# *APL: Un Lenguaje de Programación basado en Audio para Aprendices Ciegos*[1]

**Jaime Sánchez & Fernando Aguayo**
Departamento de Ciencias de la Computación, Universidad de Chile
Blanco Encalada 2120, Santiago, CHILE
{jsanchez, faguayo}@dcc.uchile.cl

**Resumen:** El desarrollo de habilidades de programación es un tema relevante en las ciencias de la computación. Los lenguajes de programación han sido usados progresivamente para satisfacer las necesidades y ensamblar con los modelos mentales de los usuarios finales. En el último tiempo se ha expandido el número de usuarios que pueden programar o aprender a programar. La mayoría de estos lenguajes se centran en usuarios videntes. Este estudio introduce APL, lenguaje de programación a través de audio para aprendices ciegos. APL es un lenguaje de programación basado en interfaces de audio para asistir a aprendices ciegos en el desarrollo de habilidades de resolución de problemas y pensamiento algorítmico. APL es también una forma de ayudar a aprendices ciegos a construir significado escribiendo programas computacionales. APL es un lenguaje de programación diseñado por y para aprendices ciegos. APL fue evaluado con aprendices de programación novatos durante y después de su implementación. Ellos conocieron el lenguaje, lo analizaron y realizaron sugerencias para su mejoramiento. La usabilidad de APL fue evaluada por los aprendices a través de la resolución de problemas de complejidad creciente y la aplicación de cuestionarios de usabilidad. Como resultado, los aprendices ciegos escribieron programas, entendieron el proceso de programación y disfrutaron la experiencia. Nuestros resultados preliminares evidencian que es posible construir un lenguaje de programación basado en audio que ensamble con las necesidades y los modelos mentales de aprendices ciegos para ayudarlos a ingresar al mundo de la programación y desarrollar habilidades de resolución de problemas.
**Palabras clave:** Entornos virtuales acústicos, aprendices ciegos, sonido especial, navegación basada en audio, usabilidad

**Abstract:** expanded the number of users who can program or learn how to program. They are focused on sighted users. This study introduces APL, an Audio Programming Language for blind learners. APL is a programming language with audio-based interfaces to assist blind learners to develop problem solving and algorithmic thinking skills. APL is also a way to help blind learners to construct meaning by making programs. APL is a programming language *by* and *for* blind learners. We tested APL with novice blind programmers during and after development. They tried, analyzed and make improvements to APL. They usability tested APL by solving problems with increasingly complexity and answering usability questionnaires. Blind users understood APL, made programs, mapped the programming process and enjoyed the experience showing that a programming language based on audio can be constructed to fit the needs and mental models of blind learners to help them to enter to the programming world.
**Key words:** Virtual acoustic environments, blind children, spatialized sound, audio-based navigation, usability

## 1. Introducción

En las últimas tres décadas se han realizado diversos intentos para acercar la programación a usarios finales: Basic, Logo, Smalltalk, Pascal, Boxer, Playground, KidSim, AgenSheets, LiveWord, Shoptalk [Frenton et al. 89, Lieberman 01, Nardi 93, Repenning 93, Soloway et al. 89, Travers 94, Wilson et al. 03]. Todos estos lenguajes de programación han contribuido a expandir el número de personas

[1] *Artículo seleccionado de RIBIE 2004, extendido y revisado para su publicación en* IE comunicaciones

 

que pueden programar y aprender a programar. Muchas de ellas han aplicado conceptos de interfaces usuarias a la programación. Ello ha generado usuarios finales con mejores habilidades para programar.
Asimismo, la literatura presenta estudios que describen la programación por demostración, programación mediante ejemplos, programación visual, programación gráfica, y programación física [Cypher et al. 93, Cypher et al. 95, Ishii et al. 97, Lieberman 01, McDaniel et al. 99, Montemayor 01, Rosson et al. 01, Travers 94]. La mayoría de estos intentos de programación (si no todos) se centran en programadores visuales.

Un aprendiz con discapacidad visual que quiere aprender a programar debe utilizar sistemas text-to-speech que "leen" comandos de programación, variables, etc. De hecho podemos asumir que existe una presunción implícita en el sentido que usuarios ciegos piensan de la misma forma que usuarios videntes.

