Programación de computadores y desarrollo de habilidades de pensamiento en niños escolares: fase exploratoria.

Hernando Taborda
Diego Medina
Universidad ICESI
Cali

Julio, 2012

AGRADECIMIENTOS
Agradecemos la ayuda brindada por Juan Carlos López García, editor de EDUTEKA, por su colaboración en el planteamiento del problema de investigación; así mismo, agradecemos a los directivos y profesores del Instituto Nuestra Señora de la Asunción (INSA) por abrirnos las puertas de su institución, y a Rodrigo Rosero por la recolección de los datos.

MARCO TEÓRICO

1. Pensamiento Computacional

La noción de pensamiento computacional es de aparición relativamente reciente en el campo de la investigación educativa y psicológica, así, la primera referencia explícita a este concepto aparece en un artículo escrito por Jeannette Wing en el 2006. Como

veremos, sin embargo, hay desarrollos mucho más tempranos de este concepto, enfocados principalmente en estudios comparativos expertos-novatos en programación.

Como Wing (2008) señala, el pensamiento computacional puede comprenderse como una nueva forma de pensamiento posibilitada gracias a la aparición de los sistemas computacionales en el siglo XX, y que implica la unión o mezcla entre pensamiento matemático, pensamiento científico y pensamiento ingenieril. Con lo cual Wing intenta mostrar que pensar computacionalmente requiere un nivel alto de abstracción, pero al mismo tiempo requiere anclar fuertemente estos pensamientos a lo real, para solucionar problemas concretos. De hecho, el auge que han tenido las herramientas computacionales en las últimas décadas muestra constantemente esta doble faz entre lo abstracto-formal y lo concreto-cotidiano. De forma específica podemos señalar dos campos de aplicación en donde esto se observa claramente: el modelado de sistemas y la solución algorítmica de problemas.

El modelado es una herramienta fundamental hoy día en múltiples disciplinas científicas, desde la cosmología hasta la psicología, pasando por la meteorología. Un supuesto importante en este amplio rango de campos de estudio es que el funcionamiento de cualquier sistema es susceptible de ser representado simbólicamente y simulado a través de algoritmos computacionales. Modelar un sistema aumenta notablemente las capacidades explicativas y predictivas de las teorías científicas. Por otro lado, la solución algorítmica de problemas se acerca a lo que Wing explica es uno de los componentes principales del pensamiento computacional, la solución de problemas mediante el uso de computadores. Entendiendo por “problema” un amplio espectro de situaciones, desde ejecutar las operaciones aritméticas elementales hasta animar mundos virtuales. En estos dos campos de aplicación, el conocimiento lógico-matemático es patente (ej. a través del uso de conectores lógicos, de operaciones aritméticas o de conocimiento algebraico), pero al mismo tiempo se trata de usarlo para solucionar problemas concretos con resultados visibles en el espacio virtual de un computador.

Una pregunta importante a plantear desde un punto de vista educativo y psicológico es entonces, ¿cuáles son los conocimientos implicados en el uso de herramientas computacionales para solucionar problemas? Una forma de dar respuesta a este interrogante siguiendo a Pane (2001) y también a Wing (2008) es señalando que el pensamiento computacional, en cualquiera de sus formas de aplicación, requiere tanto habilidades de programación como conocimiento conceptual sobre computación. Programar y conocer sobre computación, podrían parecer a primera vista conocimientos reservados para estudiantes de ingeniería, sin embargo, como se sostuvo atrás, estos son conocimientos cada vez más imbricados en un amplio espectro de actividades científicas.

No obstante, el interés de psicólogos y educadores por las actividades de programación de computadores excede el hecho de su rol instrumental en la ciencia. Una de las tesis más importantes defendidas por los investigadores en el campo de la

educación en computación es que aprender a programar fomenta una forma de pensamiento más *abstracta*, *analítica* y *eficiente*. Gran parte del sentido de comprender qué quiere decir "pensamiento computacional" recae en estos conceptos. Un ejemplo permitirá aclarar esta conexión: si se desea desplazar una pelota de un lugar A a un lugar B, bastaría con tomar la pelota con la mano, levantarla y desplazarla hasta el lugar B, de hecho esta acción apenas si podría representarse como un problema. Sin embargo, si se plantea ejecutar esta misma acción a través de un dispositivo computacional en un espacio virtual, la solución podría asemejarse a una estructura como la siguiente: MOVER ADELANTE (objeto x) 50 centímetros. Si de forma adicional se quiere hacer un segundo desplazamiento hacia la derecha se podría agregar: GIRAR DERECHA 90 grados (objeto x), MOVER ADELANTE 50 centímetros. En ambos casos la representación del movimiento implica un análisis de la estructura geométrica espacial del desplazamiento del objeto, en términos de distancias, ángulos y orientación. Al mismo tiempo implica un análisis en términos matemáticos y uso de sistemas de medición, como centímetros y ángulos. Esta descomposición de factores o variables conlleva a "pensar en múltiples niveles de abstracción" (Wing, 2006). Así, un triángulo deja de ser un percepto para convertirse en una organización de distancias y ángulos, independientemente de su tamaño, color o grosor de las líneas.

Por otro lado, la forma como se encadena la secuencia de acciones programadas puede seguir múltiples vías, las cuales crecen de forma exponencial en función de la complejidad de la acción. En este sentido, un nuevo requerimiento adicional en los procesos de programación es cómo encontrar una secuencia eficiente, para economizar tiempo y energía. Al respecto, Wing (2008) agrega que la forma como una secuencia se hace eficiente es a través de la automatización de acciones, es decir, hallar una secuencia eficiente y utilizarla de forma reiterada en múltiples situaciones sin necesidad de volver a encontrar la solución. Esta orientación hacia la eficiencia está estrechamente ligada a la consideración de las restricciones propias del sistema computacional utilizado, tales como velocidad de procesamiento de datos y memoria disponible.

