Recibido: 7 de septiembre 2018 Aceptado: 2 de noviembre 2018

Blockly Voice: un entorno de programación guiado por voz

Blockly Voice: a voice-guided programming environment

César Iván Delgado Silva, Juan Pablo Sandoval Alcocer & Wendoline Arteaga Sabja

Departamento de Ciencias Exactas e Ingeniería Universidad Católica Boliviana "San Pablo" Regional Cochabamba, Bolivia

ceides9497@gmail.com

Resumen: Hoy en día, resolver problemas utilizando el computador se ha convertido en una actividad importante. La programación es útil no solo en actividades relacionadas a la ingeniería sino también en el desarrollo de habilidades como la resolución de problemas o el pensamiento crítico. Desafortunadamente, las personas con discapacidad en los miembros superiores tienen diferentes dificultades para aprender a programar. Principalmente porque las herramientas para enseñar programación requieren interactuar con el teclado y el ratón.

Este artículo presenta "*Blockly Voice*" un entorno de programación guiado por voz para ayudar a las personas a implementar pequeños programas dictando las instrucciones a la computadora mediante un micrófono. Se ejecutó un estudio exploratorio y cualitativo con diez estudiantes de secundaria, uno con discapacidad en los miembros superiores, en el que tuvieron que resolver pequeños ejercicios de programación. Como resultado, los participantes encontraron la herramienta práctica y beneficiosa para implementar programas.

Palabras clave: Herramientas de *software*, reconocimiento de voz, interacción humano computador, programación.

Abstract: Nowadays, solving issues with a computer has become an important activity, programming is useful not only for engineering careers, but also for developing skills like problem resolutions and critical thinking. Unfortunately, people with hand related disabilities have different challenges while learning programming. Mainly because educational programming tools require interactions with the keyword or mouse.

In this paper, we propose "Blockly Voice" a voice guide programing environment to help people to implement small programs without the need to physically interact with the computer. We perform an explorative and qualitative study involving ten highschool students (one with hand disabilities) solving small programming exercises. As a result, participants found our tool as practical and beneficial to implement programs.

Key words: Software tools, speech recognition, human computer interaction.

1 Introducción

En una época donde el uso de la tecnología ha llegado a ser una necesidad; detectar y resolver problemas usando el computador presenta una oportunidad real para mejorar diferentes actividades del diario vivir. Por eso es importante introducir a los jóvenes en el mundo de la programación desde edades tempranas y así poder aumentar sus destrezas y por consiguiente sus posibilidades laborales. A pesar de que existe una gran cantidad de puestos laborales en esta área, las personas con discapacidad o pérdida de miembros superiores no tienen un fácil acceso al mundo de la programación, ya sea por las herramientas o la desmotivación que implica el uso de *hardware* (teclado y ratón) para poder programar.

Por otro lado, las herramientas de programación actuales, para personas con discapacidad en miembros superiores, no brindan una solución viable cuando se trata de aprender a programar mediante editores de código visual (Scratch, Alice, Lego, etc.) ya que forzosamente tienen que tener una interacción con el teclado y el ratón.

En este artículo presentamos un prototipo de herramienta que a través de instrucciones de voz permite la implementación de programas usando un lenguaje visual de bloques interconectados. Nuestro objetivo es proporcionar a las personas con discapacidad en miembros superiores una alternativa práctica que les permita implementar programas.

2 Programación y Personas con Discapacidad

Las personas con discapacidad en miembros superiores tienen varias dificultades para aprender a programar y/o recurrir a planes de enseñanza de programación, ya que como bien se sabe para poder programar además de ideas se debe realizar interacción física para hacer uso de una computadora, teclado y ratón; por ende, pueden llegar a sentir que no tienen un lugar en el campo de la programación y que son excluidos debido a su discapacidad. Algunas de estas personas son relegadas a pesar de poseer el potencial de una mente que piensa de manera algorítmica y usa lógica para llegar a soluciones mucho mejores que personas sin ninguna discapacidad. Se puede entender que por estas razones varias personas con discapacidad en miembros superiores abandonan la oportunidad de aprender programación.

