

UNIVERSIDAD NACIONAL DE COLOMBIA
DEPARTAMENTO DE PSICOLOGÍA
MAESTRÍA EN PSICOLOGÍA
LÍNEA DE INVESTIGACIÓN: COGNICIÓN, EDUCACIÓN Y MEDIOS

COMPRENDIENDO EL PENSAMIENTO COMPUTACIONAL: EXPERIENCIAS DE
PROGRAMACIÓN A TRAVÉS DE SCRATCH EN COLEGIOS PÚBLICOS DE
BOGOTÁ.

Tesis para optar al título de Magister en Psicología

Autor: Luis Alfredo Sánchez Ruiz
Dirigida por: Javier Alejandro Corredor Aristizábal, Ph.D.
Bogotá, Mayo de 2016

Resumen

El pensamiento computacional es una habilidad esencial para la resolución de problemas en espacios diversos que van desde el uso de computadores hasta otros contextos de la cotidianidad. Para dar cuenta de esto, se han desarrollado múltiples programas educativos que involucran tareas de programación. Sin embargo, no existe evidencia de que la participación sostenida en dichos procesos tenga efectos en el pensamiento computacional, en parte por la ausencia de una herramienta adecuada para medirlos. El objetivo de esta investigación es indagar los efectos que tiene Scratch, una herramienta de programación, en el desarrollo del pensamiento computacional y su *transfer* a una condición isomorfa. En este estudio se recopiló la información de 106 estudiantes de dos instituciones educativas públicas de Bogotá, en donde los alumnos de una institución ya habían interactuado con Scratch y los de la otra no. Los datos obtenidos se analizaron de manera cuantitativa. Los resultados sugieren que Scratch tiene un efecto significativo en el desarrollo de conceptos asociados al pensamiento computacional, que estos se agrupan de acuerdo a su dificultad y que la prueba isomorfa en LEGO funciona como una medida de *transfer*. Se concluye que el uso sostenido de Scratch fomenta la adquisición de los conceptos computacionales y que el pensamiento computacional se puede transferir a dominios análogos de aprendizaje. *Palabras Clave:* Pensamiento Computacional, Conceptos Computacionales, Transfer, Aprendizaje de Programación, LEGO, Scratch.

Abstract

Computational thinking is an essential skill for solving problems in diverse tasks including the use of computers and other contexts of everyday life. That is why many educational institutions have incorporated education programs that include programming as a basic element. However, there is little evidence that participation in such programs has a positive impact on computational thinking. The objective of this study is to investigate the effects by Scratch, a programming tool, in the development of computational concepts related to computational thinking and their transfer to an isomorphic condition. This study evaluated 106 students from two public educational institutions Bogotá. Students from one institution had already interacted with Scratch and students from the other institution were not. Data were analyzed quantitatively. The results suggest that Scratch has a significant effect on the development of computational concepts, that these concepts are grouped according to their difficulty, and that the isomorphic test with LEGO works as a transfer measure for computational thinking. It is concluded that sustained use of Scratch promotes the acquisition of computational concepts that can be transferred to similar domains of learning.

Keywords: Computational Thinking, Computational Concepts, Transfer, Learning Programming, LEGO, Scratch.

TABLA DE CONTENIDOS

Lista de Tablas	
Lista de Figuras	
Lista de Anexos	
Introducción.....	6
Marco	
Teórico.....	9
Nuevas Tecnologías y Nuevas Prácticas Digitales.....	9
Pensamiento Computacional.....	12
Dimensiones del Pensamiento Computacional.....	13
El Surgimiento del Pensamiento Computacional a través de la Programación: el Caso de LOGO.....	16
Programas Actuales para el Desarrollo del Pensamiento Computacional.....	18
Scratch.....	19
El Rol de Scratch en el Aprendizaje del Pensamiento Computacional.....	20
Aprendizaje en Scratch: Construyendo Usuarios Productores de Programas	22
Evaluación del Pensamiento Computacional y Scratch.....	24
Transfer y Aprendizaje.....	26
Método.....	32
Participantes.....	32
Instrumentos.....	32
Procedimiento.....	33
Codificación de la Información.....	35
Resultados.....	37
Análisis de Confiabilidad.....	37
Análisis Factorial.....	38
Diferencias Entre el Grupo Experimental y el Grupo Control en Desempeño.....	40
Encuesta Sobre Uso y Acceso a Tecnologías y Conocimiento de Programas.....	47
Análisis de Correlación Entre la prueba de Scratch y la de LEGO.....	53
Discusión.....	55
Referencias.....	62
Anexos	