Si bien la actividad de programación requiere la utilización de un pensamiento abstracto, eficiente y analítico, esto no agota la naturaleza del pensamiento computacional. En los artículos de Wing es posible encontrar algunos rasgos adicionales esenciales a esta forma de pensamiento. Por ejemplo, cuando se programa una simulación (ej. el funcionamiento de un ecosistema) es necesario pensar en paralelo, es decir, que la acción de un objeto (ej. crecimiento de las plantas) desencadena series de acciones simultaneas e interdependientes (ej. más depredadores y más oxigeno). La actividad de programación debe capturar este rasgo del mundo.

En conjunto, la conceptualización del pensamiento computacional parece extenderse en dos vías. Primero, como la unión de varias formas de pensamiento más elementales (matemático, ingenieril y científico) a través del uso de dispositivos

computacionales y como una serie de características emergentes, tales como el procesamiento en paralelo. Segundo, como una poderosa herramienta de abstracción y análisis de problemas científicos que recuerdan la teoría de cambio cognitivo propuesta por Karmiloff-Smith (1994), en donde el avance en el conocimiento depende del formato de representación, así cuando los movimientos se representan a través de comandos y acciones condicionadas en un espacio virtual la estructura geométrica se torna más explícita y susceptible de análisis formales.

2. Aprendizaje de Programación

Como se mostró en un capítulo precedente un componente importante del concepto de "pensamiento computacional" es la habilidad de programar, sea con el fin de crear simulaciones o solucionar problemas. En este sentido una pregunta importante es ¿cómo aprende un novato en ciencias de la computación a hacerse experto en programación? Y por extensión ¿cómo podría aprender un niño a programar en el aula de clases? Si bien estas preguntas no han sido extensamente abordadas en años recientes se han realizado algunos estudios empíricos que dan algunas luces sobre sus respuestas. No obstante, incluso antes haber iniciado estos estudios algunos investigadores ya habían señalado la que a su juicio es la principal fuente de dificultad en el aprendizaje de programación, a saber, el tipo de lenguaje empleado para programar. Así, investigadores como Lance Miller en los 70s y John Pane en los 90s mostraron las grandes diferencias entre la sintaxis y la semántica de los lenguajes naturales y los lenguajes de programación. Por ejemplo, los lenguajes naturales tienden a tener una estructura declarativa mientras que los lenguajes de programación una estructura imperativa, haciendo de la primera una estructura más ambigua. La diferencia, sin embargo, es especialmente marcada en la semántica de los conectores lógicos, así, el conector *entonces* es entendido en programación como *en esta condición,* en lugar de *después* como ocurre en el lenguaje natural. Como Pane (2001) aclara este tipo de diferencias entre lenguajes no se resuelve igualando la estructura de la programación al lenguaje natural, pues el proceso de programación debe hacer totalmente explícita la información para ejecutar los comandos, mientras que en el lenguaje natural muchos elementos se dejan a la inferencia a partir del contexto conversacional y conocimiento previo.

No obstante, a pesar de las diferencias irreductibles entre ambos lenguajes, es importante tratar de reducir las distancias a través de la creación de lenguajes similares en algunos aspectos al lenguaje natural. Desde que Miller y Pane hicieran explícita la necesidad de modificar los lenguajes de programación para facilitar su aprendizaje, se han propuesto varias alternativas de lenguajes orientados a objetos para niños y adolescentes, tales como HANDS desarrollado en la Carnegie Mellon University por John Pane o SCRATCH desarrollado en el MIT por Mitchel Resnick. La estructura y ventajas de uso de este último programa se trabajarán en la siguiente sección.

Una serie de estudios empíricos llevados a cabo principalmente en los últimos 10 años han revelado algunas dificultades más específicas asociadas al proceso de aprendizaje de programación. A partir del 2006 se inició en Estados Unidos el proyecto "Commonsense Computing" cuyo objetivo principal es determinar ¿qué saben los estudiantes de programación antes de iniciar estudios formales sobre computación? Una de las primeras tareas utilizadas fue de "clasificación" (Simon, Chen, Lewndowski, McCartney y Sanders, 2006), en la cual se pedía a los participantes escribir cómo harían ellos para ordenar una serie de 10 números en orden ascendente, de tal manera que la solución fuera valida tanto para los 10 números de ejemplo como para cualquier otra cadena de números. Participaron tres grupos de personas: estudiantes a punto de iniciar un curso de ciencias de la computación (principiantes), un grupo de estudiantes de ciencias económicas (novatos), y un grupo de estudiantes de ciencias de la computación después de 11 semanas de clases. Se quería responder a cuatro preguntas, ¿Cuál es la proporción de acierto en las respuestas? ¿Qué tipo de solución emplean los estudiantes en esta tarea? ¿Se utilizan estructuras de control (iteración y condicionales)? Y ¿Qué caracteriza el contenido de las respuestas? De forma sumaria, se encontró que el 57% de los principiantes expresaron las respuestas de forma correcta versus un 25% de los novatos. En cuanto al tipo de solución utilizada se encontró que el 35% de los principiantes manipulan los números como unidades primitivas y el 63% como una cadena de dígitos. Los valores para los novatos fueron de 36% y 53%, respectivamente. Quizás el resultado más interesante fue respecto al uso de estructuras de control, así el 65% de los principiantes expreso iteraciones y el 43% expresó condicionales, mientras que los porcentajes fueron algo más bajos para los novatos, 56% y 25%, respectivamente. Finalmente, el grupo de principiantes tendían a usar de forma más frecuente términos de la ciencia computacional (M=1.8) que el grupo de novatos (M=0.8).

