.1.4. Otras consideraciones

Asimismo, es importante tener en cuenta que el nivel de visión de los estudiantes con discapacidad visual varía notablemente en casos distintos, y también han de ser distintas las tecnologías asistenciales de preferencia (Bigham et al., 2008; Ludi et al., 2014). Además, los estudiantes pueden tener diversos grados de experiencia en el uso de tecnologías de asistencia. Bigham et al. (2008) concluyeron que los estudiantes que progresaban más rápido eran aquellos que ya tenían un nivel de usuario experto de un lector de pantalla. Otro factor que puede afectar al progreso de los estudiantes con deficiencias visuales es su familiaridad con la distribución del teclado. También se estima que las aptitudes mecanográficas son una herramienta importante para aprender a programar en un lenguaje basado en texto (Ludi, 2013; Ludi et al., 2014).

Otro factor que debe tenerse en cuenta es la accesibilidad de las herramientas diseñadas para crear interfaces gráficas de usuario (GUI, por sus siglas en inglés), ya que las herramientas que se utilizan actualmente para generar este tipo de interfaces o bien no son accesibles, o bien son de uso muy complicado para los estudiantes con discapacidad visual. Para abordar este problema, Siegfried (2006) desarrolló un lenguaje de guiones que hace posible que los programadores con discapacidad visual creen formularios de Visual Basic. Más recientemente, Konecki (2014) desarrolló GUIDL, una herramienta que permite a los estudiantes con deficiencias visuales crear interfaces gráficas de usuario para sus proyectos de programación. Un grupo pequeño de programadores neófitos adultos evaluó GUIDL, llegando a la conclusión de que les resultaba posible utilizar dicha herramienta con efectividad para crear interfaces gráficas de usuario de las que podían hacer uso en sus propios programas.

Si bien hay varios estudios que se centran en enseñar a los estudiantes con deficiencias visuales cómo programar en un lenguaje basado en texto (TBL, por sus

siglas en inglés), dichos estudios se dirigen, sobre todo, a estudiantes de enseñanza secundaria y universitaria. La sección siguiente examinará la accesibilidad de aquellos lenguajes basados en bloques que se dirigen a estudiantes de Educación Primaria.

3.2. Cómo hacer accesibles los lenguajes basados en bloques

Al aprender a programar, se dedica un periodo de tiempo significativo a la asimilación de la sintaxis de un lenguaje concreto. Esto pudiera obstaculizar el desarrollo de una comprensión de los conceptos esenciales de la programación. Lenguajes basados en bloques, como Scratch (Maloney[, Resnick, Rusk, Silverman y Eastmond], 2010), permiten a los estudiantes elaborar programas encajando una serie de bloques, lo que hace innecesario que aprendan la compleja sintaxis de un lenguaje basado en texto.

Los lenguajes con base en bloques son intrínsecamente visuales y, por lo tanto, no resultan accesibles para la mayoría de los estudiantes con deficiencias visuales. Es necesaria una alternativa a lenguajes de bloques como Scratch (Koushik y Lewis, 2016; Ludi, 2015). Una de estas alternativas es Noodle, un sistema de programación destinado a la creación de sonidos y música, que tiene elementos que pueden insertarse y organizarse utilizando exclusivamente comandos de teclado (Lewis, 2014). El concepto en el que se apoya Noodle es prometedor; sin embargo, no parece que se hayan realizado pruebas con estudiantes, y el lenguaje utilizado en las señales de audio que orientan al usuario no resulta apropiado para niños de Primaria. Este factor lo convierte en una opción inadecuada para iniciar en la programación a niños con deficiencias visuales.

Ludi (2015) y su equipo llevan tiempo esforzándose en conseguir que el lenguaje de Blockly sea accesible a estudiantes con deficiencias visuales. El lenguaje que están desarrollando hará posible la navegación exclusivamente mediante el teclado, incorporando también señales de audio que permitan comunicar el nivel de anidamiento, o estructura jerárquica, en el que se encuentre el usuario. A partir del trabajo realizado en Noodle, Lewis ha colaborado con Koushik en el desarrollo de otro lenguaje accesible basado en Blockly, al que han dado el nombre de lenguaje de Bloques Seudoespaciales (PB, por sus siglas en inglés) (Koushik y Lewis, 2016). El adjetivo *seudoespacial* hace referencia a la naturaleza distorsionada de la geometría del movimiento. En el lenguaje PB, el usuario selecciona un punto de inserción utilizando el teclado, y puede escoger el elemento del programa que desee a partir de una lista filtrada. Los elementos del programa se filtran por categoría sintáctica. Koushik y Lewis (2016) mantienen que el

PB ofrece más ventajas que otros lenguajes visuales dirigidos a todos los estudiantes en general, ya que se filtran y eliminan bloques de programa que resultan inválidos para un espacio determinado.