Existen casos de personas con discapacidad o pérdida de extremidades superiores que aseguran que a pesar de su discapacidad pueden ser incluidos dentro del área de la programación y desarrollo de software ya que no tienen la necesidad de tener que ir a una oficina dentro una empresa, no creen que es imposible programar ni aprender sobre programación. Tenemos el caso más popular de Max Strzelecki, el desarrollador que programa con los pies, que desarrolló un videojuego desde su hogar, su juego está en varias tiendas online y funciona con varias plataformas, el asegura que desde pequeño le gustaron mucho los videojuegos y siempre soñó con

desarrollar uno (Pascual, 2014). Otro caso es el de Adrian Hands un programador que usaba el morse para escribir código, después de terminar inmovilizado casi por completo por una enfermedad, solo teniendo la posibilidad de usar un dedo para escribir código, ayudó en cientos de proyectos y no dejó de lado su pasión por la programación (Johnbo, 2012). Como estos hay más casos de personas que desean aprender a programar a pesar de sus limitaciones.

3 Lenguaje de Programación Visual de Bloques Interconectados

Actualmente, varios cursos de programación utilizan lenguajes visuales para la enseñanza de programación. Un lenguaje de programación visual es un lenguaje que utiliza elementos gráficos y figuras para el desarrollo del programa. Del mismo modo que cualquier lenguaje de programación estos tienen una sintaxis y semántica.

Sintaxis. Los lenguajes de programación utilizan un conjunto de bloques predefinidos para el desarrollo del programa. Normalmente, cada bloque representa un elemento de programación como ser: estructuras de control, variables, operadores, etc. Por ejemplo, la Figura 1: muestra un programa que solicita dos números al usuario e imprime la suma. Para armar este pequeño programa se utilizaron varios tipos de bloques, el lenguaje define un conjunto finito de bloques los cuales pueden ser utilizados para desarrollar diferentes tipos de programas (Pasternak, Fenichel, & Marshall, 2017) (Espinal, 2015, p. 11).

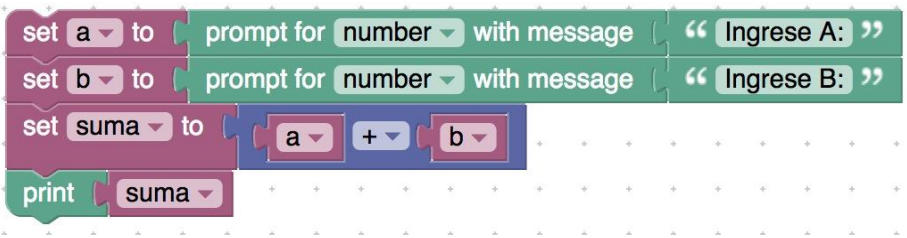

Figura 1: Blockly – Lenguaje de Programación Visual

Semántica. Así mismo los lenguajes de programación visual contienen un conjunto de reglas semánticas para garantizar la correcta formación de expresiones y programas. Por ejemplo, operaciones no legales como "(2>2)+3" no pueden ser formadas a través de bloques, ya que el lenguaje realiza una verificación semántica antes de unir un bloque con el otro, de esta forma detecta parcialmente un buen número de errores semánticos durante el desarrollo del programa.

4 Blockly Voice: Diseño e Implementación del Prototipo

Existen muchos lenguajes de programación visuales, muchos de ellos con características similares. Para este prototipo se utilizó *Blockly*, se eligió este editor porque el código fuente es libre y se lo puede usar de forma gratuita. Lo que permite adaptar el editor para cumplir el objetivo (Pasternak, Fenichel, & Marshall, 2017).