Un número importante de estudios recientes muestran que cuando se utilizan aplicaciones basadas en audio los aprendices ciegos pueden desarrollar y ensayar su cognición [McCrindle et al. 00, Mereu et al. 96, Sánchez 99, 01, 03, Sjostrom 01]. La mayoría de estos estudios se centran en el desarrollo de interfaces 3D para modelar el espacio circundante completo como prueba de concepto en relación con interfaces basadas en audio.

Unos pocos estudios analizan los efectos e impacto del sonido espacializado sobre el desarrollo de la cognición de personas ciegas [Baldis 01, Sánchez 99, 01, 03, Sánchez et al. 04]. Ellos evalúan la usabilidad de dichas aplicaciones a través de la resolución de tareas cognitivas para desarrollar relaciones espacio-temporales, memoria de corto plazo, memoria abstracta, abstracción espacial, percepción háptica, y razonamiento matemático.

Esta investigación introduce APL, un lenguaje de programación basado en audio para estimular la resolución de problemas y habilidades de pensamiento en aprendices ciegos novatos. APL fue construido por y para aprendices ciegos. Ellos evaluaron la usabilidad de diferentes versiones del software durante su implementación y contribuyeron a enriquecer y hacer que APL ensamble con sus necesidades y forma de pensamiento. Los aprendices ciegos también evaluaron la usabilidad de versiones finales de APL a través de una inspección cognitiva y la resolución de tareas cognitivas específicas.

## 2. Un Lenguaje de Programación para Aprendices Ciegos

¿Cuáles son las necesidades específicas de los aprendices ciegos para programar?, ¿por qué es complejo para ellos entender los lenguajes de programación convencionales?, ¿podemos desarrollar un lenguaje de programación para que ensamble mejor con sus necesidades?. Estas son algunas de las preguntas que guiaron nuestro estudio.

**Figura 1.** Una aprendiz ciega interactuando con APL a través de sonido.

Un lenguaje de programación comunica el programador con el computador. Para realizar esto la estructura y la lógica están diseñadas de tal forma que pueden ser interpretadas por la máquina. Los lenguajes actuales están basados en la idea de que un programador escribe líneas de comando interpretadas por el computador. Estos comandos deben ser correctamente escritos y bien definidos, de otra manera la máquina no puede entender las instrucciones y, con ello, las tareas específicas no son resueltas. Esto implica memorizar una alta cantidad de líneas de comando para escribirlas correctamente y evitar errores de parseo. La mayoría de estos lenguajes de programación están fuertemente basados en interfaces visuales.

Podemos encontrar dos dificultades principales cuando aprendices ciegos utilizan estos lenguajes. Primero, si utilizan lenguajes puros se encontraran con el problema de verificar la consistencia del programa y la correcta lectura de las líneas de comando. Segundo, si les proveemos herramientas convencionales para apoyar la construcción de programas, ellos se encontraran con la dificultad de interactuar con interfaces usuarias gráficas.

Es precisamente en respuesta a estas dos dificultades que APL intenta cerrar la brecha que existe entre la programación realizada por aprendices videntes y la programación por aprendices ciegos. APL es el primer intento por desarrollar un lenguaje de programación orientado a aprendices ciegos. La idea es que jóvenes ciegos puedan interactuar y comunicarse con el computador como es realizado por los programadores videntes a través de la creación y manipulación de sus propios programas y poder intercambiarlos con otros usuarios, ciegos o videntes. La única diferencia es la forma cómo ellos entienden la programación y el canal sensorial utilizado para interactuar con la máquina.

APL ha sido concebido para hacer más fácil la escritura de líneas de comando, asegurando la correcta escritura y evitando problemas de errores de parseo (parse error) y así facilitar la interacción máquina-programador a través de herramientas adicionales que nos son comúnmente encontradas en los lenguajes de programación.