Los entornos de programación Lady Beetle y World of Sounds se presentan como una alternativa a los lenguajes basados en bloques, y se desarrollaron con el propósito de iniciar a niños con deficiencias visuales en los conceptos básicos de la programación (Jašková y Kaliaková, 2014). El entorno de programación Lady Beetle permite que el estudiante seleccione comandos de una sola palabra sin necesidad de escribirlos con el teclado. Los comandos controlan los movimientos de un icono que representa una mariquita sobre una cuadrícula. Conforme se mueve la mariquita, el estudiante escucha las coordinadas de la casilla correspondiente. Por su parte, World of Sounds permite al usuario crear programas sencillos que producen secuencias de sonidos.

El desarrollo de estas alternativas a los lenguajes basados en bloques constituye un avance esperanzador en la búsqueda de una opción a dichos lenguajes que sea accesible. Sin embargo, también estas opciones podrían plantear dificultades a los estudiantes cuando se trata de comprender la estructura general del código mientras se utiliza un lector de pantalla. La tabla que se incluye en el Apéndice muestra que aún queda bastante camino por recorrer hasta que la investigación sobre los lenguajes de bloques alcance el mismo nivel que la que se ocupa de los lenguajes basados en texto.

3.3. Objetos físicos

3.3.1. Dispositivos programables

La naturaleza física de dispositivos programables, tales como los robots, los convierte en una herramienta de uso común para la enseñanza de la programación en el nivel básico, habiéndose demostrado que resulta igualmente atractiva para los estudiantes con deficiencias visuales (Ludi, 2013). Al enseñar programación utilizando la robótica, los robots pueden venir ya ensamblados o se puede pedir a los mismos estudiantes que construyan sus robots como parte del proceso de aprendizaje. Esta tarea acarrea sus propias dificultades, sobre todo en el caso de estudiantes con deficiencias visuales.

Dorsey Rayshun, Chung Hyuk y Howard (2014) evaluaron cuatro equipos robóticos destinados a tareas de carácter pedagógico en el curso de una serie de talleres de

verano, y estudiaron la idoneidad de su uso por parte de estudiantes con deficiencias visuales. En los talleres, se emparejó a los estudiantes con deficiencias visuales con compañeros videntes para construir robots utilizando varios equipos.

Se concluyó que el LEGO Mindstorm RCX era el de más fácil manejo para estudiantes con deficiencias visuales, ya que exigía un menor apoyo por parte de sus compañeros videntes.

Se han realizado varios estudios que profundizan en los programas de extensión educativa diseñados para aumentar la participación de estudiantes con deficiencias visuales en tareas informáticas mediante el uso de robots (Dorsey et al., 2014; Ludi, 2013; Ludi et al., 2014; Ludi y Reichlmayr, 2011). Las conclusiones de dichos estudios apuntan a un incremento del nivel de confianza del alumno tras su participación en el taller (así como de su deseo de estudiar Informática en el colegio o de dedicarse a esta materia profesionalmente).

3.3.2. Lenguajes físicos de programación

La mayoría de los sistemas utilizados en la informática física, si bien son físicos en sí mismos, se programan utilizando una interfaz gráfica de usuario (GUI) en un ordenador. En los lenguajes físicos de programación (PPL, por sus siglas en inglés), los comandos se representan mediante objetos físicos que se pueden unir para crear programas. Tern PPL utiliza bloques de madera que pueden juntarse para producir un programa. Se recurre a una cámara web para convertir los elementos físicos en un código digital (Horn y Jacob, 2007a, 2007b). Tern se evaluó inicialmente durante un periodo de una semana, con la participación de nueve niños videntes que lo utilizaron para programar robots. Sin embargo, no todos los niños fueron capaces de comprender los efectos que los programas que habían creado tenían en los robots. Esto se puede deber, en parte, a la demora que se produce entre la creación y la ejecución del código, ya que el código físico se tiene que convertir en código digital mediante una cámara web conectada al ordenador.

