I ENCUENTRO INTERNACIONAL DE EDUCACIÓN

Espacios de investigación y divulgación

29, 30 y 31 de octubre de 2014

NEES – Facultad de Ciencias Humanas – UNCPBA

Tandil – Argentina

III-5. Enfoques didácticos y experiencias innovadoras con dispositivos tecnológicos

Una propuesta para la enseñanza de programación en la Escuela Secundaria

Gabriela Cenich

Departamento de Formación Docente - Facultad de Ciencias Exactas

Universidad Nacional del Centro de la Provincia de Buenos Aires

ECienTec (Educación en Ciencias con Tecnologías)

gabcen@exa.unicen.edu.ar

1. Introducción

La enseñanza de la programación en la escuela tuvo su origen hace más de cuarenta años con la incorporación del lenguaje Logo, creado por equipo del MIT (Laboratorio de Inteligencia Artificial del Instituto Tecnológico de Massachussets) liderado por Seymour Papert (López García, 2013). Si bien en la década del ochenta comenzó a difundirse en las escuelas la enseñanza de Logo a partir de los noventa este espacio fue ocupado paulatinamente por la enseñanza de TIC (Tecnologías de la Información y la Comunicación). El interés en las clases de Informática pasó de la utilización de las computadoras como herramientas para pensar y resolver problemas (Jonassen, 1997) al manejo de aplicaciones vinculadas principalmente a tareas de oficina. En la visión de Papert los niños deberían ser capaces de diseñar, crear y expresarse con las nuevas tecnologías (Resnick, 2012). En este sentido Levis (2006) sostiene que "una verdadera alfabetización digital […] debe ofrecer los elementos básicos para el desarrollo de capacidades que permitan la comprensión y dominio del lenguaje en el que están codificados los programas".

En la actualidad han surgido diferentes emprendimientos como Code.org (perteneciente a EEUU), Program.ar y Dale Aceptar (iniciativas nacionales) para promover el desarrollo del pensamiento computacional. Wing (2010) lo define como "los procesos de pensamiento involucrados en la formulación de problemas y sus soluciones, de modo que tales soluciones se representen en una forma que pueda ejecutar eficazmente un agente de procesamiento de información". Recientemente la Sociedad Internacional para la Tecnología en Educación (ISTE) y la Asociación de Docentes en Ciencias de la Computación (CSTA) elaboraron una definición operativa "el pensamiento computacional es un proceso de solución de problemas que incluye (pero no se limita a) las siguientes características: formular problemas de manera que permitan usar computadores y otras herramientas para solucionarlos; organizar datos de manera lógica y analizarlos; representar datos mediante abstracciones, como modelos y simulaciones; automatizar soluciones mediante pensamiento algorítmico (una serie de pasos ordenados); identificar, analizar e implementar posibles soluciones con el objeto de encontrar la combinación de pasos y

recursos más eficiente y efectiva; generalizar y transferir ese proceso de solución de problemas a una gran diversidad de éstos" (ISTE, NSF & CSTA, 2012). En el proceso de aprendizaje para programar o codificar programas las personas no sólo aprenden conceptos matemáticos e informáticos sino que también aprenden otras habilidades como estrategias para resolver problemas, diseño de proyectos y comunicación de ideas. Las personas "no sólo aprenden a codificar, codifican para aprender" (Resnick, 2013).

Si bien una de las razones para aprender programación en la actualidad se debe a que la ubicuidad de las tecnologías favorece el aumento del número de oportunidades laborales para profesionales de la computación (Marés, 2014), otra razón que resulta fundamental se relaciona con la idea de formar personas que no sólo sean consumidoras de tecnología sino también puedan expresarse a través de ellas. Es decir que puedan por ejemplo, crear sus propias páginas Web, componer música y crear sus propios mundos simulados en vez de sólo acceder a páginas, bajar música o participar en ambientes virtuales (Resnick, 2002).