APL está pensado para usuarios específicos de forma que la interfaz pueda adaptarse a sus necesidades particulares. Como ejemplo, los lenguajes de programación comunes utilizan un mínimo de unidades de almacenamiento o variables apropiadas para las necesidades del usuario. De esta forma, es difícil de imaginar un lenguaje sin strings o variables enteras. Todas emergen de las necesidades del usuario. En contraste, APL implementa un nuevo y poco convencional tipo de variable que es capaz de almacenar sonidos para posteriormente ser manipuladas por los aprendices ciegos. El sonido no es una función de APL, sino que es tratado como una unidad fundamental.

## 3. APL: Lenguaje de Programación a través de Audio

Nuestro objetivo fue desarrollar una aplicación que facilite la manipulación de un lenguaje de programación para aprendices ciegos. Para ello, proporcionamos elementos facilitadores que orienten a los programadores ciegos novatos, mediante la disminución de la complejidad de la sintaxis de un lenguaje convencional y la incorporación de funciones para analizar los programas finales.


**Figura 2.** Capas de APL

### 3.1 Modelamiento

Desarrollamos un modelo de implementación para APL. Dos condiciones básicas fueron establecidas: el software debía ser fácil de implementar y flexible para permitir cambios sin modificar todo el software. APL tiene dos capas principales: Interfaz de audio y lógica de programación (ver Figura 2).

La interfaz de audio considera dos estados: Listas circulares y pregunta o query. Las listas circulares contienen una lista de comando que puede ser escogida de acuerdo al estado del programa, tales como ciclo, condición, entrada, salida, y variable. La pregunta es utilizada para definir los nombres y valores de las variables e ingreso/salida de audio. El texto es opcional.

La lógica de la programación consiste en cuatro estados: 1. Ejecutar el programa, 2. Finalizar ciclo o condición, 3. Borrar el último comando, y 4. Guardar el comando y verificar el próximo paso. Todos ellos

facilitan la correcta semántica y permiten que la lógica sea entendible por el computador. La semántica es visible para el computador pero invisible para el usuario final. Guardar comandos y verificar el próximo paso permite controlar acciones a través de un auto-cuestionamiento acerca de si es una lista o una pregunta, gatillando uno de los estados de la interfaz de audio.

### 3.2 Desarrollo

APL fue desarrollado utilizando Java 1.4.1 y el sintetizador de voz de Java FreeTTS. Estas son excelentes herramientas para la administración de medios y cercanas a la lógica de la máquina. APL tiene los siguientes módulos: Base de datos, integridad, kernel e interfaz persona-computador.

La base de datos almacena los comandos y su integridad. La integridad lee la base de datos desplegando una cierta estructura y manipulando comandos. APL tiene los siguientes comandos: Ciclo, condición, ingreso, salida y variables. El ciclo y la condición tienen el mismo significado que en lenguajes convencionales. La entrada y salida están basadas en audio y texto a voz (text-to-speech) definido por el programador ciego y guardado como una variable para su posterior manipulación. Las variables son sonido, texto y número. El sonido es la principal variable y la más utilizada. El kernel administra la interacción entre el usuario y las reglas de programación. La interfaz persona-computador controla el audio de ingreso y salida y traduce el texto de ingreso y salida a audio. La interfaz está enteramente basada en audio por lo que los programadores ciegos no tienen directa interacción con la pantalla. APL tiene dos modos: programador y ejecución.

APL es un lenguaje de programación para aprendices ciegos. Favorece el ingreso de información y agrega feedback adaptado a los usuarios. Para realizar esto los programadores ciegos novatos no deben escribir los comandos deseados. Por el contrario, ellos pueden seleccionarlos de una lista clasificada por categorías diseñadas cuidadosamente para ayudar a buscar los comandos requeridos. Esto asegura una correcta semántica de los comandos y ayuda a tener un parseo adecuado para la máquina.

Los programadores ciegos pueden escoger comandos dentro de las siguientes categorías: ciclo, condición, entrada, salida, ciclo y variables. Ellas constituyen la primera categoría de comandos y pueden ser almacenadas en una lista anidada completamente navegada a través de flechas hacia arriba y abajo. APL lee cada comando desde una lista de acuerdo a como lo solicite el programador. Esto es realizado a través de un texto a voz integrado al software.