La propia naturaleza física de los lenguajes físicos de programación significa que pueden ser una poderosa herramienta de aprendizaje para los niños con deficiencias visuales. Sin embargo, Term no es un lenguaje accesible en sí mismo. Por otro lado tenemos Torino, un lenguaje físico de programación que se ha diseñado para integrar a los estudiantes con deficiencias visuales (Thieme[, Morrison, Villar, Grayson y Lind-

ley], 2017). Torino presenta unas cápsulas que se pueden unir para crear programas que producen sonido y música. Cada cápsula cuenta con unos diales que sirven para controlar los parámetros, y permiten que el estudiante cambie la muestra de sonido o la nota, así como la duración. Potencialmente, el carácter físico de los lenguajes Torino puede permitir que el estudiante obtenga una visión conjunta de la estructura general de un programa.

3.3.3. Modelos en 3D

Es práctica común entre los profesores de la Informática utilizar diagramas, gráficos o animaciones para ilustrar conceptos de programación tales como las estructuras de datos. «La mayoría de las herramientas que se emplean para enseñar estructuras de datos, pensamiento algorítmico y programación básica son muy visuales» (Papazafiropulos[, Fanucci, Leporini, Pelagatti y Roncella, 2016]; 491). Si bien las tecnologías de asistencia permiten que los estudiantes con deficiencias visuales tengan acceso a la información, carecen de la virtud de presentar un concepto complejo de forma sencilla, como sí puede hacerlo una representación visual.

Pueden utilizarse modelos en tres dimensiones para representar conceptos abstractos de forma que los haga accesibles a estudiantes con deficiencias visuales. Como parte de su investigación, Stefik et al. (2011) entrevistaron a profesores de una escuela para niños con discapacidad visual, llegando a la conclusión de que, siempre que ello sea posible, los nuevos conceptos deben introducirse utilizando objetos físicos. En respuesta a esto, desarrollaron «objetos didácticos» para impartir conceptos clave de programación, como las variables. Jašková y Kaliaková (2014) utilizaron una mesa táctil con una cuadrícula de 10x10 para enseñar a los niños a formular algoritmos simples. Se les propuso la tarea de escribir una secuencia de comandos en un editor de textos que guiaba a una abeja para que esta siguiera una ruta prefijada en la cuadrícula táctil. Los estudiantes simulaban la ejecución del programa al mover la abeja con las manos.

Con la aparición de las impresoras en tres dimensiones, se hizo mucho más sencillo producir modelos en 3D. Papazafiropulos et al. (2016) emplearon modelos impresos en 3D para realizar un pequeño estudio de viabilidad relativo a la enseñanza de conceptos, como las estructuras de datos y los algoritmos, a niños con deficiencias visuales. El modelo que utilizaron cuenta con cilindros de alturas diversas, de forma que la altura del cilindro simboliza el valor de un elemento. Los

cilindros se insertan en una bandeja que representa una matriz. De esta forma, se enseña el modo en el que los algoritmos de ordenación y búsqueda se pueden aplicar a las matrices.

La impresión en 3D fue utilizada también por Kane y Bigham (2014) como parte de un taller semanal de programación en el que los niños produjeron un código para generar visualizaciones físicas de datos. Los autores se encontraron con que la capacidad de generar e imprimir sus propios mapas táctiles resultaba muy motivadora para los niños, aunque la velocidad de impresión en 3D se convertía en una limitación, puesto que había que dejar que las tareas de impresión se prolongaran a lo largo de toda la noche. También señalaron que es necesario disponer de herramientas universales que se puedan utilizar para crear con facilidad gráficos táctiles.

Lego proporciona un método rápido y sencillo de creación de modelos básicos en 3D que pueden utilizarse para impartir conceptos de programación a estudiantes con deficiencias visuales. Capovilla[, Krugel y Hubwieser], (2013) llegaron a esta conclusión al emplear modelos de Lego para enseñar algoritmos de ordenación y búsqueda a un grupo pequeño de estudiantes con deficiencias visuales. Tras familiarizar a los estudiantes con los algoritmos gracias a los modelos de Lego, se les pidió que resolvieran tareas de ordenación y búsqueda en una hoja de cálculo. Todos los participantes fueron capaces de resolver las tareas encomendadas.