Esta sección describe el prototipo de entorno de desarrollo (IDE) que se diseñó para la programación guiada por comandos de voz. El entorno de desarrollo sigue un flujo de 3 etapas para procesar las instrucciones de voz y generar el código visual de bloques interconectados. Dicho flujo se puede apreciar en la Figura 2:.

Figura 2: Blockly Voice – Flujo de Trabajo

- **Fase 1: Reconocimiento de Voz.** En esta fase se analiza la voz del usuario y se reconocen las oraciones dictadas por el mismo. Dando como salida un conjunto de cadenas de caracteres que representan las instrucciones dadas por el usuario.
- **Fase 2: Análisis Sintáctico & Generación de Código.** Esta fase toma como entrada las instrucciones dictadas por el usuario como cadenas de texto. Se analiza la sintaxis y la semántica de las instrucciones para luego generar un código intermedio.
- **Fase 3: Integración con el IDE.** En esta fase se toma como entrada el código intermedio generado por la fase anterior. El mismo se procesa, para luego generar el programa usando un lenguaje visual de bloques interconectados.

Las siguientes subsecciones detallan cada una de estas fases.

4.1 Reconocimiento de Voz

Reconocer la voz de una persona no es una tarea trivial, debido a que las personas tienen diferentes tonos de voz y formas diferentes de pronunciar las palabras. Para realizar el reconocimiento de voz se utilizó una herramienta llamada *Annyang* (Ater, 2018). La misma graba periódicamente la voz del usuario, procesa el

sonido y devuelve un conjunto de cadenas de texto que representan las posibles oraciones que el usuario pronunció. *Blockly Voice*, toma como entrada este conjunto de cadenas, analiza cada una de las cadenas y revisa si alguna de ellas cumple sintácticamente la gramática.

4.2 Analizador Sintáctico & Generación de Código

Para poder reconocer y procesar las instrucciones de voz se define una gramática propia. La misma ayuda a determinar si una instrucción de voz es sintácticamente válida o no (Krishnamurthi, 2017) (Navarro, 2017, pp. 43 - 73). Si bien, esta nueva gramática es parecida a las gramáticas de lenguajes populares como Java, C, Python, entre otros; se diferencia de las anteriores en dos aspectos: soporta instrucciones en castellano, el lenguaje más usado en países latinoamericanos (excepto Brasil) y además está diseñada para favorecer la programación guiada por voz. A continuación, presentamos parcialmente la gramática diseñada en este proyecto usando la notación Backus-Naur Form (BNF) (Evans, 2011, pp. 19 - 52).

Valores. Los valores en nuestra gramática están definidos como se muestra en la Figura 3:

```
1.  <Value> ::= <Boolean> | <Number> | <Text>
2.
3.  <Boolean>  ::= <True> | <False>
4.  <True> ::= "verdadero"
5.  <False> ::= "falso"
6.
7.  <Number> ::= <NaturalNumber> | <NegativeNumber> | <DecimalNumber>
8.  <NaturalNumber>::= [0-9]+
9.  <DecimalNumber>::= [0-9]*"."[0-9]+
10. <NegativeNumber>::= [-]<Number>
11.
12. <Texto>::= <String>
13. <String>::= <Comillas> <whiteSpace> <Char> <whiteSpace> <Comillas>
14. <Comillas>::= '"'
15. <Char>::= [A-Za-zÑñÁáÉéÍíÓóÚúÜü \_\.\s\xF1\xD1 \t\n\r 0-9]*
```

Figura 3: Blockly Voice – Valores

Como se puede ver en la Figura 3: (línea 1), un valor en nuestra gramática puede ser: un booleano (<*Boolean*>), un número (<*Number*>) o una cadena (<*Text*>). La principal diferencia aquí con otros lenguajes es que los valores lógicos están definidos en castellano "*verdadero*" y "*falso*". Similar a otros lenguajes de programación las cadenas también están delimitadas por comillas, note que los acentos y diéresis son caracteres válidos (línea 15). Los valores están directamente asociados a la forma en

que el usuario realiza el dictado usando el micrófono. Por ejemplo, si el usuario pronuncia la palabra "*verdadero*", el lenguaje directamente reconocerá que la palabra es un valor lógico.