Para moverse a través de la lista de comandos el programador tiene que usar flechas hacia arriba y abajo desplazándose fácil y completamente a través de la lista circular. Para escoger un comando el programador tiene que seleccionar la flecha correcta y una nueva lista es desplegada sustentando el comando precedente. Así, una lista de comando nueva y válida es completada y almacenada en memoria. Los programadores ciegos repiten este proceso tantas veces como sea necesario seleccionando entre un número fijo de comandos para culminar con el producto deseado. Para ejecutar la aplicación programada el usuario tiene que presionar F8.

La definición de comandos es:

**Ingreso:** El programador ciego usa este comando para crear un requerimiento de ingreso cuando ejecuta el programa, seleccionando la variable deseada, teclado o sonido. Este ingreso debe ser guardado como una variable definida por el programador.

**Salida:** Este es medio disponible para que el programador cree una salida del producto. Aquí utilizamos sólo interfaces de sonido. El sonido tiene dos modos: 1. El programador define una variable y la guarda con el sonido correspondiente, y 2. El programador define una variable y la guarda con una palabra correspondiente o frase ingresada a través del teclado y leída como texto a voz cuando se ejecuta el programa.

**Ciclo:** Es un comando que permite ejecutar el programa varias veces en un sector definido. Puede salir solamente del sector definido cuando se logra la condición de ciclo definido. El ciclo es completado mediante la utilización de la opción finalizar ciclo.

**Condición:** Es un comando que compara entre variables ya definidas por el programador. Si las variables son correctas los comandos entre marcas son ejecutadas. Luego, una nueva opción es creada en la lista de comandos: fin de la condición.
**Variable:** Consiste en la finalización de variables. Ellas pueden ser texto o sonido. El texto es leído a través de texto a voz.

Durante el modo de ejecución los comandos programados son ejecutados. Ellos pueden interactuar con el usuario y la máquina de acuerdo a los requerimientos definidos por el programador. Este módulo no tiene una alta interacción con el usuario debido a que la meta principal es ejecutar el programa.

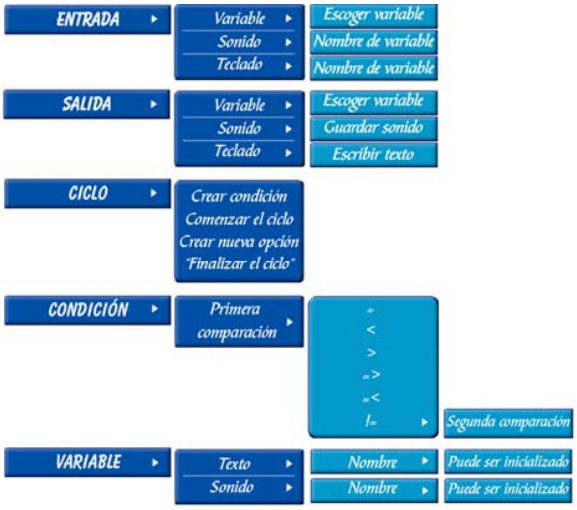


**Figura 3.** Árbol de la lista de comandos

## 4. Resultados de la Evaluación Preliminar de Usabilidad

El primer prototipo de APL fue evaluado con usuarios expertos y usuarios finales.

**Evaluación con usuarios expertos.** APL fue evaluado con cuatro usuarios expertos. La meta principal fue evaluar APL con usuarios que tengan experiencia y conocimiento en programación para validar la funcionalidad e identificar posibles problemas y errores en orden a obtener feedback, sugerencias y algunos mejoramientos.

Los expertos fueron informados acerca de la correspondencia de los comandos de APL con comandos genéricos de lenguajes convencionales. Ellos también programaron el software utilizando los comandos disponibles. Los expertos hicieron comentarios y contestaron preguntas prediseñadas. Como resultado, los expertos detectaron problemas de funcionamiento, tales como: a. APL no soporta ciclos múltiples, b. APL no permite hacer condiciones de cascada, c. Las condiciones de APL no permiten salir de los ciclos, d. Algunos problemas ocurrieron cuando se manipulan variables de teclado (tipos de variables como int, var, y string no podían ser definidas), e. El sonido y las voces pueden ser confusas en algunos aspectos, f. El ejecutor se cayó cuando el ciclo no fue definido apropiadamente.