Lista de Tablas

Tabla 1. Rubrica de calificación	35
Tabla 2. Factores agrupados para la prueba de Scratch	38
Tabla 3. Factores agrupados para la prueba de LEGO.	39
Tabla 4. Análisis de Mann-Whitney para la media total de la aplicación: Prueba en Scratch y en LEGO	41
Tabla 5. Análisis de Mann-Whitney para la media total de la prueba de Scratch y LEGO por separado	41
Tabla 6. Análisis de Mann-Whitney para los factores de la prueba de Scratch y LEGO	42
Tabla 7. Rangos promedio de los factores de las pruebas de Scratch y LEGO	43
Tabla 8. Agrupación de variables de la encuesta de uso, acceso y conocimiento de programas	47
Tabla 9. Análisis de Mann-Whitney para las variables agrupadas de la encuesta	48
Tabla 10. Análisis mediante el modelo lineal univariante entre el total general de ambas pruebas y las covariables	51
Tabla 11. Análisis mediante el modelo lineal univariante entre el total de la prueba de Scratch y las covariables	52
Tabla 12. Análisis mediante el modelo lineal univariante entre el total de la prueba en LEGO y las covariables	53
Tabla 13. Correlación de Spearman entre la prueba de Scratch y la prueba de LEGO	54

Lista de Figuras

Figura 1. Grupo experimental vs grupo control –Factor 1 Scratch	44
Figura 2. Grupo experimental vs grupo control –Factor 2 Scratch	45
Figura 3. Grupo experimental vs grupo control –Factor 3 Scratch	45
Figura 4. Grupo experimental vs grupo control –Factor 1 LEGO	46
Figura 5. Grupo experimental vs grupo control –Factor 2 LEGO	46
Figura 6. Grupo experimental vs grupo control –Acceso a tecnologías	49
Figura 7. Grupo experimental vs grupo control –Uso dado a las tecnologías.	50
Figura 8. Grupo experimental vs grupo control –Programas que conocen	50

Lista de Anexos

ANEXO 1. Prueba en Scratch TPC	67
ANEXO 2. Prueba en LEGO TPC	69
ANEXO 3. Formato de equivalencias en LEGO (Versión 1)	71
ANEXO 4. Encuesta sobre uso, acceso de tecnologías y conocimiento sobre programas académicos.	74
ANEXO 5. Consentimiento informado y asentimiento informado	76
ANEXO 6. Plantilla de ejercicios en Scratch	79
ANEXO 7. Plantilla de ejercicios en LEGO	82
ANEXO 8. Ejercicio de la prueba en LEGO	85

Introducción

El pensamiento computacional (PC) es un término acuñado por Jeannette Wing para describir un conjunto de habilidades de pensamiento, hábitos y enfoques integrales para la resolución de problemas relativos a la programación que no solo se limitan al uso de un ordenador (Lee et al, 2011). Más específicamente el PC integra aspectos como el pensamiento algorítmico, matemático y de diseño, que se utilizan para la construcción de secuencias y estructuras de objetivos que subyacen a las tareas de programación (Lee et. al, 2011). De cierta manera, se puede entender el PC como un conjunto de procesos de pensamiento que se relacionan con la creación y modificación de programas informáticos para la resolución de diversas tareas a través de un computador. El PC permite a la persona que lo posee ser capaz de tener un rol activo en la interacción con las nuevas tecnologías (Brennan y Resnick, 2012). A pesar de la importancia del pensamiento computacional, la investigación internacional muestra que los conceptos de pensamiento computacional no son el centro de las prácticas educativas. Por el contrario, los colegios se concentran en el uso y consumo de los programas pero no su modificación o transformación (Wilson, Sudol, Stephenson y Stehlik, 2010).