3.4. Retroalimentación auditiva y táctil

Se pueden utilizar variaciones de tono y timbre en los sonidos (así como una respuesta táctil en forma de vibraciones) para indicar los diferentes estados de un objeto o representación virtual. PLUMB EXTRA (EXploring data sTRuctures using Audible Algorithm Animation, o exploración de estructuras de datos utilizando animaciones acústicas de algoritmos) se desarrolló con el objeto de permitir el acceso de estudiantes universitarios con deficiencias visuales a simulaciones de algoritmos diseñadas para manipular estructuras de datos (Calder[, Cohen, Lanzoni, Landry y Skaff], 2007). El programa se basa en PLUMB, un sistema diseñado para hacer posible que los estudiantes con deficiencias visuales naveguen por gráficos (Calder[, Cohen, Lanzoni y Xu], 2006). El sistema PLUMB EXTRA permite al estudiante explorar el estado de las estructuras de datos en cualquier momento mediante una serie de indicadores sonoros. En el estudio realizado por Calder et al. (2007), se describe el desarrollo del sistema, que, sin embargo, solo se ha evaluado de forma limitada.

En el curso de una serie de talleres, Dorsey et al. (2014) utilizaron distintas vibraciones y notas de piano en un mando de Wii para indicar las distintas posiciones de un robot mientras se desplazaba por un laberinto. Los resultados de este estudio indican que los estudiantes con deficiencias visuales son capaces de realizar un tipo de tareas que se considera muy visual siempre que dispongan de la suficiente retroalimentación auditiva y táctil.

4. Análisis

Este examen ha mostrado que los lenguajes basados en texto dominan la bibliografía, pese a que los lenguajes basados en bloques están más presentes en el sistema de Educación Primaria en materia de Informática, como destaca un reciente informe de la Royal Society (The Royal Society, 2017). Según el Plan Nacional de Estudios (Department for Education, 2014), en Inglaterra todos los niños deberían ser capaces de aprender los conceptos básicos de programación a partir de los 5 años de edad. Sin embargo, el carácter intrínsecamente inaccesible de los lenguajes basados en bloques, junto con su uso generalizado en las clases de Informática en la Educación Primaria, pueden tener como efecto que los niños con deficiencias visuales se vean excluidos de las clases de programación. Se han dado los primeros pasos para conseguir que los lenguajes basados en bloques se hagan accesibles a los estudiantes con deficiencias visuales. Sin embargo, todavía queda mucho camino por recorrer, y es precisa una investigación más amplia en la materia.

Las investigaciones sobre la utilización de lenguajes basados en texto por parte de estudiantes con deficiencias visuales han detectado las dificultades a las que estos se enfrentan para comprender la estructura general del código, ya que solo pueden escuchar una línea de código a la vez, lo que les obliga a depender en gran medida de la memoria a corto plazo. Aunque se ha demostrado que es posible hacer asequibles estos lenguajes a estudiantes con deficiencias visuales, las dificultades mencionadas todavía podrían suponer una barrera. Por otro lado, los lenguajes físicos de programación pueden potencialmente permitir que los estudiantes con deficiencias visuales desarrollen una comprensión de la estructura del código a través del tacto, siempre que los distintos bloques o elementos utilizados por estos lenguajes presenten diferencias físicas. Por lo tanto, se han de investigar los posibles beneficios derivados de la utilización de lenguajes físicos de programación por parte de estudiantes con deficiencias visuales, así como los procesos de aprendizaje relacionados con el uso de tales lenguajes.

La literatura relativa a los lenguajes basados en texto ha identificado varios de los desafíos a los que se enfrentan los estudiantes con deficiencias visuales, así como posibles estrategias para hacerlos frente. Se pueden utilizar estas investigaciones como base teórica de la enseñanza de la programación a estudiantes de Secundaria con deficiencias visuales, pero hay que profundizar más en las mismas. Si se consigue que los estudiantes de Educación Primaria se inicien con éxito en la programación mediante lenguajes físicos de programación o mediante lenguajes accesibles basados en bloques, estos estudiantes podrían acceder a la Educación Secundaria habiendo ya asimilado los conceptos básicos. De esta forma, se podría facilitar la transición del alumno a los lenguajes basados en texto y, por consiguiente, sería posible reducir la magnitud de algunas de las dificultades inherentes a tales lenguajes en la actualidad. Esto pone de manifiesto la necesidad apremiante de realizar investigaciones sobre estrategias que hagan accesible la programación a los estudiantes de Educación Primaria con deficiencias visuales.

5. Conclusión

Gran parte de las investigaciones realizadas en este campo hace hincapié en la definición de las medidas que hay que tomar, así como en el impacto de las mismas en la percepción y motivación del estudiante, prestándose una atención limitada a los aspectos pedagógicos de la enseñanza de la programación a estudiantes con deficiencias visuales. Sin duda, es este un terreno que requiere mayor estudio.