A partir del contexto expuesto surge la necesidad de replantear las prácticas educativas en la materia "Taller de Programación" del 4º Año de la orientación "Técnico en Informática Profesional y Personal" de la EEST Nº2 de la ciudad de Tandil. Desde el inicio de la especialidad se abordaron en esta asignatura los contenidos de programación utilizando Pascal[1]. Este lenguaje presenta un entorno de desarrollo limitado frente a los nuevos lenguajes creados para facilitar la introducción de la programación en la escuela tales como Scratch, Alice, Kodu, RoboMind, RoboLab, AppInventor, etc. (López García, 2013). En el presente trabajo, desde una visión socio constructivista del aprendizaje (Castorina y col., 1999), se plantea la propuesta de enseñanza para programación utilizando RoboMind que comenzó su implementación en el ciclo lectivo en curso. Se describe la fundamentación teórica, el entorno de programación utilizado y los aspectos más relevantes de su desarrollo en el aula, entre los que se destacan la alta motivación y compromiso con la tarea que demostraron los alumnos.

2. Fundamentación de la propuesta de enseñanza

Aprender a programar es una tarea compleja que involucra la utilización de

estrategias para la resolución de problemas, diseño de proyectos y comunicación de ideas (Resnick, 2013). Esto genera un escenario problemático en el que los métodos tradicionales no se pueden adaptar de manera adecuada para la enseñanza de la programación. Se requiere de enfoques centrados en el alumno que les permitan desarrollar experiencias promotoras de nuevos aprendizajes (Miliszewska & Tan, 2007).

Desde el constructivismo social (Vygotsky, 1978) el aprendizaje no es un proceso puramente interno, sino un constructo social mediado por el lenguaje utilizado en el discurso social. En esta perspectiva la noción de aprendizaje significa proceso de enseñanza-aprendizaje e incluye al que aprende, al que enseña y la relación social entre ellos (Castorina y col., 1999). "El aprendizaje surge de la puesta en relación entre lo que el aprendiz aporta al acto de aprender y los elementos y componentes de las situaciones y actividades en las que se desarrolla este acto" (Coll, 2010, p. 35).

El profesor orienta y guía el proceso de construcción de significados y de atribución de sentido a los contenidos a enseñar. Coll (2010) destaca dos formas de intervención dinámica del docente: *ayudas distales* y *ayudas proximales*. Las primeras se relacionan con el diseño y la planificación de actividades (selección y organización de los contenidos, tipos de actividades, las consignas y pautas de realización, los materiales y los recursos, previsiones de desarrollo de la clase, etc.). Las segundas tienen lugar el aula durante el desarrollo de las actividades de enseñanza y aprendizaje a través de las interacciones entre profesor y alumno y entre alumnos entre sí (preguntas y respuestas, interpelaciones, pautas de trabajo, explicaciones, etc.).

Para proyectar prospectivamente ambos tipos de ayuda es de interés reflexionar sobre el concepto de "zona de desarrollo próximo" (ZDP). Vygotsky (1978) la definía como "la distancia entre el nivel real de desarrollo, determinado por la capacidad de resolver independientemente un problema, y el nivel de desarrollo potencial, determinado a través de la resolución de un problema bajo la guía de un adulto o en colaboración con otro compañero más capaz". Estos espacios construidos en la interacción con profesores y compañeros de estudio serían en los que tendrían lugar los procesos de construcción de significados y de atribución de sentido a los contenidos escolares. En esta dirección, el diseño y el seguimiento de las prácticas áulicas encuentran un marco teórico adecuado en la

Teoría de la Actividad (TA). En el modelo de Engeström (1987) un Sistema de Actividad (SA) se compone de personas, herramientas, un objetivo o motivo, reglas socioculturales y roles. La noción de sistema refiere no sólo a los elementos sino también a las relaciones entre ellos. Esto permite diseñar en forma dinámica una situación de enseñanza y aprendizaje considerando la evaluación del sistema y su modificación a través de la retroalimentación. Lo que contribuye a brindar ayuda distal y ayuda proximal ajustadas a las experiencias de construcción de conocimientos de los estudiantes (Coll, 2010).