En el contexto colombiano poco es lo que se sabe sobre la enseñanza del pensamiento computacional y sus conceptos. Por ejemplo, en los procesos para la alfabetización de hombres y mujeres en TICs desarrollados por el Ministerio de Tecnologías de la Información y las Comunicaciones (MinTIC, 2014) se mostró que el tipo de uso dado a las tecnologías por parte de los colombianos es en gran parte pasivo (Ipsos-Napoleón Franco, 2010 y 2012). Esta información se ve corroborada por el Departamento Administrativo Nacional de Estadística (DANE) cuyas estadísticas muestran que en el 2010, el 90% de la población entre 18 y 24 años solo usaba internet para comunicación y para un uso pasivo de

las redes sociales, mientras que para el año 2014, el 63.2% de las personas mayores de 5 años que usaron internet, lo usaron con este mismo objetivo (Corporación Colombia Digital, 2014). En la misma línea, las estadísticas muestran que de esa misma población solo un 36.7% reportó usar internet para educación y aprendizaje (DANE, 2015).

Estas estadísticas pueden relacionarse con la ausencia de procesos educativos sistemáticos que involucren el desarrollo del pensamiento computacional con un uso creativo y constructivo de las nuevas tecnologías. Por ejemplo, Patricia Jaramillo (2005) encontró en dos instituciones educativas de Bogotá que, a pesar de las estrategias por darle a las instituciones dotación en computadores y software educativo con el fin de promover las TICs, los avances en competencias tecnológicas de los estudiantes se encuentran por debajo de los estándares internacionales y el uso de las herramientas informáticas por parte de los docentes no es el adecuado. Estudios en el ámbito internacional muestran una realidad similar. Por ejemplo, diversas investigaciones muestran que la enseñanza en TICs en los Estados Unidos se concentra exclusivamente en aspectos de la computación basados en habilidades de consumo pasivo, como por ejemplo el uso del procesador de palabras y de internet, pero no se concentra en el desarrollo de aspectos conceptuales, como el desarrollo de pensamiento algorítmico, la programación, modelación y abstracción, entre otros (Wilson, Sudol, Stephenson, and Stehlik, 2010).

En este sentido y debido a la poca información que se tiene sobre las prácticas activas de los colombianos con las tecnologías y específicamente aquellas prácticas asociadas al fortalecimiento de los conceptos inherentes a la ciencia computacional, se pretende mediante esta investigación evaluar empíricamente los efectos del uso sostenido de una herramienta multimedia de programación llamada Scratch (cuyas características se explicarán más adelante) sobre el desarrollo de los *conceptos computacionales*. Por tal

razón, el objetivo principal de este estudio es evaluar si mediante el uso de Scratch los jóvenes desarrollan *conceptos* lógicos inherentes al pensamiento computacional.

Particularmente, se quiere evaluar de qué forma varia el desempeño de los estudiantes que han tenido contacto con Scratch en el desarrollo de ejercicios que den cuenta de esos conceptos, con respecto a estudiantes que no han usado dicho programa. De tal manera, también se podrán caracterizar aspectos generales del aprendizaje de estos conceptos computacionales. Además, se pretende determinar si esos conceptos adquiridos mediante Scratch se pueden transferir a un dominio análogo.