La investigación de Simon et. al. (2006) muestra en definitiva que la diferencia de desempeños entre principiantes y novatos es importante y constante en casi todas las mediciones, siempre a favor de los primeros. Esto indica que las personas que ingresan a carreras afines a ciencias de la computación tienden a tener mejores habilidades de programación (en contraste también con los estudiantes más avanzados de ciencias de computación quienes mejoran notablemente su desempeño). Sin embargo, más allá de estas diferencias, los investigadores recalcan que sin instrucción formal es posible encontrar soluciones estructuralmente correctas y acordes con los lenguajes de programación. Estos resultados son similares a los hallados por Pane, Ratanamahatana y Myers (2001) en una investigación realizada con niños entre los 10 y 11 años de edad. En el primer estudio reportado, se pidió a los niños describir cómo harían ellos (en el lugar de un computador) para mover un Pacman dentro de una sección del laberinto en el popular juego de los 80s. Los resultados de este estudio mostraron la dificultad de los niños para usar estructuras de control, de hecho, sólo el 20% de la muestra uso procesos de itaración, y hubo una prevalencia de uso de los condicionales

de 0.4 por participante. En un segundo estudio se observó que las palabras clave (keywords) que señalaban algún tipo de conexión lógica (AND, OR, BUTNOT, THEN, ELSE) se usaban de forma incorrecta; así por ejemplo, en las oraciones en donde se usaba el condicional Y (AND) el 76% de las ocasiones se hacía de forma incorrecta.

FORMULACIÓN DEL PROBLEMA

Desde los años 80s cuando ingresaron al mercado los computadores personales, la necesidad de solucionar problemas a través de esta herramienta electrónica ha ido en aumento. Desde herramientas de uso masivo como Internet, hasta programas especializados como SPSS, cada vez más los programas informáticos permean nuestra vida cotidiana. En este nuevo ambiente social, se hace cada vez más importante comprender el significado de lo que Wing (2006; 2008) llama "pensamiento computacional", y construir además dispositivos educativos que permitan su promoción eficaz. Con este propósito, algunos investigadores han enfatizado en la necesidad de enseñar a programar computadores en las escuelas y colegios como un componente clave para fomentar el desarrollo del pensamiento computacional desde edades relativamente tempranas. En este marco, el entorno gráfico de programación SCRATCH desarrollado en el MediaLab del MIT en Estados Unidos se ha propuesto como un instrumento importante para alcanzar este objetivo. No obstante, y a pesar de la amplia aceptación que ha tenido en la comunidad de educadores, no hemos encontrado investigaciones publicadas que hayan informado acerca del impacto que el uso del SCRATCH genera en el aula de clases. Por otro lado, tampoco hemos encontrado estudios publicados en donde se analice la forma como el SCRATCH podría hacer frente a las dificultades reportadas en algunas investigaciones que tienen los niños al programar (Pane, 2001), tales como la dificultad de comprender el significado de los conectores lógicos. No obstante, es de destacar que en una investigación realizada por Rosenbaum sobre el uso del SCRATCH se encontró que la capacidad de socializar los resultados e incluir espacios de expresión emocional mejoraba la comprensión del sistema y motivaba a los niños a continuar su proceso de aprendizaje.

Por lo tanto, debido a la notable carencia de investigaciones que liguen el uso de SCRATCH al pensamiento computacional surgen varias preguntas específicas. Así, por ejemplo, ¿Hasta qué punto este programa informático realmente genera aprendizaje de habilidades de programación? ¿Qué tipo de actividades deben proponerse para hacer un uso efectivo del entorno de programación? ¿Genera un cambio a nivel del pensamiento más allá del aprendizaje de comandos? La presente investigación busca iniciar un trabajo sistemático en torno a estos interrogantes con niños colombianos. En general, nuestra pregunta de investigación es entonces la siguiente: **¿De qué forma el uso del entorno gráfico de programación SCRATCH, junto con las actividades pedagógicas propuestas por los maestros en el Instituto de Nuestra Señora de la**

Asunción (INSA, Cali, Colombia), promueven el desarrollo del pensamiento computacional y el aprendizaje de habilidades de programación en niños de grado 3º de primaria?

OBJETIVOS

Objetivo General:
Examinar y describir la forma como el entorno gráfico de programación SCRATCH y la demanda de las tareas propuestas en clase promueven el desarrollo tanto del pensamiento computacional como de habilidades de programación.

Objetivos Específicos:

1. Caracterizar la demanda cognitiva de los diferentes tipos de actividades de programación con SCRATCH propuestos a niños de grado 3º de primaria del INSA.
2. Describir las dificultades que los niños experimentan durante la realización de las actividades de programación.

METODOLOGÍA

Para cumplir con los objetivos se utilizó la herramienta del Análisis de Tareas (Otálora, 2007), desarrollada en psicología para examinar en profundidad la estructura de una situación problema, su demanda cognitiva y los niveles de aprendizaje de los sujetos en desempeños reales. El análisis de tareas se llevó a cabo con base en dos actividades puntuales ya empleadas por los docentes de primaria en el INSA, denominadas “Mi acuario y yo”, y “Ciclo de Vida”. Se escogieron estas actividades debido a dos razones; primero, son actividades de introducción al manejo del entorno de programación en grado 3º de primaria; y segundo, son actividades de corta duración acorde con los tiempos destinados en el proyecto. Dado que son actividades semejantes en cuanto al manejo del entorno, todos los análisis se harán de forma global, y cuando sea necesario especificar la tarea se hará de forma explícita.

El análisis de tareas se realizó en dos niveles: nivel objetivo y nivel subjetivo. La meta del primer análisis es hacer una descripción profunda del problema presentado y su estructura. Esto incluye la descripción de los objetivos, restricciones y características estructurales de la tarea. El segundo nivel, el subjetivo, se dividirá en cuatro partes, el análisis de la demanda cognitiva, la descripción de un desempeño ideal, la descripción de un desempeño real y el análisis del desempeño real. Para llevar a cabo el análisis subjetivo se realizó un proceso de recolección de datos que se describe a continuación.