Variables. Las variables están definidas como se muestra en la figura 4.

```
1. <Variable>::= <Identifier>
2.
3. <Identifier>::= <IdChar>+
4. <IdChar>::= [A-Za-zÑñÁáÉéÍíÓóÚúÜü\s\xF1\xD1]+
```

Figura 4: Blockly Voice – Variables

Como se puede ver en la Figura 4: (línea 1), para declarar una variable el usuario solo debe pronunciar un identificador (nombre de la variable). En este caso como se ve en las líneas 3 y 4 que un identificador es un conjunto de caracteres.

Expresiones Aritméticas. Las expresiones aritméticas en el lenguaje se definen como se muestra en la Figura 5:

```
1.  <Expression>::= <Value>  | <Variable> | <BinaryOperation>
2.
3.  <BinaryOperation>::= <Expression> <whiteSpace> <operation> <whiteSpace> <Expression>
4.
5.  <operation>::=    "igual a" | "=" |"es igual a"
6.                  | "diferente a" | "es diferente a"
7.                  | "menor a" | "es menor a"
8.                  | "menor igual a"| "es menor igual a"
9.                  | "mayor a" | "es mayor a"
10.                 | "modulo de" | "modulo"| "módulo de" | "módulo"
11.                 | "más" |"menos"
12.                 | "por"| "entre"
```

Figura 5: Blockly Voice – Expresiones matemáticas

La gramática para expresiones aritméticas es muy similar a otros lenguajes. Como se ve en la línea 1, una expresión puede ser un valor, una variable o una operación binaria. Una expresión binaria está definida por una sub-expresión un espacio en blanco un operador binario y otra sub-expresión (ver línea 3). Esta definición recursiva permite crear expresiones matemáticas largas y compuestas. La principal diferencia con otros lenguajes es que las palabras asociadas a la operación están en castellano y han sido pensadas para que el usuario pueda dictar las mismas.

Por ejemplo, el usuario podrá pronunciar expresiones como "2 más 3 menos 5 por 8".

Estructuras de Control. La Figura 6:, muestra la gramática para la estructura de control condicionales.

```
1.  <StatementIF>:: <If> <whiteSpace> <Expression> <whiteSpace> <Then> <whiteSpace>
2.                          <Statements> <whiteSpace>
3.                  <Else> <whiteSpace> <Statements>
4.
5.                  | <If> <whiteSpace> <Expression> <whiteSpace>
6.                    <Then> <whiteSpace> <Statements>
7.
8.  <If>::= "si" | "sí"
9.  <Else>::= "sino"
10. <Then>::= "entonces"
```

Figura 6: Blockly Voice – Estructuras de control

La gramática es similar al lenguaje de programación C. La diferencia es que la gramática propuesta utiliza palabras reservadas en castellano, como se puede ver en las líneas 8, 9, 10. Del mismo modo, la gramática para las estructuras *repeat* y *while*, la única diferencia es que se utilizan otras palabras claves.

Entradas y Salidas. Definimos unas instrucciones para poder soportar entradas y salidas de datos. La Figura 7:, muestra la gramática para las entradas y las salidas.

```
1.  <Prompt>::= <PromptText>
2.              | <PromptNumber>
3.
4.  <PromptNumber>::= "pedir número" | "ingresar número" | "leer número"
5.                                   | "pedir un número" | "ingresar un número"
6.                                   | "leer un número"
7.  <PromptText>::= "pedir texto" | "ingresar texto" | "leer texto"|"pedir un texto"
8.                                  |"ingresar un texto" | "leer un texto"
9.
10. <Imprimir>::=  <Print> <whiteSpace> <Sentences> <whiteSpace>
11.          | <Print> <whiteSpace> <Sentences>
12.
13. <Print>::= "imprimir"
```

Figura 7: Blockly Voice – Entradas y salidas

Existen dos opciones para solicitar información al usuario, una para solicitar un número y otra para solicitar una cadena. Por ejemplo, el usuario puede pronunciar "pedir un número" o cualquiera de las frases aceptadas (ver línea 7). Así también, el usuario puede imprimir el resultado de una expresión utilizando la palabra "imprimir". Por ejemplo, "imprimir 2 más 3".