Actualmente, los programas más utilizados en los cursos de introducción a la programación en la Educación Primaria del Reino Unido se basan en bloques (The Royal Society, 2017) y no resultan accesibles para estudiantes con deficiencias visuales. Por consiguiente, es necesaria una investigación más detallada sobre posibles alternativas a los lenguajes basados en bloques que sean asequibles para estos estudiantes. Los lenguajes físicos de programación abren una vía prometedora, habida cuenta de su potencial a la hora de permitir que los estudiantes lleguen a asimilar la estructura general de un código.

6. Resumen

Una serie de estudios ha investigado las formas en las que se puede hacer accesible a estudiantes con deficiencias visuales el aprendizaje de lenguajes basados

en texto (Bigham et al., 2008; Dorsey et al., 2014; Kane y Bigham, 2014; Ludi, 2013; Ludi et al., 2014; Ludi y Reichlmayr, 2011; Smith et al., 2000; Stefik et al., 2011), aunque dichos estudios se centran, sobre todo, en estudiantes de Educación Secundaria y universitaria. También se han examinado los lenguajes basados en bloques, y la manera de hacerlos asequibles para estudiantes con deficiencias visuales (Koushik y Lewis, 2016; Lewis, 2014). Los Pseudospacial Blocks (PB) constituyen un avance prometedor, que se adapta en gran medida a las necesidades de estudiantes de Educación Primaria con deficiencias visuales. Sin embargo, hay que señalar que, como ocurre con los lenguajes basados en texto, los estudiantes pueden verse en dificultades a la hora de comprender la estructura general del código al utilizar los PB.

Se pueden emplear objetos de carácter físico para captar la atención tanto de estudiantes con visión normal como de estudiantes con deficiencias visuales. Un ejemplo de esto es el uso de la robótica (Dorsey et al., 2014; Ludi, 2013; Ludi et al., 2014; Ludi y Reichlmayr, 2011). Este enfoque tiene el inconveniente de que, en la actualidad, todavía se apoya en los lenguajes basados en texto, y el uso de tales lenguajes da lugar a nuevas complicaciones, que ya se han analizado. Por otro lado, los lenguajes físicos de programación pueden ser una poderosa herramienta potencial para la enseñanza de la programación a estudiantes de Educación Primaria con deficiencias visuales, ya que combinan su carácter físico con el hecho de que permiten una fácil comprensión de la estructura general de un programa.

Los modelos en 3D (Kane y Bigham, 2014; Papazafiropulos et al., 2016; Stefik et al., 2011), junto con la retroalimentación auditiva y táctil (Calder et al., 2007; Dorsey et al., 2014), se han revelado como un apoyo eficaz en el proceso de enseñanza, pero no pueden utilizarse de forma aislada para enseñar la programación, por lo que han de combinarse con otras estrategias.

7. Directrices

Basándose en la bibliografía, se han elaborado una serie de directrices para educadores y planificadores curriculares que se ocupan de estudiantes con deficiencias visuales. Sin embargo, debe tenerse en cuenta que estas directrices tienen su origen en la literatura disponible en la actualidad, y pueden modificarse según se avance en la investigación y se aporten más datos.

1. Los lenguajes físicos de programación accesibles pueden constituir una alternativa adecuada a los lenguajes basados en bloques para iniciar en la programación a niños con discapacidad visual.

2. Es posible enseñar conceptos sencillos de programación a niños con discapacidad visual mediante el uso de objetos en 3D; por ejemplo, creando un algoritmo que controle el movimiento de una mariquita sobre una cuadrícula táctil.

3. La elección de un lenguaje es un factor importante cuando se utilizan lenguajes de programación basados en texto en las actividades lectivas. O bien se ha de elegir un lenguaje diseñado especialmente para estudiantes con discapacidad visual, o bien uno de uso general, con una sintaxis sencilla y un uso limitado de caracteres no alfanuméricos, como, por ejemplo, Ruby.

4. Hay que asegurarse de elegir un entorno de programación que sea completamente accesible, y cuya navegación sea sencilla utilizando un lector de pantalla. Si no se dispone de un entorno adecuado, se puede hacer uso de un simple editor de texto, aunque la falta de herramientas de depuración puede suponer un problema.

5. Con frecuencia, conceptos abstractos que normalmente se aprenden utilizando representaciones visuales se pueden enseñar de forma efectiva a estudiantes con deficiencias visuales empleando objetos en 3D. Por ejemplo, las estructuras de datos se pueden explicar mediante cilindros de distintos tamaños que se insertan en una bandeja.