Participantes

Se observó el desempeño en tiempo real de un niño de género masculino de 8 años de edad que cursa grado 3º de primaria en el colegio INSA, para lo cual se contó con el consentimiento informado de ambos padres, de los docentes de la institución, y del mismo niño, que en adelante le llamaremos J. La selección del participante se realizó a partir de una muestra de niños que el docente a cargo de la materia de informática consideraba tenían un desempeño promedio. A partir de esta muestra el participante final se eligió de forma aleatoria.

Adicionalmente, se tomaron datos de registro de avance de otros tres niños del mismo curso. El registro se tomó en dos momentos en cada una de las sesiones de la actividad, al inicio y al final. Esto se realizó con el objetivo de capturar el proceso de solución de problemas de forma secuencial y sin necesidad de observar todo el proceso.

Actividad y Cronograma de Registro

Como se mencionó antes, las actividades elegidas durante las cuales hacer el seguimiento fueron “Mi acuario y yo” y “Ciclo de vida”. Ambas actividades tienen una duración de tres semanas, con un total de nueve sesiones, de hora y media cada una. Debido a problemas de disponibilidad de equipos y personal de registro se decidió hacer un muestreo de las sesiones y tiempos de observación. Se registraron un total de 6 actividades en las siguientes fechas del 2012: 13 de Abril, 17 de Abril, 19 de Abril, 24 de Abril, 26 de Abril y 4 de Mayo. En cada sesión se registraron aproximadamente sólo los primeros 30 minutos de la actividad. Esto genera un total aproximado de 180 minutos de grabación.

Respecto a los registros de avance se tomaron un total de seis registros de inicio y seis registros de final por cada uno de los tres participantes extras, para un total de 36 registros.

Procedimiento e Instrumentos

Se llevó a cabo un registro de observación naturalista con el niño seleccionado. El registro se realizó con una cámara de video Sony con un miniCD de grabación. El tiempo máximo de grabación del CD es de 30 minutos. La cámara se montó sobre un trípode enfocando la pantalla del computador en la cual se observan los desempeños del niño en tiempo real. En algunas ocasiones la cámara enfocaba las instrucciones del profesor y las interacciones que el niño tenía con sus compañeros.

ANÁLISIS DE TAREAS

Análisis Objetivo

***Definición del problema**:*

1) Meta del problema: Mi acuario y yo: crear en SCRATCH 1.4 una animación de un acuario con al menos cuatro peces con diferentes disfraces. Ciclo de vida: Construir con ayuda de Scratch una presentación donde se muestre por medio de una historieta animada los cambios físicos que sufren algunos seres vivos hasta llegar a una edad adulta.

2) Restricciones de la tarea: para el caso de "Mi acuario y yo" se plantea explícitamente la restricción de un mínimo de 4 objetos en movimiento. Se deben usar las siguientes funciones: mover al presionar, esperar, rebotar si toca el borde, ir a, esconder, mostrar, cambiar disfraz. Deben usarse estructuras de control iterativas (repetición).

Elementos estructurales:

1) Barra de herramientas: se explora *archivo*, en donde se controlan las funciones básicas de los proyectos scratch (nuevo, abrir, guardar, importar, exportar, notas, abandonar).

2) Paleta de bloques: se definen el orden del movimiento de los objetos, su apariencia y la estructura de control. A continuación se presenta la estructura de cada bloque.

Movimiento: 1. Distancia, parametrizada de dos formas (cantidad de pasos o coordenadas cartesianas), 2. Giro, parametrizado en grados y orientación con flecha (izquierda, derecha), 3. Orientación del movimiento, parametrizada de dos formas (marco de referencia egocéntrico o marco de referencia alocéntrico), 4. Velocidad, parametrizada en coordenadas por segundos, 5. Movimientos especiales (rebote, fijar posiciones en cada uno de los ejes x o y de forma separada).

Control: 1. Inicio de actividad mediante bandera verde, o presionar tecla, o presionar objeto (la actividad está definida por cualquiera de los otros bloques), 2. Tiempo de espera parametrizado en segundos, 3. Iteración, parametrizado en número de ciclos o por siempre, 4. Comunicación entre objetos (mediante mensajes y con opción de definir los receptores), 5. Condicionales parametrizados mediante los siguientes operadores: si, si no, hasta, 6. Detener.

Apariencia: 1. Control cambio de disfraz, que se puede hacer de dos formas diferentes (número de disfraz o siguiente en la lista), 2. Visualizar estados intencionales (pensamientos o palabras), 3. Efectos visuales (de una lista con 7 opciones), 4. Cambio

de tamaño, parametrizado de dos formas (unidades de incremento o porcentaje de cambio), 5. Mostrar/ocultar objeto (se puede mostrar de forma especial llevándolo al frente).

3) Lista y edición de objetos: la lista de objetos se encuentra dividida en dos partes, el escenario o fondo utilizado y los objetos particulares. Los objetos se pueden duplicar o borrar haciendo clic con el botón secundario del ratón. A su vez es posible definir los cambios de apariencia del escenario. Scratch contiene tres formas de generar un nuevo objeto: pintar uno nuevo, abrir uno del archivo de forma controlada o abrir uno de forma aleatoria.

Abrir de archivo: scratch tiene almacenado un total de 308 figuras divididas en seis categorías: animales (con 75 objetos), dibujos de fantasía (con 48 objetos), letras (con 10 tipos), personas (con 104 objetos, en diferentes posiciones y vestidos), cosas (con 49 objetos) y vehículos (con 22 objetos). Muchos de los objetos incluyen programas.

Objeto sorpresa: abre un objeto al azar de los 308 disponibles.

Pintar un nuevo objeto: con el editor de pinturas es posible crear un nuevo objeto o modificar uno preexistente. El ambiente es similar al programa Paint de Microsoft, con una barra de herramientas, paleta de colores, herramientas de cambio de tamaño y giro de los objetos (zoom, expansión, reducción), opciones de área a dibujar (borde o relleno), herramientas de control (importar, exportar y borrar) y un lienzo en donde se observa el dibujo creado.