3. Descripción de la propuesta de enseñanza para la materia "Laboratorio de Programación"

La propuesta de enseñanza se llevó a cabo en la materia "Laboratorio de Programación" del 4º Año de la EEST Nº2 "Ing. Felipe Senillosa" de la ciudad de Tandil. Este espacio pertenece a la Formación Técnico Específica de la estructura curricular Técnico en Informática Profesional y Personal.

La implementación de las actividades se realizó en el primer cuatrimestre del año 2014 en las dos divisiones del 4º Año con una carga horaria de dos horas semanales.

Las clases tuvieron lugar en el Laboratorio de Informática de la Escuela debido a que no todos los alumnos, por diferentes razones, tenían disponibles las netbooks otorgadas por el Programa "Conectar Igualdad.com.ar". Por lo cual algunos estudiantes compartían en grupos de dos una computadora. Aún teniendo su propio equipo los alumnos resolvieron las actividades en interacción con sus compañeros, consultándose y compartiendo logros y dudas. Los docentes promovieron que los alumnos intentaran resolver los problemas con la menor ayuda posible, monitoreando el desarrollo de las actividades y haciendo devoluciones que les permitieran seguir avanzando. Ante dificultades en la comprensión del problema a resolver o ante la presencia de obstáculos generalizados en el desarrollo de la actividad se realizaron puestas en común para brindar explicaciones o elaborar una solución en colaboración.

3.1 Diseño de la propuesta

La meta principal de la propuesta de enseñanza se orienta a promover en los alumnos el desarrollo de sus primeras experiencias en programación en el marco de la programación estructurada. Para ello se propone fomentar aprendizajes en los alumnos de los primeros conceptos (ver Tabla 1) y prácticas (ver Tabla 2) en el desarrollo de actividades planteadas como desafíos con un nivel de complejidad incremental.

Concepto	Descripción
Secuencia	Identificación de una serie de pasos para una tarea.
Bucles (loops)	Ejecución de la misma secuencia varias veces.
Condicionales	Toma de decisiones basada en condiciones.
Operadores lógicos	Producen un resultado booleano.

Tabla 1. Conceptos de programación

Práctica	Descripción
Abordar la creación de programas de manera iterativa e incremental	Desarrollar un poco, probarlo y desarrollar un poco más.
Probar y depurar	Comprobar el buen funcionamiento del programa. Encontrar y corregir errores.
Reutilizar código	Incorporar código de otros programas en la construcción de programas nuevos.
Escribir de manera clara de los programas	Utilizar comentarios e indentación para facilitar la lectura y comprensión a otros.

Tabla 2. Prácticas de programación

Este planteo de la enseñanza inicial de la programación utilizando RoboMind intenta superar algunos obstáculos observados con el uso de Pascal como primer lenguaje de programación. Entre ellos se pueden citar los inconvenientes derivados de utilizar las instrucciones de programación en idioma inglés y la dificultad que representa el tipo de problemas que se pueden resolver, en cuanto a su complejidad por el nivel de abstracción que requieren.

El diseño de la curso se realizó sobre la base del modelo “Diseño de propuestas de e-formación colaborativa” (Cenich, 2009) que propone plantear de manera integral los distintos elementos de una propuesta de enseñanza mediante la definición de un Sistema de

Actividad Cero (SA 0) (Engeström, 1987). Se presenta a continuación la descripción de los componentes del SA general del curso:

Objeto: promover en los alumnos aprendizajes de conceptos y prácticas iniciales de programación estructurada.

Resultado: programas realizados por los alumnos.

Sujeto: alumno.

Comunidad:

-Alumnos: del 4º Año de la orientación Técnico en Informática Profesional y Personal.

-Practicantes: dos alumnos del Profesorado en Informática que realizaron sus prácticas docentes.