Implementación. Se implementó un generador de código que recibe de entrada un conjunto de instrucciones escritas bajo la nueva gramática (sección 4.3.1). La herramienta analiza si las instrucciones son sintácticamente válidas y genera un código intermedio. El código intermedio generado es definido por *Blockly,* el editor de código que usamos para desarrollar el experimento. El editor utiliza este código intermedio como entrada para generar el código visual con bloques. Por ejemplo, la Figura 8: muestra el código intermedio generado para la instrucción "2 más 3 por 5".

```
<block type="math_arithmetic">
  <field name="OP">ADD</field>
  <value name="A">
    <block type="math_number">
      <field name="NUM">2</field>
    </block>
  </value>
  <value name="B">
    <block type="math_arithmetic">
      <field name="OP">MULTIPLY</field>
      <value name="A">
        <block type="math_number">
          <field name="NUM">3</field>
        </block>
      </value>
      <value name="B">
        <block type="math_number">
          <field name="NUM">5</field>
        </block>
      </value>
    </block>
  </value>
</block>
```

Figura 8: Blockly – código intermedio en XML

4.3 Integración con el IDE

Una vez generado el código intermedio se procesa y se muestra usando el lenguaje visual de bloques de *Blockly,* lenguaje que permite al usuario dictar un programa instrucción por instrucción, en este sentido, *Blockly Voice* proporciona también comandos de voz para interactuar con el IDE. Por ejemplo, comandos como "copiar", "pegar" o "deshacer". Del mismo modo, para poder realizar pruebas a los programas *Blockly Voice* permite interactuar con el editor (usando la voz) para poder ejecutar el código y solicitar entradas al usuario, mediante comandos como "ejecutar", "cerrar", "aceptar", entre otros.

5 Estudio Exploratorio con Usuarios

Esta sección describe el diseño y los resultados del estudio exploratorio de usuarios para analizar la viabilidad del prototipo en la resolución de ejercicios de programación.

5.1 Participantes

El estudio exploratorio fue realizado con 10 estudiantes de colegio de 15 a 19 años, 7 varones y 3 mujeres. Uno de los 10 estudiantes tiene una discapacidad en los miembros superiores. Se clasificó a los estudiantes en 3 grupos de personas de acuerdo con su experiencia en programación:

- **Básica**, se considera que un participante tiene experiencia básica en programación si conoce los conceptos de entrada, salida, operaciones aritméticas y variables.
- **Intermedia**, se considera que un participante tiene experiencia intermedia, si ya realizó ejercicios con estructuras de control condicionales e iterativas.
- **Avanzada**, se considera que un participante tiene una experiencia avanzada, si maneja conceptos de funciones, descomposición funcional, abstracción de datos, entre otros.

La Tabla 1 detalla la edad, sexo y experiencia de programación de los participantes.

Tabla 1. Edad, sexo, experiencia de los participantes.

Participante	Edad	Sexo	Experiencia en programación
P1	15	M	Básica
P2	15	M	Básica
P3	15	M	Básica
P4	17	M	Básica
P5	18	M	Básica
P6	18	F	Intermedia
P7	19	M	Avanzada

Participante	Edad	Sexo	Experiencia en programación
P8	18	F	Intermedia
P9	18	F	Intermedia
P10	17	M	Intermedia

5.2 Tareas y Sesión de Trabajo

Con cada uno de los participantes realizó una sesión de trabajo que contó con las siguientes etapas.