4) Editor de programas, disfraces y sonidos: en la parte central de la interface se puede llevar a cabo el proceso de programación. En específico la tarea explora el editor de programas y el editor de disfraces. En la parte superior de los editores aparece el objeto seleccionado con tres informaciones básicas: nombre, ubicación (en coordenadas) y orientación. Se incluye una función para especificar la orientación del objeto mediante un vector que puede ser manipulado con el ratón.

Editor de programas: 1. Un conjunto amplio de comandos predefinidos (alrededor de 100 en total), con ranuras en la parte superior e inferior, que señalan los lugares en donde se puede ensamblar un nuevo comando (excepto comandos de inicio y detención en el bloque control). 2. La pila de comandos de un nuevo programa se elabora mediante dos procesos: a) arrastre de un comando de la paleta de bloques al editor de programas, y b) el ensamblaje de un nuevo comando debajo del anterior, 3. La modificación de un programa ya existente se realiza de tres maneras: a) se puede incrustar un nuevo comando en cualquier sitio en donde haya una ranura, sin importar el orden, b) para borrar un comando basta señalarlo y arrastrarlo al área de paleta de bloques, c) para inactivar un comando es necesario arrastrarlo y ubicarlo aparte del programa principal. 4. La parametrización de los comandos (que funcionan como

variables) se realiza modificando los valores de las casillas en blanco de cada comando.

Editor de disfraces: en este editor se da la posibilidad de dar múltiples apariencias a un mismo objeto. Cuando se selecciona la pestaña “disfraces” aparece el listado de los disfraces de cada objeto. Los disfraces pueden crearse de 4 formas: 1. Pintar, que abre el editor de pinturas, 2. Importar, que permite seleccionar un objeto del disco duro, 3. Cámara, que permite tomar una foto, 4. Arrastrar imágenes de la web. Al igual que los objetos existe la posibilidad de duplicar cada disfraz. Es posible asignar un nombre a cada disfraz.

5) Escenario: en la parte superior derecha de la interface aparece un recuadro en donde se observa el resultado de la animación, con tres formas de visualización: pantalla completa, tamaño medio o tamaño pequeño. De forma adicional se puede controlar el inicio (bandera verde) que activa el movimiento de todos los objetos en paralelo y la detención del programa (botón rojo).

Análisis Subjetivo

Demanda cognitiva: a continuación se detallan las habilidades y conocimientos relacionados con pensamiento computacional involucrados en la solución óptima de los problemas “mi acuario y yo” y “ciclo de vida”. Por “óptimo” hemos tratado de reflejar una forma de solución ideal que no sea la más básica ni tampoco la más avanzada posible, sino una forma de solución de nivel intermedio.

1. Habilidades Computacionales

a) Traducción del espacio real a un espacio virtual mediante instrumentos computacionales: diferentes actividades simbólicas que se realizan sobre espacios reales, como la escritura y el dibujo, se deben traducir a un espacio virtual, mediante el uso de los comandos y las herramientas apropiadas.

b) Programación:

Procesos de Computación

- Uso de variables y asignación de valores (parametrización): en el bloque Control la variables “esperar” se parametriza en segundos y “repetir” en ciclos. En el bloque Movimiento la variable mover se parametriza en pasos, “girar” en grados, y las variables relativas a localización se parametrizan en ejes de coordenadas.
- Paralelismo: dado que se requiere coordinar el movimiento de al menos cuatro peces en el acuario, los algoritmos deben controlar estos movimientos en paralelo.

- Insertar comandos en una estructura de datos: ubicar un comando faltante o adicional en una secuencia de datos ya creada mediante la operación de arrastre de la paleta de bloques al editor de programas (dos métodos posibles: insertar primero y luego recomponer elementos ó primero recomponer y luego insertar).

Sintaxis de la programación

- Manejo de palabras clave (keywords):
  - Y: de forma implícita como *secuenciador* (no como conjunción booleana)
  - Si no (else): de forma explícita como parte de las estructuras de control
  - Entonces: de forma explícita como *consecuente* en las estructuras de control. De forma implícita como *secuenciador* en el orden vertical seguido.
- Manejo de estructuras de control
  - Uso de condicionales: el inicio del movimiento de los objetos se condiciona a un input específico (presionar tecla o bandera verde).
  - Uso de iteraciones: para continuar el movimiento de los peces de forma continua –y sin editar largas cadenas de instrucciones- se deben usar iteraciones (repetir, por siempre si).

c) Elementos de modelado: un componente fundamental del pensamiento computacional es la habilidad de modelar la realidad mediante el uso de los computadores. Aunque en un nivel muy básico, esto se ve demandado especialmente en la activad “el ciclo de vida”, en donde los niños deben simular de forma dinámica los diferentes estadios por los que pasa un organismo. En específico, la animación muestra el efecto del paso del tiempo sobre la anatomía del organismo. Esta habilidad está estrechamente relacionada con el desarrollo del pensamiento científico, dado que el diseño de experimentos implica también modelar un aspecto de la realidad y pensarla en términos de variables que interactúan causalmente.

d) Automatización: la programación del movimiento de diferentes objetos en las actividades requiere la utilización de soluciones similares, es decir, la utilización de bloques en secuencias parecidas. La automatización implica además determinar en qué tipo de situaciones es válido generalizar las soluciones previamente construidas y en cuáles no. Además de calibrar la generalización, también es importante determinar con cuidado los valores de los parámetros de las variables utilizadas, pues si bien puede haber una semejanza estructural, el contenido puede variar considerablemente.