Marco Teórico

Para poder ahondar en las distintas aristas que exhibe el pensamiento computacional vale la pena entender su influencia y limitaciones. De esta manera y para comenzar, en este apartado se mostrará lo que sucede cuando los jóvenes se sumergen en una dinámica de formación como usuarios en la nueva realidad digital, la cual, hoy por hoy, ha trascendido las fronteras de los escenarios tradicionales. Con esto en mente, se demarcará un panorama en cuanto a las herramientas de programación y sus implicaciones cognitivas y educativas, mediante la explicación de una de las herramientas precursoras de la programación en computadoras: LOGO. A partir de ello y como contraste se presentará Scratch, la cual es una herramienta más actual. Se destacarán sus características y ventajas, enfatizando su papel en la evolución del pensamiento computacional y las dimensiones que se le atribuyen. Dimensiones que han contribuido en la explicación del surgimiento de usuarios empoderados en la creación y entendimiento de las tecnologías con las que interactúan. A su vez, se hará un contraste con otras plataformas de características muy semejantes a las de Scratch con el fin de hacer hincapié en el alcance de dichas herramientas y sus limitaciones o ventajas frente a lo que propone Scratch. Por último, se revisará de acuerdo a la literatura el rol que cumple Scratch en el aprendizaje del pensamiento computacional y la importancia de evaluar el *transfer* de los conocimientos computacionales a otros dominios.

Nuevas Tecnologías y Nuevas Prácticas Digitales

Los avances tecnológicos se han venido desarrollando de manera vertiginosa en la actualidad (Marín, 2010). Dichos progresos son tan contundentes que implican que las personas se vean avocadas a adaptarse a su uso sin preocuparse por el cómo funcionan. Esto tiene sentido ya que comúnmente se asume que hay que tener un conjunto de habilidades específicas de ciertas disciplinas como la ingeniería para poder comprender

cuál es su lógica implícita (Olabe, Basogain y Basogain, 2015). En otras palabras, las personas emplean los nuevos artefactos y los convierten en parte de su cotidianidad pero no alcanzan a entender bajo que principios lógicos fueron creados (Marín, 2010).

Ahora bien, se reconocen los entornos virtuales como estructuras las cuales están asociadas con un hardware y software, en donde en cuanto al primero se requiere de conocimientos en electrónica y matemáticas con el fin de desarrollar la parte física de un computador. Por otro lado, el desarrollo de software parece ser más accesible ya que el internet ha contribuido significativamente a que los usuarios estén inmersos en espacios que permiten la difusión de textos, gráficas, imágenes y videos, como por ejemplo YouTube, Flickr, Facebook y las paginas para creación de blogs entre otros (Lopez y de la Llata Gómez, 2010). Cabe agregar que el entendimiento del software esta más de la mano entre aquellas personas que nacieron en la era digital que quienes son migrantes digitales.

El término “nativos digitales” acotado por Marc Prensky en el 2001 (García, Portillo, Romo & Benito, 2007) hace referencia a aquellas personas que han crecido con las tecnologías y poseen una destreza consumada (Lopez y de la Llata Gómez, 2010). Por otro lado existen los migrantes digitales los cuales han tenido que adaptarse a la fuerza a esta tecnificación de la sociedad. El factor esencial de este tema se basa en que estos nativos digitales suelen ser en su mayoría usuarios no mayores de 30 años quienes satisfacen sus necesidades (académicas, de ocio, sociales, entre otras) mediante la tecnología, que, en muchos casos están ancladas a los nuevos dispositivos y los aplicativos que se han desarrollado para los mismos (Marín, 2010). Sin embargo, la interacción con dicha tecnología tiene un carácter pasivo, por lo tanto no es una generalidad que en estos individuos resida la intención en la creación y comprensión del funcionamiento de los contenidos que ellos mismos consumen.

Como se mencionó anteriormente los nativos digitales parecen encajar en un tipo de usuario que no necesariamente está enterado ni interesado en cómo funcionan los programas y/o aplicaciones que usan a diario, no obstante existen otros tipos de usuarios interesados en crear, aprender y difundir. De acuerdo a Resnick et al (2009) los usuarios tienden a moverse en cuatro diferentes roles de participación, los cuales son: a) *Consumidor pasivo* que refiere a aquellos usuarios que observan las acciones de otros usuarios pero nunca intervienen por si mismos; b) *consumidor activo* que agrupa aquellos usuarios que dan retroalimentación a lo que han hecho otros usuarios, c) *productores pasivos* que es la categoría que abarca a las personas que son capaces de crear proyectos pero no se arriesgan a compartirlos en la web; y por último d) los *productores activos*, quienes son los usuarios que se disponen a crear, compartir e interactuar con lo desarrollado por otras personas y ellos mismos.