- **Preguntas generales**, al inicio se preguntó a los participantes sobre su experiencia previa de programación, En particular, a cada participante se preguntó: ¿Qué actividades realizas con la computadora? ¿Tienes alguna experiencia con asistentes de voz? ¿Tienes alguna experiencia previa con programación?
- **Material de Aprendizaje,** a cada participante se le proporcionó material de aprendizaje para que pueda familiarizarse con la herramienta. Que cuenta con un tutorial sobre cómo usar *Blockly Voice* e ilustra la gramática propuesta a través de algunos ejercicios. Después de leer el material de aprendizaje, a cada participante se le dió un tiempo para que pueda familiarizarse con el editor y con los comandos.
- **Tareas,** a cada participante se le entregó un conjunto de 3 ejercicios básicos de programación:
    - Ejercicio 1: Dado un número ingresado por el usuario imprimir el doble.
    - Ejercicio 2: Dado dos números ingresados por el usuario imprimir el doble de la suma de dichos números.
    - Ejercicio 3: Dados dos números ingresados por el usuario, que representan la base y la altura de un rectángulo, respectivamente, mostrar el perímetro de dicho rectángulo.
- **Retroalimentación,** después de la sesión de ejercicios, se realizaron un número de preguntas abiertas a cada participante. Las respuestas se obtuvieron de forma verbal sin presionar al participante para obtener alguna respuesta. En particular, se preguntó: ¿Qué tan cómodo te sientes con la herramienta?, ¿Los comandos de la herramienta te parecen sencillos?,

¿Volverías a usar la herramienta?, finalmente se preguntó ¿Qué cambiarías de la herramienta?

5.3 Resultados

Preguntas generales. Los participantes entre 15 y 16 años afirmaron que normalmente utilizan la computadora para estar en redes sociales, hacer sus tareas y jugar videojuegos. Por otro lado, los participantes mayores de 17 años, utilizan la computadora para ciertas tareas de programación y diseño. De los 10 participantes solo 3 tuvieron experiencia previa con asistentes de voz (Siri, Google, Cortana, etc). Cinco de los participantes tienen una experiencia básica en programación, cuatro tienen experiencia intermedia, y uno tiene experiencia avanzada.

Tareas, todos los participantes lograron hacer los ejercicios propuestos. En promedio, les tomó entre 2 a 7 minutos resolverlos. Los que necesitaron más tiempo fue debido a que tenían problemas de pronunciación en algunas palabras, que luego de unos intentos lograron corregir hablando más claro, un poco más fuerte o un poco más despacio.

Retroalimentación, a continuación, detallamos la retroalimentación de los participantes en 3 categorías:

- **Gramática,** en las pruebas realizadas los 10 participantes encontraron comprensible e intuitivo el dictado de las instrucciones. Argumentaron que la gramática estaba estructurada de forma similar al lenguaje natural que ellos usan diariamente. Aseguraron que sólo debían pensar cómo era la solución de los problemas planteados y luego dictar a la computadora paso a paso lo que debía hacer para resolverlos, al ver como los bloques se ponían en posición ellos sabían que su dictado era correcto.
- **Lenguaje de bloques,** dos participantes comentaron que el lenguaje de bloques les pareció algo nuevo, pues nunca antes habían utilizado una herramienta que funcionará con un lenguaje de programación visual (el más conocido Scratch). El resto de los participantes mencionaron que el lenguaje de bloques es una manera sencilla de entender la lógica que se va componiendo al dictar las instrucciones con la herramienta. A todos los participantes les pareció agradable la visualización de sus comandos en bloques ordenados y sintácticamente correctos.
- **Pronunciación y reconocimiento de voz,** durante el experimento algunos participantes experimentaron problemas con el reconocimiento de voz. En particular, a un participante le costaba la pronunciación de ciertas palabras usadas en la gramática, pero luego de practicar un poco la pronunciación dejó de tener problemas. El reconocimiento de voz tuvo algunos fallos por problemas de internet en algunas sesiones, pero no fueron de gran

importancia, ya que todos los participantes lograron completar el tutorial y los ejercicios con éxito.