2. Conocimiento conceptual

a) Conocimiento biológico: las actividades de “mi acuario y yo” y “el ciclo de vida” involucran el uso y la comprensión de algunas relaciones biológicas, tales como *ecosistema* y *ciclo vital*.

b) Conocimiento geométrico: la meta final de un movimiento ordenado en un espacio bidimensional es redescrito primero en términos matemáticos y algorítmicos, haciendo explícitos los tres componentes básicos de la geometría Euclidiana: la distancia, el ángulo y la orientación. Se matematiza en la medida en que se utilizan patrones de medición, tales como pasos y ángulos. Hacen parte de un algoritmo cuando cada una de las propiedades espaciales se representa en térmicos de variables y parámetros.

3. Planificación cognitiva

La habilidad de planificar se demanda en dos momentos de la actividad. Primero, en la construcción del proyecto por escrito, en donde se delimita la estructura del problema, la meta, las restricciones y los pasos para alcanzarla. Segundo, en la elaboración de la sintaxis (construcción de algoritmos), los comandos deben estar organizados de tal manera que los bloques generen la consecuencia deseada, para esto es necesario especificar con antelación esa organización. Como especifican algunos autores, la organización de un programa no sigue necesariamente la organización del lenguaje natural, lo que constituye una fuente importante de dificultad para los aprendices. Ver Anexo 1 para un resumen de la demanda cognitiva.

Descripción del desempeño real

Descripción de cada una de las actividades por fecha

Si bien se tomó registro de 6 días, debido a problemas técnicos de grabación no fue posible analizar el video de la primera sesión del 13 de Abril de 2012.

Fecha	Actividad	Descripción
17 de Abril-2012	Mi acuario y yo	La actividad se centra en la edición del fondo del acuario. El maestro elabora un ejemplo de un fondo con ayuda de algunos estudiantes. Explica cómo usar el editor de pinturas de SCRATCH y corrige errores de su manejo. Sólo se usa el tablero interactivo.
19 de Abril-2012	Mi acuario y yo	Los estudiantes comienzan a diseñar el fondo de su propio acuario, de acuerdo con lo que habían planeado inicialmente en su proyecto. Luego comienzan a diseñar los objetos que

		tendrán que animar, empiezan por los peces.
19 de Abril-2012	El ciclo de vida	Abren el avance del proyecto, con el fondo y los objetos principales ya elaborados. La primera consigna es dibujar una carita de acuerdo a como se estén sintiendo en ese momento. La carita debe moverse y mostrar una expresión. La segunda consigna es dar movimiento a cada uno de los objetos, en el caso de J empieza por la mariposa. Los movimientos de los objetos se deben coordinar de tal manera que muestra la idea de un ciclo.
24 de Abril-2012	Mi acuario y yo	Los niños continúan con la edición de los objetos. J construye un total de 5 objetos.
26 de Abril-2012	Mi acuario y yo	Continúa con la edición de los objetos. Se centra en la decoración. Luego crea varios disfraces de algunos objetos para crear la ilusión de movimiento. Al final comienza a escribir los programas que darán movimiento a todos los objetos
26 de Abril-2012	El ciclo de vida	Continúa construyendo los bloques de movimiento para los objetos. Terminan de decorar los objetos. J se esfuerza por coordinar la secuencia de bloques de forma correcta.
4 de mayo-2012	El ciclo de vida	Terminan de construir los bloques de movimiento y terminan dibujando una carita expresando cómo se siente cada uno. J. no logra finalmente dar el movimiento correcto a los objetos.

Nivel de exploración del ambiente del problema de cada una de las actividades por fecha

Fecha	Actividad	Exploración Ambiente
17 de Abril-2012	Mi acuario y yo	(3) Lista y edición de objetos: pintar un nuevo objeto
19 de Abril-2012	Mi acuario y yo	(1) Barra de herramientas (2) Paleta de bloques: movimiento (distancia), control (inicio, espera, iteración), apariencia

		(cambio disfraz). (3) Lista y edición de objetos: pintar un nuevo objeto. (4) Editor de guiones y disfraces: editor de guiones (todos)
19 de Abril-2012	El ciclo de vida	(1) Barra de herramientas (2) Paleta de bloques: movimiento (distancia), control (inicio, espera, iteración), apariencia (cambio disfraz, visualizar estados). (3) Lista y edición de objetos (4) Editor de guiones y disfraces: editor de guiones (todos), editor de disfraces (pintar).
24 de Abril-2012	Mi acuario y yo	(1) Barra de herramientas (3) Lista y edición de objetos
26 de Abril-2012	Mi acuario y yo	(1) Barra de herramientas (2) Paleta de bloques: movimiento (distancia), control (inicio, espera, iteración), apariencia (cambio disfraz). (4) Editor de guiones y disfraces: editor de guiones (todos), editor de disfraces (pintar)
26 de Abril-2012	El ciclo de vida	(1) Barra de herramientas (2) Paleta de bloques: movimiento (distancia, giro), control (inicio, espera, iteración), apariencia (cambio disfraz). (4) Editor de guiones y disfraces: editor de guiones (todos), editor de disfraces (pintar)
4 de mayo-2012	El ciclo de vida	(1) Barra de herramientas (2) Paleta de bloques: movimiento (distancia), control (inicio, espera, iteración), apariencia (cambio disfraz, visualizar estados). (4) Editor de guiones y disfraces: editor de guiones (todos), editor de disfraces (pintar)

El análisis del nivel de exploración del ambiente del problema (análisis objetivo) muestra una exploración de aproximadamente el 50% (18 de 35) entre ambas actividades en cinco sesiones. Esto muestra que actividades sencillas realizadas con niños pequeños permiten un amplio manejo de las potencialidades del programa.

Análisis del desempeño real

1. Traducción del espacio real a un espacio virtual mediante instrumentos computacionales: ninguno de los niños observados mostró dificultad para realizar esta acción en cuanto al manejo del editor de pinturas ni en la comprensión del escenario. Es notorio que en clases pasadas han logrado un buen manejo de estos recursos. Sin embargo, fue notoria la dificultad de J en la actividad de ciclo de vida para plasmar los giros de los objetos, es decir la cantidad de giro necesario para producir el efecto deseado. Tras varios intentos no lo logra adecuadamente.