Esta nueva era de la información conlleva a que los nativos digitales no solo sean usuarios conocedores de destrezas diferenciadas en el uso de dispositivos, sino que también requiere de jóvenes con la capacidad de desarrollar productos digitales. Para ello es necesario dirigir la atención a otro tipo de herramientas de usos más especializados como photoshop, corelDRAW, 3D Studio max, entre otras, las cuales cuentan con principios básicos en común que son propios de la programación computacional, estos conceptos esenciales son los que cimentan la mayoría de software que existe en la actualidad. Ahora, estas herramientas mencionadas son programas que no tienen como propósito el desarrollo de habilidades de programación, para ello existen otras herramientas todavía más especializadas las cuales se dividen de acuerdo al lenguaje de programación, que son en su mayoría, un conjunto de símbolos y reglas que permiten la construcción de estructuras de código que a su vez conforman los programas (Morales & Morales, 2014).

Las herramientas informáticas destinadas para la programación también son selectas y parecen esquivas a cualquier tipo de usuario. Muchos de estos lenguajes de programación son complejos, requieren que las personas involucradas adquieran de memoria una extensa variedad de elementos y su correcta forma de escritura de acuerdo a la interfaz, por lo tanto, la inmersión de los jóvenes en un rol activo con las tecnologías se hace más distante. En este propósito, este estudio pretende mostrar como una herramienta de programación (Scratch) basada en un lenguaje menos abstracto, contribuye a que los jóvenes que interactúan con ella sean capaces de desarrollar los conceptos computacionales inherentes a dicha interfaz, los cuales, a su vez, como ya se mencionó en este apartado, son esenciales en la configuración de usuarios productores activos.

Pensamiento Computacional

El pensamiento computacional es un proceso cognitivo que refleja un conjunto de habilidades a menudo orientadas al desarrollo de un producto de programación, además de orientarse hacia un impacto pedagógico (Selby y Woollard, 2013). El PC abarca cinco destrezas: la primera es la habilidad de pensar en abstracciones, esto hace referencia a la capacidad de distinguir distintas capas de una secuencia de código y además de distinguir las relaciones entre ellas. La segunda es la habilidad de pensar en términos de descomposición, es decir la capacidad de desagregar los códigos en segmentos, lo cual es necesario cuando se enfrenta a tareas o sistemas de código complejos. La tercera habilidad es la de pensar algorítmicamente, la cual se interpreta como el procedimiento paso a paso para resolver un problema de acuerdo a lo que se presente. En la cuarta habilidad se piensa en términos de evaluación de los procesos de programación y así determinar su eficiencia y posibles resultados. Por último en la quinta y última habilidad el pensamiento se origina a

partir de generalizaciones, lo que hace referencia a la capacidad de moverse de un contexto específico a uno más amplio y complejo (Selby y Woollard, 2013).

Las habilidades mencionadas son descritas en función de un dispositivo o artefacto, que en este caso sería el computador, donde las formas de pensar se hacen explícitas al momento de programar (Selby y Woollard, 2013). Aun así, esto no excluye que estas habilidades de pensamiento computacional se puedan dar en otras disciplinas o contextos cotidianos.

Como se dijo al principio, este concepto de pensamiento computacional tiene el propósito de impactar a nivel pedagógico en los currículos educativos. A tal punto que se espera que los espacios educativos introduzcan herramientas informáticas que sean acordes a la edad y además que estimulen este tipo de pensamiento en los jóvenes. Para el caso de esta investigación la herramienta seleccionada es Scratch, la cual reúne varias de las habilidades ya descritas y las agrupa en tres dimensiones de acuerdo a Brennan y Resnick (2012), éstas por lo tanto serán abordadas a continuación.