A partir de estos resultados, nosotros rediseñamos y mejoramos el prototipo tanto en la interfaz del programador como en la ejecución del programa considerando los comentarios y las sugerencias.

**Evaluación de la usabilidad por usuarios finales.** APL fue evaluado a través de un estudio cualitativo de estudio de caso observando y registrando la interacción de los usuarios con el software. La usabilidad de APL fue evaluada con tres aprendices ciegos que no sabían programar cuyas edades eran de 17 a 20 años. La evaluación fue realizada durante cuatro sesiones interactivas de 2.5 horas cada una. Primero, se introdujeron los conceptos básicos de programación y luego ellos resolvieron tareas de ejercicio mental relacionadas. Segundo, los aprendices interactuaron con APL a través de la utilización de comandos básicos para crear programas simples tales como “Hola Mundo” y “Jalisco” (un juego de adivinanzas). Tercero, ellos escribieron sus propios programas tales como “piense una idea y escriba un programa” (ver Figura 1). Finalmente, ellos escribieron un pequeño programa para que ciegos y videntes puedan aprender Braille.

La meta principal fue familiarizar a los usuarios con conceptos y elementos que pueden ser usados para resolver problemas con APL. La idea fue que los usuarios interactúen con APL, identifiquen los algoritmos clásicos y estructura de datos en ciencias de la computación, y desarrollen el pensamiento algorítmico para resolver un problema. Los usuarios

entendieron el funcionamiento y la meta principal de APL. Ellos también aprendieron elementos básicos de programación.

Los usuarios ciegos fueron observados por un grupo de tres investigadores y las conductas principales fueron registradas mediante el uso de un instrumento que utiliza una escala Likert. Luego de finalizar la evaluación los aprendices contestaron un test de usabilidad.

En un comienzo ellos entendieron algunos conceptos de programación, pero fue complejo para los aprendices entender el concepto y las funciones de diferentes comandos, tales como ciclo, condición y variable. Este entendimiento tomó más tiempo del esperado. Durante la tercera sesión los aprendices entendieron estos conceptos y realizaron comentarios concernientes a APL basados en su experiencia de interacción y expectativas acerca de la funcionalidad.

Nosotros observamos que metodologías tales como explicaciones teóricas y ejercicios paso a paso no les ayudaron a entender completamente los conceptos básicos de programación. Esto puede ser explicado debido al hecho que aprendices ciegos tienden a basarse fuertemente en la experiencia concreta antes de construir el pensamiento abstracto. Ellos no construyeron un modelo mental de APL y programación hasta que tuvieron su propia idea y conocieron completamente el significado. Luego, transcurriendo desde aquello que sabían a lo desconocido, los aprendices desarrollaron el entendimiento de la programación.

Los aprendices ciegos fueron capaces de interactuar con APL a través de la utilización de comandos como ingreso, salida, variables, ciclos y condición. Ellos podían escribir algunos programas siguiendo instrucciones paso a paso así como escribir programas que emergieron de sus propias ideas.

Durante la última sesión los jóvenes evidenciaron entender los conceptos de programación y aplicarlos a sus programas. Al realizar esto los usuarios ciegos construyeron habilidades de pensamiento lógico, los verbalizaron con sus compañeros y observadores, y escribieron programas para probar su factibilidad. Ellos estuvieron muy motivados y satisfechos como resultado de la interacción con APL. Asumimos que estos son resultados preliminares pero muy prometedores para nuestra investigación.

En la medida que los usuarios entendieron los conceptos y utilizaron y aplicaron comandos a su programación, fueron capaces de diseñar instrucciones lógicas, entender, reproducir y verbalizarlas. Ellos también construyeron algoritmos que podían ser reproducidos mentalmente. Los usuarios estuvieron muy motivados y mostraron un alto interés y entusiasmo cuando programaban.