2. Procesos de computación: los niños observados muestran una comprensión adecuada del manejo de las variables y su parametrización, en especial hacen un uso extenso de “mover”, “cambiar disfraz” y “esperar”. En todos estos casos, se observa un uso continuo en diferentes organizaciones y parámetros, por ejemplo, en momentos diferentes del *ciclo de la vida* a la variable “esperar” se le asignan diferentes valores. De igual forma la variable “cambiar disfraz” se emplea continuamente para producir efectos de movimiento aparente. Por otro lado, la habilidad de paralelismo en los procesos de cómputo, si bien parece comprenderse, es de más difícil ejecución. En específico, la coordinación de las velocidades planteó dificultades interesantes, que exigían una continua retroalimentación mediante la ejecución del programa para ubicar los problemas de programación.

3. Sintaxis de la programación: Es clara para el niño la necesidad de uso de estructuras de control que regulen la dinámica de los objetos. En especial la necesidad de usar condicionales y la iteración. Sin embargo, se observa que la comprensión de lo que implica la iteración no es completa, más exactamente en qué situaciones se debe usar, y se dificulta por el uso del término “por siempre”. Este problema fue bastante generalizado en todas las observaciones hechas.

Es clara también la importancia del uso de las palabras clave. Se hace un uso extenso del conector “y” de forma implícita en la secuencia de los bloques, y del “entonces” de forma implícita como resultado de oprimir ciertos comandos. Si bien para el niño es evidente la necesidad de su uso, no es claro si comprende plenamente el significado de estos términos en el dominio de la programación.

4. Automatización: en varias oportunidades los niños aplicaron los mismos procedimientos para solucionar problemas semejantes. Por ejemplo, aplicaban de forma continua el operador “por siempre” cuando debían iterar una secuencia de acciones, o utilizaban “cambiar disfraz” cuando necesitaban producir ilusiones de movimiento. No obstante, se observaron dificultades en cuanto a la generalización de estos procedimientos en situaciones incorrectas. Así, por ejemplo, cuando a J se le pidió poner en movimiento un objeto (carita) inmediatamente ensambló un operador de inicio con uno de repetir por siempre. Procedimiento que había aplicado en una sesión anterior de forma repetida y que lo llevó a una solución errónea.

5. Conocimiento conceptual: los niños manejan correctamente los conceptos involucrados con biología y con geometría. Se hace un uso amplio de la “distancia”,

pero no es claro en las grabaciones la forma como el niño comprende las unidades de medición utilizadas y cómo las relaciona con las medidas típicas.

6. Planificación cognitiva: uno de los pocos problemas observados estuvo relacionada con la forma en que los niños debían reorganizar los bloques para generar un movimiento concreto, que previamente no se había logrado con la organización inicial. No obstante, los niños mostraron dos estrategias de solución de este problema.

a) Proceso combinatorio: los niños generan diferentes combinaciones y corren cada una de ellas hasta hallar la correcta. No es un proceso aleatorio en la medida en que mantiene ciertas restricciones. Sabe por ejemplo, que ciertos comandos deben estar dentro de la estructura de control “por siempre”, o que mover debe estar antes que cambio de disfraz, etc.

b) Submetas: se establece la meta general y las submetas necesarias para alcanzarla, cada una de estas a su vez se puede descomponer en otras submetas, hasta definir el conjunto específico de comandos a utilizar. Por ejemplo, (definir) (vincular con algoritmos). En este caso, fue fundamental el rol del maestro como mediador de esta estrategia, en la medida en que era él quien inicialmente proponía formas de razonamiento que motivaban la utilización de esta estrategia. Por ejemplo, con frases como “¿ahora qué necesitas?”, o “¿estás seguro que esa es la mejor forma?, o “¿crees que realmente necesitas colocar eso?”

DISCUSIÓN

Los resultados del análisis de tareas muestran en detalle la forma como el uso del entorno gráfico de programación SCRATCH, junto con las actividades educativas propuestas en el aula, promueven el desarrollo del pensamiento computacional, la adquisición de conocimiento conceptual académico y habilidades de planificación cognitiva. El programa ofrece un claro soporte para algunos de los elementos que en la literatura sobre aprendizaje de la programación se han señalado como los más problemáticos para los aprendices (Pane, 2002), tales como el uso de iteraciones y de condicionales. Ambas referidas al manejo de estructuras de control de acciones. Estas funciones en específico se facilitan mediante el uso de bloques prediseñados que sirven de marco para construir un programa correctamente.

Sin embargo, en este punto es importante preguntarse si la comprensión que aparentemente se genera del manejo de las estructuras de control en SCRATCH puede transferirse a otro tipo de situaciones más cercanas a los lenguajes de programación reales. Por ejemplo, en la serie de investigaciones sobre Commonsense Computing las tareas que se utilizaban para observar las habilidades de programación eran problemas cotidianos muy sencillos, tales como ordenar un conjunto de objetos o números. En estas situaciones era necesario escribir la solución de una forma coherente, esto se

hacía así en la medida en que se asemejaba a la situación real de desempeño de un programador experto. Como se señaló en la introducción, los adultos mostraban tener ciertos conocimientos de programación, aunque fallaban en el uso de varios componentes fundamentales de la programación, como el uso de estructuras de control (ej. Sólo el 25% uso condicionales). Los resultados de investigaciones realizadas con niños de primaria fueron similares. Por lo tanto, es necesario seguir investigando para determinar hasta qué punto el uso de un entorno de programación como SCRATCH realmente tiene impacto en las habilidades de programación a este nivel.