-Docentes del curso.

División del trabajo:

-Alumnos: resolver los desafíos propuestos.

-Practicantes y docentes del curso: acompañar, guiar y animar el proceso de aprendizaje individual y grupal, observando y evaluando el desarrollo general del curso para mediar en consecuencia. Evaluar las producciones finales.

Herramientas: RoboMind, Apuntes del profesor.

Reglas:

-Participar en las actividades, proponer, discutir, valorar y negociar significados a partir de las formulaciones de docentes y alumnos.

-Tener actualizado el registro completo de las actividades desarrolladas.

El SA 0 está compuesto por tres actividades componentes:

-*"Acciones básicas"* (**SA 0.1**): tiene como objetivo principal que el alumno se familiarice con el entorno de trabajo y elabore sus primeros programas para que el robot se desplace y pinte en el escenario propuesto.

-*"Andar y decidir"* (**SA 0.2**): se plantea como meta que el alumno logre escribir programas en los que los movimientos del robot estén supeditados a la toma de decisiones.

-*"Mejorar las condiciones"* (**SA 0.3**): tiene como fin profundizar en el planteamiento de las condiciones a través de la utilización de operadores lógicos.

3.1.1 Entorno de programación RoboMind

RoboMind es un lenguaje de programación que está diseñado para familiarizar a las personas con las reglas básicas de las ciencias de la computación mientras programan las acciones para seguir por el robot.

Se utilizó la versión 2.6 de RoboMind debido a que a partir de la versión 3.0 el software dejó de ser open source y gratuito.

En el entorno de trabajo se observan dos áreas principales una destinada a la edición del código y otra que muestra las acciones del robot al ejecutar el script (ver Figura 1). Esto facilita la prueba y depuración de los programas, ya que a través del botón "Play" (ubicado en la parte inferior izquierda del entorno) se pueden ir observando los resultados parciales a medida que se escribe el código.

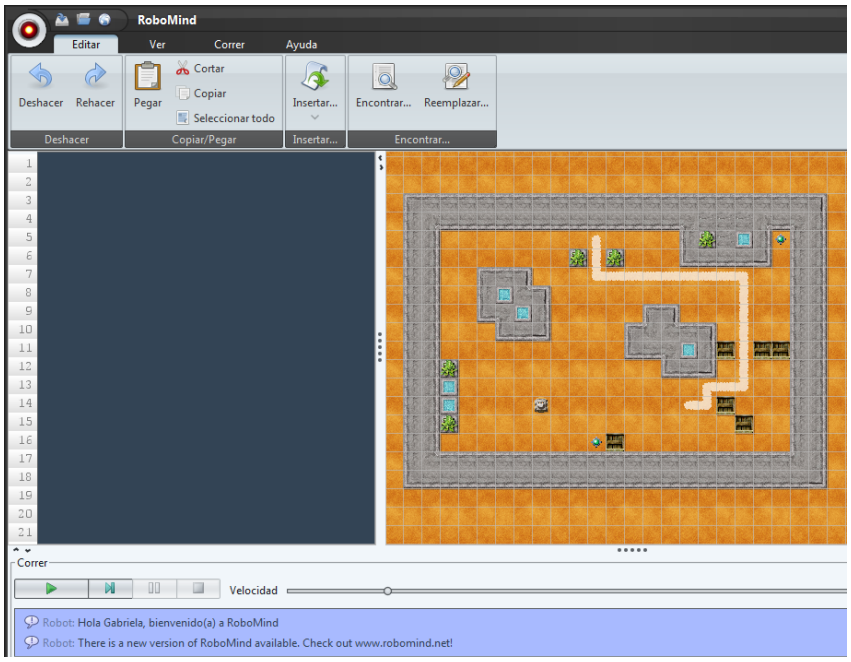


Figura 1. Entorno de trabajo del software RoboMind

La edición del código se ve agilizada por presentar una versión en español y disponer de las instrucciones a través del botón "Insertar" lo que disminuye los errores de escritura.