6. Los estudiantes con deficiencias visuales a menudo experimentan grandes dificultades para lograr una comprensión general de la estructura de un código escrito en lenguajes basados en texto. Una posible estrategia de apoyo consiste en proporcionar un ejemplo del código de que se trate en braille (para aquellos estudiantes que dominen este lenguaje).

7. La elección de un tema adecuado en el que basar las actividades de programación puede lograr que estas resulten accesibles e interesantes para los estudiantes con deficiencias visuales. Por ejemplo, aquellas tareas que consistan en programar un dispositivo físico, como un robot, pueden resultar muy atractivas. Sin embargo, es preciso aportar información que no sea visual sobre la posición

en la que esté el robot en cada momento: esto se puede conseguir recurriendo, entre otras cosas, a la retroalimentación visual y táctil.

8. Referencias bibliográficas

Albusays, K., y Ludi, S. (2016). Eliciting programming challenges faced by developers with visual impairments. En: *Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering - chase '16* (pp. 82-85). Austin, tx, ee. uu. https://doi.org/10.1145/2897586.2897616.

Baker, C. M., Milne, L. R., y Ladner, R. E. (2015). StructJumper[: A tool to help blind programmers navigate and understand the structure of code]. En: *Proceedings of the 33rd Annual acm Conference on Human Factors in Computing Systems - chi '15* (pp. 3043–3052). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/2702123.2702589.

Bigham, J. P., Aller, M. B., Brudvik, J. T., Leung, J. O., Yazzolino, L. A., y Ladner, R. E. (2008). Inspiring blind high school students to pursue computer science with instant messaging chatbots [formato pdf]. *acm sigcse Bulletin*, *40*(1), 449. https://doi.org/10.1145/1352322.1352287.

Calder, M., Cohen, R. F., Lanzoni, J., Landry, N., y Skaff, J. (2007). Teaching data structures to students who are blind [formato pdf]. En: *Proceedings of the 12th annual sigcse conference on Innovation and technology in computer science education - iticse '07* (Vol. 39, p. 87). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/1268784.1268811.

Calder, M., Cohen, R. F., Lanzoni, J., y Xu, Y. (2006). plumb: An interface for users who are blind to display, create, and modify graphs. En: *Proceedings of the 8th International acm sigaccess Conference on Computers and Accessibility - assets '06* (p. 263). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/1168987.1169046.

Capovilla, D., Krugel, J., y Hubwieser, P. (2013). Teaching algorithmic thinking using haptic models for visually impaired students [formato pdf]. En: *2013 Learning and Teaching in Computing and Engineering* (pp. 167–171). ieee. https://doi.org/10.1109/LaTiCE.2013.14.

Cheong, C. (2010). Coding without sight: Teaching object-oriented java programming to a blind student [formato pdf]. En: *Eighth Annual Hawaii International Conference on Education* (pp. 1–12). Hawaii International Conference on Education.

Department for Education (2014). *The national curriculum in England - Framework document* [formato pdf]. Department for Education.

Dorsey, R., Chung, H. P., y Howard, A. (2014). Developing the capabilities of blind and visually impaired youth to build and program robots [formato pdf]. En: *28th Annual International Technology and Persons with Disabilities Conference*. San Diego: California State University, Northridge.

Francioni, J. M., y Smith, A. C. (2002). Computer science accessibility for students with visual disabilities [formato pdf]. En: *Proceedings of the 33rd sigcse Technical Symposium on Computer Science Education - sigcse '02* (Vol. 34, p. 91). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/563340.563372.

Franqueiro, K. G., y Siegfried, R. M. (2006). Designing a scripting language to help the blind program visually [formato pdf]. En: *Proceedings of the 8th International acm sigaccess Conference on Computers and Accessibility - assets '06* (p. 241). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/1168987.1169035.

Horn, M. S., y Jacob, R. J. K. (2007a). Designing tangible programming languages for classroom use [formato pdf]. En: *Proceedings of the 1st International Conference on Tangible and Embedded Interaction - tei '07* (p. 159). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/1226969.1227003.

Horn, M. S., y Jacob, R. J. K. (2007b). Tangible programming in the classroom with tern [formato pdf]. En: *chi '07 Extended Abstracts on Human Factors in Computing Systems - chi '07* (p. 1965). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/1240866.1240933.