Los aprendices ciegos también realizaron diversos comentarios y sugerencias para mejorar APL, tales como: APL debe tener un feedback a través del uso de la barra espaciadora del teclado para recordar la última palabra escrita, iniciar APL por sí mismo y hacerlo más rápido cuando se utilizan las flechas.

Nuestra experiencia con aprendices ciegos programando APL muestra que herramientas convencionales de programación visual no son compatibles con estos aprendices. Parece ser que el problema no es sólo adaptar la programación visual a aprendices ciegos [Siegfried 02]. Esto puede ser más complejo con programadores ciegos novatos. El background teórico de estas herramientas es complejo para ser fácilmente entendido por estos usuarios. Un interesante resultado de nuestra evaluación de usabilidad evidencia que la interactividad puede tener implicaciones y un significado un tanto diferente en usuarios ciegos. Ello necesita ser estudiado más completamente en futuros estudios.

## 5. Conclusión

Este trabajo ha introducido APL, un lenguaje de programación basado en audio para aprendices ciegos. Diseñamos APL para ayudar a aprendices ciegos a entrar al mundo de la programación considerando sus necesidades y modelos mentales. Los aprendices ciegos fueron capaces de interactuar y programar APL utilizando comandos basados en audio. Ellos pudieron entender los conceptos de programación y aplicarlos a sus programas. Los aprendices siguieron instrucciones y programaron sus propias ideas.
Como resultado de nuestro estudio podemos señalar que un uso adecuado del audio como medio sensorial

interactivo puede ayudar a aprendices a aprender a programar. Esto puede ser aplicado para resolver problemas y desarrollar habilidades de pensamiento. Nosotros observamos que a través de las sesiones de evaluación los aprendices ciegos desarrollaron pensamiento algorítmico y habilidades de pensamiento lógico, las verbalizaron y escribieron programas para probar su factibilidad.

La experiencia de interacción con APL fue motivadora e interesante para los usuarios ciegos. Cuando la evaluación concluyó expresaron satisfacción como una consecuencia de su experiencia de programación.

APL no es una propuesta alternativa a la programación visual para aprendices ciegos. Nosotros creemos que para tareas de alto nivel debiera existir interfaces que permitan un acceso adecuado a la programación visual tal como en [Smith et al. 00]. Nuestra propuesta apunta a una dirección diferente. Nosotros pensamos que APL puede ayudar y motivar a nuevos programadores a escribir programas para resolver problemas. Si esta meta puede ser lograda, toda la justificación de APL debiera sustentarse en la estimulación del desarrollo de habilidades tempranas de programación en aprendices novatos, para motivarlos a entrar posteriormente al mundo de la programación visual con interfaces accesibles.

Estamos aprendiendo inicialmente la forma como aprendices ciegos modelan el proceso de programación mediante el uso de APL. Nosotros creemos que este proceso es algo diferente al realizado por usuarios videntes. El concepto de interacción a través de la programación tiene un significado diferente para ellos. Esto debiera ser estudiado en investigaciones futuras. ¿Qué significa la interacción para estos usuarios? ¿Qué significa programar para ellos?, ¿Qué herramientas ensamblan mejor con sus necesidades? Este estudio es un primer paso en nuestro intento por proveer un lenguaje de programación basado en audio que sea robusto y flexible, que satisfaga las necesidades del aprendiz ciego novato y que lo ayude a disfrutar del aprendizaje de la programación y así desarrollar su cognición.

## AGRADECIMIENTOS

Este reporte fue financiado por el Fondo Nacional de Ciencia y Tecnología FONDECYT, Proyecto 1030158.

## REFERENCIAS

[Baldis 01] J. Baldis, Effects of spatial audio on memory, comprehension, and preference during desktop conferences. Proceeding of the ACM CHI ´01, Vol 3, 1, (2001), pp. 166-173.

[Cypher et al. 03] A. Cypher, D. Halbert, D. Kurlander, H. Lieberman, D. Maulsby, B. Mayers, A. Turransky (eds.), Watch What I Do: Programming by Demonstration. Cambridge, MA: The MIT Press, 1993

[Cypher et al. 95] A. Cypher, D. Smith, KidSim: End-user programming of simulations. In Proceedings of ACM CHI´95, Denver, 1995.