El robot se desplaza sobre un mapa que puede ser elegido entre un conjunto básico que trae por defecto la versión o también se pueden diseñar nuevos escenarios aparte e incorporarlos.

Entre las instrucciones básicas del robot se encuentran las que permiten su desplazamiento, mover una baliza (objeto que se puede incorporar a un mapa) y pintar, se destacan además la posibilidad de insertar bucles y estructuras de selección para las cuáles el robot utiliza su sensor para ver al frente, izquierda y derecha (ver Figura 2).

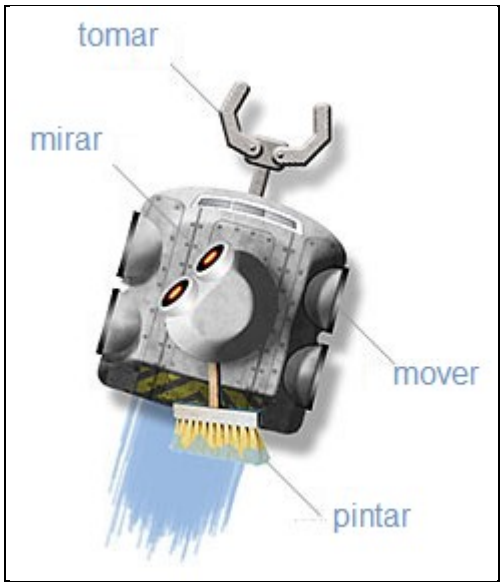


Figura 2. Descripción del robot

Se destacan en la página del software cuatro ventajas que presenta su utilización:

-*Comienzas desde ya*: el lenguaje de programación y plataforma son fáciles de comprender y se pueden usar rápidamente.

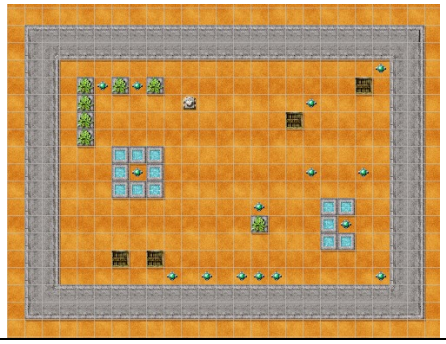
-*Sin dependencias externas:* como entornos de desarrollo y compiladores que compliquen las cosas.

-*Seguro:* el programa que se haga no puede dañar de ninguna forma tu computadora.

-*Apto para proyectos:* es la herramienta ideal para proyectos interdisciplinarios o para cursos técnicos o de ciencias de la computación.

4. Resultados

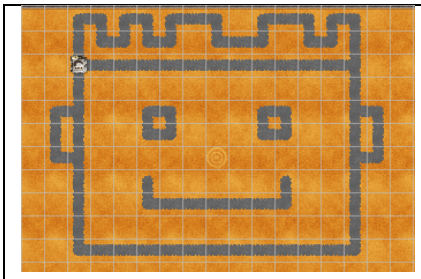
El SA *"Acciones básicas"* se desarrolló a través del planteo de un conjunto de desafíos que comprendieron desde tareas sencillas, de familiarización con el entorno y con las posibilidades de movimiento y de pintar del robot, hasta actividades más complejas en las cuales se requirió pensar una estrategia para resolver el problema. Se describen a modo de ejemplo algunas de las actividades desarrolladas por los alumnos:

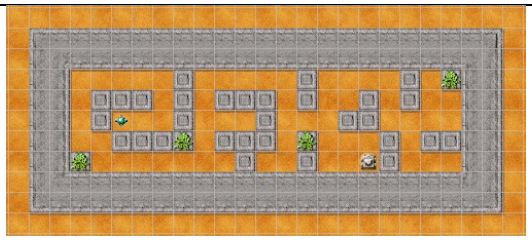
Escribe el script que permita al robot encerrar con una línea negra 3 balizas en la menor superficie posible en el mapa "rm-lf-task5.map" 

Los alumnos elaboraron diferentes soluciones, el docente promovió una puesta en común para acordar entre todos cuál sería la mejor solución y por qué. Se observó que algunos alumnos no habían comprendido bien el problema por lo que habían escrito un código que realizaba una tarea que no daba respuesta adecuada a la cuestión planteada.