6 Discusión y Trabajo Futuro

Proceso enseñanza-aprendizaje. En nuestro experimento, evaluamos la factibilidad técnica de la herramienta para la elaboración de pequeños programas. Sin embargo, hace falta un estudio más detallado y extenso para poder evaluar la utilidad de una herramienta de estas características en el proceso de enseñanza-aprendizaje de la programación.

Estudio Exploratorio. Nuestro estudio exploratorio se realizó con 10 participantes, aunque este es un número pequeño, tratamos de realizar el experimento con participantes con diferente experiencia en programación. Si bien no podemos generalizar los resultados de nuestro estudio, creemos que el mismo proporciona evidencia relevante acerca de la viabilidad de la herramienta desarrollada a la hora de programar utilizando la voz.

Personas con discapacidad en miembro superiores. El público objetivo de nuestra herramienta, son las personas con discapacidad en miembros superiores. Sin embargo, para nuestro experimento, fue difícil encontrar participantes con discapacidad que tengan interés en programación. Como trabajo futuro, se pretende coordinar con centros de atención a personas con discapacidad y evaluar la utilidad de nuestra herramienta en el proceso enseñanza-aprendizaje de programación.

Otras aplicaciones. Durante el estudio exploratorio los participantes identificaron otras aplicaciones para *Blockly Voice*. Entre ellas resaltan: poder programar operaciones comunes en el celular usando la voz, programar un robot con la voz, mientras este está en funcionamiento (*live programming*) y programar televisores usando la voz.

7 Trabajo Relacionado

Hoy en día, existen herramientas que se propusieron para enseñar a programar a los jóvenes o personas interesadas en la programación (Martinez & Molina, 2017, p. 22) (Carralero, 2016). A continuación, hablaremos de las más conocidas:

- **Scratch.** Es una conocida herramienta drag and drop (jalar y pegar) que se basa en ir poniendo bloques en un orden lógico para que estos realicen tareas que los propios jóvenes van proponiendo. Su interfaz es atractiva para niños lo cual ayuda en llamar su atención para el aprendizaje.
- **Alice.** Es un entorno parecido al de scratch, solo que este funciona en un ámbito 3D, enseña a los jóvenes programación orientada a objetos y

programación de eventos. En Alice, los estudiantes arrastran y sueltan cuadros gráficos con el fin de animar un objeto y crear un programa.

- **Lego Mindstorms.** Quizás sea una de las herramientas más conocidas de la famosa marca de construcciones Lego. A diferencia de los anteriores ejemplos la base de esta es la robótica, a través de la robótica acerca a los niños a la programación. Los kits de Lego Mindstorms, que pueden adquirirse en versiones educativas y de consumo, incluyen sensores y motores. Los kits vienen con lenguajes propios de Lego, pero pueden ser modificados para trabajar con lenguajes de terceros.

Sin embargo, ninguna de estas herramientas brinda una verdadera oportunidad para aprender a programar (y/o poder programar) a las personas con alguna discapacidad en miembros superiores y las que sí ayudan en el entorno de programación son muy limitadas y tienen deficiencias.

Herramientas orientadas a personas con discapacidad. Existen unas cuantas herramientas que hacen el entorno de programación y/o aprendizaje más accesible a personas con distintas discapacidades.

- **Darci USB,** Es un dispositivo que empieza a dictar el texto en morse a la computadora.
- **VoiceCode,** fue desarrollada por un programador que quedó inmovilizado de los miembros superiores por un largo periodo de tiempo, entonces buscó una forma para poder programar sin tener que usar sus manos y desarrolló VoiceCode, funciona como si se estuviera hablando con un robot al que se le da instrucciones (Meyer, 2015).
- **MouseKey,** incorporado con un teclado alfanumérico funciona al igual que Darci USB, pero es necesario su control con una mano (López-Escribano, 2012, p. 9).
- **Dragon Dictate,** un software de pago, es una alternativa que puede integrarse a algunos programas para el dictado de texto intuitivamente, se puede personalizar algunos comandos de voz (Nuance, 2018).