[Frenton et al. 89] J. Frenton, K. Beck, Playground: An object-oriented simulation system with agent rules for children of all ages. In Proceedings of ACM OOPSLA ´89. NewYork, 1989, pp. 123-137.

[Ishii et al. 97] H. Ishii, B. Ullmer, Tangible Bits: Toward Seamless Interfaces between People, Bits and Atoms. In Proceedings of ACM CHI ´97. ACM, Atlanta,GA, 1997, pp. 234-241.

[Lieberman 01] H. Lieberman, Your Wish Is My Command: Programming by Example. Morgan Kaufmann, San Francisco, 2001.

[McCrindle et al. 00] R. McCrindle, D. Symons, Audio space invaders. Proceedings of the Third International Conference on Disability, Virtual Reality and Associated Technologies, (2000), pp. 59-65.

[McDaniel et al. 99] R. McDaniel, B. Myers, Getting more out of Programming-By-Demonstration. In Proceedings of ACM CHI ´99. ACM, Pittsburgh PA, 1999, pp. 442-449.

[Mereu et al. 96] S. Mereu, R. Kazman, Audio enhanced 3D interfaces for visually impaired users. Proceedings of ACM CHI ‘96, ACM Press. (1996).

[Montemayor 01] J. Montemayor, Physical programming: software you can touch. In Proceedings of ACM SIGCHI, ACM Press, pp. 81-82, March 2001.

[Nardi 93] B. Nardi, Small Matter of Programming Perspectives on End User Computing. MIT Press, Cambridge MA, 1993.

[Repenning 93] A. Repenning, AgentSheets: A tool for building domain-oriented dynamic, visual environments. Ph.D. Dissertation, Dept. of Computer Science, University of Colorado at Boulder, 1993.

[Rosson 01] M. Rosson, C. Seals, Teachers as simulation programmers: minimalist learning and reuse. In Proceedings of CHI ´01. ACM, Seattle WA, 2001, pp. 237-244.

[Sánchez 99] J. Sánchez, Interactive 3D sound hyperstories for blind children. Proceedings of ACM CHI ´99, (1999). Pittsburg PA, pp. 318-325.

[Sánchez 01] J. Sánchez, Interactive virtual acoustic environments for blind children. Proceedings of ACM CHI ´2001. Seattle Washington, April, (2001), pp. 23-2.

[Sánchez 03] J. Sánchez, AudioBattleShip: Blind Learners Collaboration through Sound. Proceedings of ACM CHI ´03, (2003). Fort Lauderdale Florida, pp. 798-799.

[Sánchez et al. 04] J. Sánchez, H. Flores, Memory enhancement through audio. Proceedings of The Sixth International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2004, Atantla, Giorgia, USA, October 18-20, pp. 24-31.

[Siegfried 02] R. Siegfried, A scripting language to help the blind to program visually. ACM SIGPLAN Notices, Vol 37 (2), pp.53-56, February 2002.

[Sjostrom 01] C. Sjostrom, Using haptics in computer interfaces for blind people. Proceeding of the ACM CHI ´01, Vol 3, 1, (2001), pp. 245-246.

[Smith et al. 00] A. Smith, J. Francioni, S. Matzek, Java Programming tool for students with visual disabilities. Proceedings of ACM ASSETS ´00, pp. 142-148, 2000.

[Soloway et al. 89] E. Soloway, J. Spohrer, Studying de Novice Programmer. Lawrence Erlbaum, Hillsdale, NJ, 1989.

[Travers 94] M. Travers, Recursive interfaces for reactive objects. In Proceedings of ACM CHI ´94. ACM, Boston, 1994, pp. 379-385.

[Wilson et al. 03] A. Wilson, M. Brunett, L Beckwith, O. Granatir, L. Casburn, C. Cook, M. Durham, G. Rothermel, Harnessing curiosity to increase correctness in end-user programming. In Proceedings of ACM CHI ´03, Fort Lauderdale Florida, 2003, pp. 305-312.