El último ejercicio de esta serie proponía: "Plantear un desafío a resolver por tus compañeros". Se observó en el desarrollo de esta actividad un gran compromiso de la mayoría de los alumnos por crear un desafío original y que tuviese cierta complejidad para resolverlo. Algunos ejemplos de las propuestas de los alumnos son:

-Marcar con una línea negra un espiral partiendo desde el centro del mapa, los espacios restantes pintarlos de blanco. Se debe utilizar el mapa "openarea.map"	
-Elegir el mapa openarea.map. Dibujar un espiral desde afuera hacia adentro pintando 4 líneas blancas, 4 negras.	
-Dibujar la siguiente cara:	-Elegir el mapa maze2.map. Pintar de color blanco todas las zonas de tierra del mapa seleccionado.





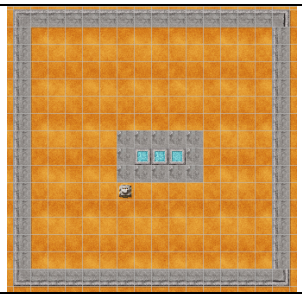
A medida que los ejercicios fueron elaborados se ponían a disposición del resto de la clase para que intentara resolverlos, esto generó consultas e intercambio de ideas entre los creadores del desafío y sus compañeros.

A la vez que los alumnos avanzaron en las resoluciones afianzaron como herramienta el entorno de trabajo de RoboMind, lo que les permitió incorporar las instrucciones para tomar y depositar una baliza para proponer nuevos desafíos a sus compañeros.

La evaluación de este SA en forma dinámica permitió agregar y modificar los ejercicios planteados atendiendo a las dificultades y logros de los alumnos en el marco de las reglas planteadas y de la transición del entorno de trabajo de ser objeto a herramienta de mediación.

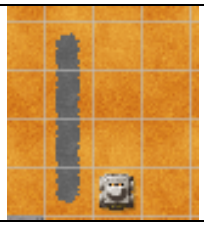
En el SA *"Andar y decidir"* se propuso a los alumnos resolver problemas en los que el robot tendría que determinar sus acciones de acuerdo a la evaluación de ciertas condiciones para lo cual utilizaría sus sensores. Se planteó en la en pizarra el siguiente desafío:

Utilizando el escenario "mapa_if_1.map" logra que el robot gire alrededor del obstáculo.



Los alumnos propusieron una primera solución de acuerdo a los conocimientos que tenían hasta el momento. A través de preguntas el docente guió a los estudiantes hacia la elaboración de algoritmos más generales que se pudieran aplicar en diversas situaciones: ¿cómo podría el robot recorrer un obstáculo sin conocer sus dimensiones?, ¿realizar más de una vuelta?, ¿cuántas vueltas?. Los alumnos compartieron ideas y buscaron en el entorno de trabajo aquellas instrucciones que pudieran utilizarse para resolver las cuestiones propuestas. Luego de elaborar el programa en sus máquinas se les plantearon ejercicios en los que serían necesarios utilizar ciclos y condiciones para poder resolverlos de manera adecuada. Algunas tareas realizadas fueron:

a) Abre el mapa copyLine1.map y has que el robot pinte puntos blancos a la derecha de la línea negra.
b) Modifica el programa anterior para que una vez que terminó de pintar los puntos blancos los una formando una línea blanca.