Sin embargo, estas herramientas no utilizan un lenguaje de programación visual, que normalmente es usado para la enseñanza de programación, por lo que no brindan una gran ayuda a las personas con discapacidad en miembros superiores a la hora de aprender a programar (y/o programar) en editores, descartando así cualquier modo viable y sencillo de aprendizaje en programación.

8 Conclusión

Este trabajo presenta *Blockly Voice*, un entorno de programación guiado por voz, como una alternativa a los tradicionales *IDEs* que requieren bastante interacción física con el ambiente de programación. Así también se propuso una gramática en castellano con el objetivo de apoyar el proceso de enseñanza-aprendizaje de programación a estudiantes en latino-américa.

Presentamos un caso de estudio con 10 participantes quienes usaron nuestra herramienta para resolver pequeños ejercicios de programación. Nuestros resultados muestran que todos los participantes fueron capaces de realizar pequeños programas y resolver satisfactoriamente los ejercicios de programación planteados. Cabe resaltar que, el único participante con discapacidad resolvió los ejercicios de manera similar a los demás participantes, sin inconvenientes.

Referencias Bibliográficas

[1] Pascual, J. A. (9 de Octubre de 2014). Max Strzelecki, el desarrollador que programa con los pies. Obtenido de ComputerHoy: https://computerhoy.com/noticias/software/ max-strzelecki-desarrollador-que-programa-pies-19425

[2] Johnbo. (26 de Marzo de 2012). Adrian Hands, el impresionante ejemplo de un programador que usaba Morse para escribir código. Obtenido de Genbeta: https://www.genbeta.com/desarrollo/adrian-hands-el-impresionante-ejemplo-de-un-programador-que-usaba-el-morse-para-escribir-codigo

[3] Espinal, J. (2015). Manual introducción a la programación: principios básicos. Republica Dominicana: SDQ Training Center.

[4] P. E., F. R., & M. A. (2017). Tips for Creating a Block Language with Blockly. IEEE Blocks and Beyond Workshop, 21 - 24.

[5] Ater, T. (10 de Agosto de 2018). Annyang Documents. Obtenido de Annyang: https://github.com/TalAter/ annyang/tree /master/docs

[6] Krishnamurthi, S. (2017). Programming Languages: Application and Interpretation. USA: Brown University.

[7] Navarro, G. (2017). Teoría de la Computación (Lenguajes Formales, Computabilidad y Complejidad) Apuntes y Ejercicios. Chile: Departamento de Ciencias de la Computación Universidad de Chile.

[8] Evans, D. (2011). Introduction to Computing Explorations in Language, Logic, and Machines. Virginia, USA: Creative Commons.

[9] M. L., & M. H. (2017). JUEGO DE ENSEÑANZA DE PROGRAMACIÓN PARA NIÑOS. Bogota: FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS DEPARTAMENTO INGENIERÍA DE SISTEMAS.

[10] Carralero, N. (2016). ENTORNOS PARA ENSEÑAR PROGRAMACIÓN EN SECUNDARIA. NUEVOS ENFOQUES. España: IES Pedro Mercedes. Junta de Comunidades de Castilla-La Mancha.

[11] Meyer, B. (2015). VoiceCode. Obtenido de VoiceCode: https://voicecode.io.

[12] López-Escribano, C. (2012). Scratch y Necesidades Educativas Especiales: Programación para todos. España: RED. Revista de Educación a Distancia.

[13] Nuance. (2018). Dragon Dictate. Obtenido de Dragon Dictate: https://www.nuance.com/es-es/ dragon.html.