En relación con el tópico específico del aprendizaje de programación es interesante notar que de acuerdo con las observaciones hechas en el aula uno de los elementos de más difícil comprensión fuel el paralelismo de la programación, aspecto que ya había sido reportado en investigaciones previas. Sin embargo, es interesante que en un contexto como una clase sobre programación el paralelismo surja como un obstáculo, pues esto crea las condiciones necesarias para mejorar su aplicación a este dominio. Como Wing señala en varias oportunidades, el pensamiento computacional hace uso de habilidades que se encuentran ya inscritas en nuestra vida cotidiana, como pensar en la ejecución de procesos que ocurren de forma simultánea, y por esta razón el aprendizaje del pensamiento computacional se debería alimentar de este tipo de recursos.

A pesar de las preguntas que quedan abiertas sobre las habilidades de programación, la investigación actual muestra que el uso de SCRATCH puede tener impacto en campos de conocimiento que si bien se circunscriben al pensamiento computacional, no se limitan al aprendizaje de la programación. Así, es importante destacar la contribución que hace el uso del entorno de programación para aprender cómo modelar la realidad en términos de variables que interactúan y así promover un pensamiento más abstracto. Al respecto, es importante pensar en nuevas actividades de aula que tengan como meta principal el aprendizaje de esta habilidad, actividades que sigan el esquema de las actividades integrativas en donde el conocimiento académico sirve de base para fomentar esta forma de pensamiento.

Por último, el uso del entorno SCRATCH parece tener una importante influencia en el desarrollo de la planificación, que si bien no es un elemento constituyente o esencial del constructo “pensamiento computacional”, sí parece ser un proceso cognitivo fuertemente requerido para su implementación eficaz. Algunos de los errores de programación observados durante las grabaciones muestran que gran parte del tiempo que los niños gastan programando, lo invierten en la planificación del orden correcto de las instrucciones y corrigiendo errores derivados de ésta. Una hipótesis interesante para explorar a futuro relacionada con este proceso cognitivo es que puede existir una relación entre la planificación de las acciones y las habilidades de programación (construcción de las pilas de instrucciones). Es probable que cuando los niños aclaran la meta a cumplir y los medios, esto tiene un impacto en la calidad de la secuencia de acciones programadas. En este sentido, tanto las actividades previas de

planificación de la actividad general, como el soporte que brinda el profesor en clase para aclarar las metas, restricciones y requerimientos, serían fundamentales para promover el aprendizaje de las habilidades de programación. No obstante, nuevamente sería necesario llegar a cabo investigaciones adicionales para determinar con más precisión la fuerza de esta relación, y la forma como las actividades de aula podrían orientarse a promover con más fuerza el desarrollo de ambas habilidades al mismo tiempo.

REFERENCIAS

Cooper, S., Pérez, L., & Rainey, D. (2010). Communications of the ACM, 53 (11), 27-29.

Fletcher, G., & Lu, J. (2009). Communications of the ACM, 52 (2), 23-25.

Guzdial, M. (2008). Paving the way for computational thinking. Communications of the ACM, 51, (8), 25-27.

Karmiloff-Smith, A. (1994). Más allá de la modularidad. Alianza, Barcelona.

Lewadowski, G., Bouvier, D., McCartney, R., Sanders, K., & Simon, B. (2007). Commonsense computing (episode 3): Concurrency and concert tickets, In

Otálora, Y. (2007). El análisis de tareas como una herramienta de estudio del funcionamiento cognitivo encubierto. Manuscrito no publicado.

Pane, J., Ratanamahatana, C., & Myers, B. (2001). Studying the language and structure in non-programmers' solutions to programming problems, 54, 237-264.

Proceedings of theThird International Workshop on Computing Education Research (2007), 133–144.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmont, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Communications of the ACM, 52 (11), 60-67.

Simon, B., Chen, T., Lewadowski, G., McCartney, R., & Sanders, K. (2006). Commonsense computing: What students know before we teach (Episode 1: Sorting), In *Proceedings of theThird International Workshop on Computing Education Research* (2006), 133–144.

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

Wing, J. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society, 366, 3717-3725.

Documento publicado en EDUTEKA
http://www.eduteka.org/investigacion_Scratch.php
Diciembre 01, 2013

ANEXO

DEMANDA COGNITIVA	
Habilidades Computacionales	Trabajo sobre espacios virtuales: utilizar herramientas computacionales para llevar a cabo diferentes actividades simbólicas, como escritura y dibujo.
	Programación: especificado abajo
	Elementos de modelado: representación de la realidad en diferentes niveles de abstracción, como estructuras de datos, como variables que interactúan causalmente, como funciones matemáticas, etc.
	Automatización: hallar la solución más eficiente y determinar cuándo es permitido aplicarle en otros contextos y cuando no.
Conocimiento Conceptual	Conocimiento biológico: el uso correcto del programa requiere la comprensión de conceptos biológicos, como ecosistema, ciclo de vida, adaptación, etc.
	Conocimiento geométrico: el uso de representaciones espaciales Euclidianas, en términos de distancias, ángulos y orientación. Uso de sistemas de medición.
Planificación Cognitiva	Definir metas a largo, mediano y corto plazo, coordinarlas mediante las acciones apropiadas en un ambiente con restricciones especificadas.

DEMANDA COGNITIVA DE PROGRAMACIÓN	
Procesos de computación	Uso de variables y asignación de valores: representar un problema en términos de variables como distancia y tiempo, cada una parametrizable.
	Paralelismo: es necesario coordinar diferentes movimientos que ocurren simultáneamente, y frecuentemente a diferentes velocidades.
	Manipulación de estructuras de datos: detección y corrección de errores de programación, cómo insertar o eliminar nueva información en estructuras ya creadas de forma correcta.
Sintaxis de la programación	Manejo de palabras clave (keywords): uso de conectores lógicos, como conjunción, disyunción, entonces.
	Manejo de estructuras de control: uso de condicionales para dar coherencia lógica al programa, y uso de iteraciones para hacer más eficiente la solución.