Se observó en general que los alumnos elaboraron dos tipos de algoritmos para realizar la tarea (ver Tabla 3). Uno que empleaba sólo la secuencialidad y otro que utilizaba las estructuras de ciclo y decisión. Esto permitió evaluar en una puesta en común las limitaciones y ventajas de cada tipo de solución.

```
pintarBlanco()
detenerPintar()
adelante(1)
pintarBlanco()
detenerPintar()
adelante(1)
pintarBlanco()
detenerPintar()
adelante(1)
pintarBlanco()
detenerPintar()
```

```
repetir(4){
   si(izquierdaEsNegro() ){
      pintarBlanco()
      detenerPintar()
      adelante(1)
      }
}
```

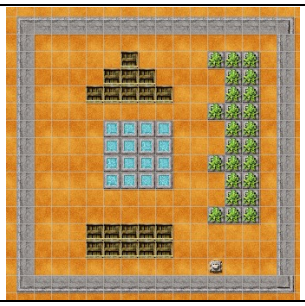
Tabla 3. Algoritmos planteados por los alumnos

Se propusieron actividades que incrementaban su complejidad hasta llegar al concepto de selecciones anidadas. Se planteó además la necesidad de escribir programas

legibles utilizando la indentación y comentarios para poder compartirlos con los compañeros.

En la evaluación de este SA se destaca que los alumnos requirieron más apoyo del docente para poder realizar las tareas y se utilizó la pizarra para aclarar dudas generales del grupo a través del diálogo y la participación de los alumnos. Esto permitió que alumnos más avanzados colaboraran con otros guiándolos en la elaboración de la solución.

En el SA *"Mejorar las condiciones"* se plantearon desafíos que permitieron a los alumnos indagar y evaluar sobre diferentes maneras de establecer condiciones para obtener un mismo resultado. El docente diseñó el mapa "obstáculos.map" y propuso el siguiente desafío:

Escribe un programa que permita al robot pintar una línea de color alrededor de cada obstáculo. Al pintar el siguiente obstáculo debe hacerlo en sentido contrario al que pintó el anterior. Utiliza el mapa "obstáculos.map"	

En la resolución del ejercicio algunos alumnos observaron que ya habían escrito código que les serviría para recorrer el perímetro del cuadrado y del rectángulo. Lo reutilizaron haciéndole las modificaciones necesarias para ajustarlo a la nueva situación.

Para rodear la figura del triángulo escalonado encontraron dificultad al llegar al vértice superior, pero observando cómo se desplazaba y giraba su cabeza el robot lograron la mayoría de los alumnos elaborar una solución. A continuación se muestra el código que se elaboró en colaboración en la pizarra:

```
repetir(20){
  si(derechaEsObstaculo() y frenteEsClaro()){
    adelante(1)
    }otro{
      derecha()
      adelante(1)
      si( derechaEsObstaculo()){
        izquierda()
```

```
    }
  }
}
```

Para la última figura los alumnos trabajaron con el código anterior tratando de realizarle las adaptaciones que les permitieran cumplir con el desafío.

Se destaca en la evaluación de este SA la profundización en las prácticas que son objeto de la presente propuesta: *abordar la creación de programas de manera iterativa e incremental*, *probar y depurar, reutilizar código y escribir de manera clara de los programas*. Además se destaca como retroalimentación del sistema la posibilidad de compartir y evaluar diferentes algoritmos que utilizaban distintos operadores lógicos en la definición de las condiciones.

5. Comentarios finales

César Coll (2010, p. 43) se pregunta "¿Cómo podemos ayudar a aprender a las personas?". En este trabajo se reformula la pregunta y se intenta aproximar una respuesta a la cuestión ¿Cómo podemos ayudar a aprender a programar a los alumnos?.

Para ello se planteó una propuesta fundamentada, con un diseño dinámico en el sentido que permite evaluar y realizar los ajustes adecuados de acuerdo a la evolución de la actividad. Un diseño que contempla a los alumnos, los docentes, los contenidos y sus relaciones en el marco de una actividad potencialmente promotora de aprendizajes.