Howard, A. M., Park, C. H., y Remy, S. (2012). Using haptic and auditory interaction tools to engage students with visual impairments in robot programming activities [formato pdf]. *ieee Transactions on Learning Technologies*, *5*(1), 87–95. https://doi.org/10.1109/TLT.2011.28.

Jašková, Ľ., y Kaliaková, M. (2014). Programming Microworlds for visually impaired pupils [formato pdf]. En: G. Futschek y C. Kynigos (eds.), *Proceedings of the 3rd International Constructionism Conference*. Viena.

Kane, S. K., y Bigham, J. P. (2014). Tracking @stemxcomet[: Teaching programming to blind students via 3D printing, crisis management, and Twitter] [formato pdf]. En: *Proceedings of the 45th acm Technical Symposium on Computer Science Education - sigcse '14* (pp. 247-252). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/2538862.2538975.

Konecki, M. (2014). guidl as an aiding technology in programming education of visually impaired [formato pdf]. *Journal of Computers*, *9*(12), 2816-2821. https://doi.org/10.4304/jcp.9.12.2816-2821.

Koushik, V., y Lewis, C. (2016). An accessible blocks language. En: *Proceedings of the 18th International acm sigaccess Conference on Computers and Accessibility - assets '16* (pp. 317-318). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/2982142.2982150.

Lewis, C. (2014). Work in progress report: Nonvisual visual programming [formato pdf]. En: *Proceedings of the 25th Psychology of Programming Annual Conference (ppig 2014)*.

Ludi, S. (2013). Robotics programming tools for blind students [formato pdf]. En: *28th Annual International Technology and Persons with Disabilities Conference*. San Diego: California State University, Northridge.

Ludi, S. (2015). Position paper: Towards making block-based programming accessible for blind users. En: *2015 ieee Blocks and Beyond Workshop (Blocks and Beyond)* (pp. 67-69). ieee. https://doi.org/10.1109/blocks.2015.7369005.

Ludi, S., Ellis, L., y Jordan, S. (2014). An accessible robotics programming environment for visually impaired users. En: *Proceedings of the 16th international acm sigaccess conference on Computers & accessibility - assets '14* (pp. 237-238). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/2661334.2661385.

Ludi, S., y Reichlmayr, T. (2011). The use of robotics to promote computing to pre-college students with visual impairments. *acm Transactions on Computing Education*, *11*(3), 1–20. https://doi.org/10.1145/2037276.2037284.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., y Eastmond, E. (2010). The Scratch programming language and environment [formato pdf]. *acm Transactions on Computing Education*, *10*(4), 1–15. https://doi.org/10.1145/1868358.1868363.

Papazafiropulos, N., Fanucci, L., Leporini, B., Pelagatti, S., y Roncella, R. (2016). Haptic models of arrays through 3D printing for computer science education. En: *International Conference on Computers Helping People with Special Needs* (pp. 491-498). Springer, Cham. https://doi.org/10.1007/978-3-319-41264-1_67.

Raman, T. V. (1996). Emacspeak—direct speech access [formato pdf]. En: *Proceedings of the second annual acm conference on assistive technologies - assets '96* (pp. 32-36). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/228347.228354.

Remy, S. L. (2013). Extending access to personalized verbal feedback about robots for programming students with visual impairments. En: *Proceedings of the 15th International acm sigaccess Conference on Computers and Accessibility - assets '13* (pp. 1-2). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/2513383.2513384.

Sánchez, J., y Aguayo, F. (2005). Blind learners programming through audio [formato pdf]. En: *chi '05 extended abstracts on Human factors in computing systems - chi '05* (p. 1769). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/1056808.1057018.

Sánchez, J., y Aguayo, F. (2006). apl: Audio Programming Language for blind learners. En: K. Miesenberger, J. Klaus, W. L. Zagler y A. I. Karshmer (eds.), *Computers Helping People with Special Needs. icchp 2006. Lecture Notes in Computer Science* (1.ª ed., pp. 1334–1341). Springer, Berlín, Heidelberg. https://doi.org/10.1007/11788713_192.

Siegfried, R. M. (2006). Visual programming and the blind: The challenge and the opportunity [formato pdf]. En: *sigcse '06 Proceedings of the 37th sigcse Technical Symposium on Computer science education* (Vol. 38, pp. 275-278). Houston, Texas: acm. https://doi.org/10.1145/1124706.1121427.