De acuerdo a lo expresado por Coll (Ibíd., p.35) "El aprendizaje surge de la puesta en relación entre lo que el aprendiz aporta al acto de aprender y los elementos y componentes de las situaciones y actividades en las que se desarrolla este acto". En esta experiencia los conocimientos previos de los alumnos sobre programación eran escasos y generales, por lo cual fue muy importante la contribución a la motivación promovida por la incorporación de RoboMind como primera herramienta de programación. El interés y compromiso de los estudiantes en las actividades desarrolladas se puso de manifiesto por la instalación del software en los equipos hogareños y los avances de algunos alumnos realizados fuera del horario escolar.

La pregunta planteada al comienzo se reformula en cada aula con las características propias de su contexto, por ello esta propuesta tiene la finalidad de compartir un diseño y una experiencia para abrir un diálogo que permita avanzar en la construcción de nuevas respuestas.

Referencias bibliográficas

Castorina J., Ferreiro E., Kohl M., Lerner D. (1999). Piaget-Vigotsky: contribuciones para replantear el debate. Argentina. Paidós Educador.

Cenich, G. (2009). Tesis: Diseño de propuestas de e-formación colaborativa: un modelo desde la perspectiva de la Teoría de la Actividad. Magíster en Tecnología Informática Aplicada en Educación, Universidad Nacional de La Plata, La Plata.

Coll, C. (2010). Enseñar y aprender, construir y compartir: procesos de aprendizaje y ayuda educativa, en C. Coll (coord.). Desarrollo, aprendizaje y enseñanza en la educación secundaria. España: Editorial Graó, 31-61.

ISTE, NSF & CSTA (2012). Pensamiento Computacional Una habilidad de la era digital al alcance de todos. Recuperado 12/05/14 de http://www.eduteka.org/modulos/9/272/2082/1

Engeström, Y. (1987). Lerning by expanding: An activity-theoretical approach to developmental research. Helsinki, Orienta-Konsultit.

Jonassen, D. (1997). Instructional Design Models for Well-Structure and Ill- Structure Problem-Solving Learning Outcomes. Educational Technology: Research and Development, 45 (1), 65-95.

Levis, D. (2006). Alfabetos y saberes: la alfabetización digital. Comunicar, vol. XIV, Nº 026, 78-82. Recuperado 01/05/14 de http://recyt.fecyt.es/index.php/comunicar/article/view/25911

López García, J. (2013). Programación de computadores, un asunto de interés para todos. Recuperado 10/05/14 de http://www.eduteka.org/code.php

Marés L. (2014). La programación y los niños. Mirada-Relpe Reflexiones Iberoamericanas sobre las TIC y la Educación, 90-95. Recuperado 05/05/2014 de http://www.relpe.org/publicaciones-relpe/

Miliszewska, I. & Tan, G. 2007. Befriending Computer Programming: A Proposed Approach to Teaching Introductory Programming. Issues in Informing Science and Information Technology, 4.

Resnick, M. (2013). Learn to Code, Code to Learn. Recuperado 01/05/14 de https://www.edsurge.com/n/2013-05-08-learn-to-code-code-to-learn

Resnick, M. (2012). Reviving Papert's Dream. Educational Technology, vol. 52, no. 4, 42-46.

Resnick, M. (2002). Rethinking Learning in the Digital Age. In The Global Information Technology Report: Readiness for the Networked World, edited by G. Kirkman. Oxford University Press. Recuperado 03/05/14 de llk.media.mit.edu/papers/mres-wef.pdf

Vygotsky, L. (2000). El desarrollo de los procesos psicológicos superiores. Barcelona. Crítica (Trabajo original publicado en 1978).

Wing, J.M. (2010). Demystifying computational thinking for non-computer scientists. Unpublished manuscript in progress. Recuperado 08/05/14 de http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

[1]Lenguaje concebido por Niklaus Wirth en los años 70 para la enseñanza de técnicas de programación a estudiantes universitarios. En la actualidad se utiliza en los primeros años de carreras de Informática.