Siegfried, R. M., Diakoniarakis, D., Franqueiro, K. G., y Jain, A. (2005). Extending a scripting language for visual basic forms [formato pdf]. *acm sigplan Notices*, *40*(11), 37. https://doi.org/10.1145/1107541.1107547.

Smith, A. C., Francioni, J. M., y Matzek, S. D. (2000). A Java programming tool for students with visual disabilities [formato pdf]. En: *Proceedings of the fourth International acm Conference on Assistive Technologies - assets '00* (pp. 142-148). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/354324.354356.

Stefik, A., Alexander, R., Patterson, R., y Brown, J. (2007). wad: A feasibility study using the Wicked Audio Debugger. En: *15th ieee International Conference on Program Comprehension (icpc '07)* (pp. 69-80). ieee. https://doi.org/10.1109/icpc.2007.42.

Stefik, A., Hundhausen, C., y Smith, D. (2011). On the design of an educational infrastructure for the blind and visually impaired in computer science [formato pdf]. En: *Proceedings of the 42nd acm technical symposium on Computer science education - sigcse '11* (p. 571). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/1953163.1953323.

Stefik, A., y Siebert, S. (2013). An empirical investigation into programming language syntax. *acm Transactions on Computing Education*, *13*(4), 1-40. https://doi.org/10.1145/2534973.

Stefik, A., Siebert, S., Stefik, M., y Slattery, K. (2011). An empirical comparison of the accuracy rates of novices using the Quorum, Perl, and Randomo programming languages [formato pdf]. En: *Proceedings of the 3rd acm sigplan workshop on Evaluation and usability of programming languages and tools - plateau '11* (p. 3). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/2089155.2089159.

The Royal Society (2017). *After the reboot: computing education in UK schools* [formato pdf].

Thieme, A., Morrison, C., Villar, N., Grayson, M., y Lindley, S. (2017). Enabling collaboration in learning computer programing inclusive of children with vision impairments. En: *Proceedings of the 2017 Conference on Designing Interactive Systems - dis '17* (pp. 739-752). Nueva York, Nueva York, ee. uu.: acm Press. https://doi.org/10.1145/3064663.3064689.

Alex Hadwen-Bennett. King's College, Londres (Reino Unido).

Sue Sentance. King's College, Londres (Reino Unido).

Cecily Morrison. Microsoft Research, Cambridge (Reino Unido).

Hadwen-Bennett, A., Sentance, S., y Morrison, C. (2019). Cómo conseguir que la programación sea accesible a estudiantes con discapacidades visuales: examen de la bibliografía. *Integración: Revista digital sobre discapacidad visual, 74*, 127-150.

Apéndice
Referencias bibliográficas cruzadas por tema

	Cómo hacer accesibles los lenguajes basados en texto	Cómo hacer accesibles los lenguajes basados en bloques	Objetos físicos	Retroalimentación auditiva y táctil
(Bigham et al., 2008)	x			
(Kane y Bigham, 2014)	x		x	
(Cheong, 2010)	x			
(Smith et al., 2000)	x			
(Francioni y Smith, 2002)	x			
(Ludi y Reichlmayr, 2011)	x		x	
(Dorsey et al., 2014)	x		x	x
(Ludi, 2013)	x		x	
(Ludi et al., 2014)	x		x	
(Sánchez y Aguayo, 2005)	x			
(Sánchez y Aguayo, 2006)	x			
(Andreas Stefik, Siebert, et al., 2011)	x			
(Andreas Stefik, Hundhausen, et al., 2011)	x		x	
(Andreas Stefik y Siebert, 2013)	x			
(Konecki, 2014)	x			
(Baker et al., 2015)	x			
(Siegfried, Diakoniarakis, Franqueiro, y Jain, 2005)	x			

	Cómo hacer accesibles los lenguajes basados en texto	Cómo hacer accesibles los lenguajes basados en bloques	Objetos físicos	Retroalimentación auditiva y táctil
(Franqueiro y Siegfried, 2006)	x			
(Siegfried, 2006)	x			
(A. Stefik et al., 2007)	x			
(Lewis, 2014)		x		
(Ludi, 2015)		x		
(Koushik y Lewis, 2016)		x		
(Thieme et al., 2017)			x	
(Papazafiropulos et al., 2016)			x	
(Capovilla et al., 2013)			x	
(Calder et al., 2007)				x
(Calder et al., 2006)				x
(Howard, Chung Hyuk Park, y Remy, 2012)	x		x	x
(Jašková y Kaliaková, 2014)		x	x	x
(Remy y L., 2013)			x	