Dimensiones del Pensamiento Computacional

Dentro del pensamiento computacional existen tres dimensiones que se han depurado a través de la programación con Scratch (Brennan & Resnick, 2012) pero que son extrapolables a otros entornos que indaguen sobre el PC. La primera conocida como *conceptos computacionales* (la cual será evaluada en este estudio en relación con scratch). La segunda entendida como *prácticas computacionales* y la tercera llamada *perspectivas computacionales*. A continuación se hace una descripción detallada de cada una de estas dimensiones:

Los *conceptos computacionales* son una serie de elementos esenciales que definen las posibles relaciones moldeables dentro de un ambiente de programación. Para el caso

específico de Scratch se han identificado siete conceptos que se pueden transferir a otros escenarios, tanto de programación como cotidianos, estos conceptos son: a.) las *secuencias*, que aluden a una serie de pasos que se deben seguir para determinar la acción que se quiere producir, b.) *los loops* o bucles, corresponden a un determinado tipo de acción que lleva al usuario a repetir la misma acción por determinado tiempo; c.) *los eventos*, que en pocas palabras corresponden a la causalidad de una cosa sobre otra, en este caso de un bloque sobre otro, d.) *el paralelismo*, que son secuencias de instrucciones que se dan al mismo tiempo; e.) *los condicionales*, que se resumen en el uso de un bloque que condiciona los resultados de una o varias acciones; f.) *los operadores* que proveen el soporte para el uso de expresiones lógicas, matemáticas y textuales, permitiendo encadenarlas entre si; y por último g.) *los datos* que implican almacenar, recuperar y actualizar valores, para ello Scratch ofrece dos contenedores de datos que son las variables y las listas (Brennan y Resnick, 2012).

Por otro lado, las *prácticas computacionales* corresponden a las prácticas de diseño que se observan en los jóvenes al interactuar con herramientas de programación como Scratch. Se trata entonces del salto que se da del *qué* se está aprendiendo al *cómo* se aprende con dichos conceptos computacionales. Brennan y Resnick (2012) han identificado cuatro tipos de prácticas computacionales que son las siguientes: 1) *Ser gradual e interactivo*, que se refiere a un proceso adaptativo en donde el proyecto en curso puede ir cambiando de acuerdo a pequeñas aproximaciones al resultado esperado, 2) *hacer pruebas y depurar*, que hace referencia a un ejercicio de ensayo y error, donde el usuario debe responder a dificultades emergentes en el proceso creativo para conseguir el resultado esperado; 3) *reutilizar y combinar*, que es principalmente construir a partir de lo ya desarrollado por otros; finalmente 4) *Abstrayendo y modularizando*, que consiste en construir algo grande a

partir de pequeñas partes, y además poder dividir distintas partes de un mismo escenario en módulos separados.

Ahora bien, la última dimensión del *pensamiento computacional* que desarrollan estos autores se denomina “*las perspectivas computacionales*”. Ésta hace referencia precisamente a los cambios de perspectiva de los jóvenes sobre sí mismos, sus relaciones con los demás y con la tecnología. Estos cambios se dividen en tres momentos. El primer momento se da cuando los estudiantes se dan cuenta que al involucrarse con estos elementos computacionales pueden *expresarse* de una manera distinta a como lo harían en una red social, donde solo serían consumidores pasivos. El segundo momento es cuando se establecen *conexiones* con otras personas que hacen parte de la comunidad digital, como en el caso de Scratch, en donde pueden adquirir conocimientos y mejorar los proyectos que suban a la plataforma, puesto que hay una constante retroalimentación, un trabajo *con* los otros y *para* los otros. El último momento es donde los jóvenes se atreven a *cuestionar* la realidad que se les presenta, se sienten empoderados, como participantes activos dispuestos a negociar con el mundo tecnológico con el fin de darle sentido (Brennan y Resnick, 2012).

Estas dimensiones están situadas en el trabajo con Scratch y como se puede apreciar inciden gradualmente en el crecimiento de usuarios con un rol activo en la producción de contenidos. De tal manera cada dimensión enmarca a la anterior, no son independientes una de otra, dando a entender que a medida que el aprendiz adquiere los conceptos computacionales va construyendo unas prácticas computacionales respectivas a esos conceptos, y a su vez se genera un giro en la forma en que conciben la tecnología. Estas dimensiones han ido emergiendo a partir de herramientas de programación con propósitos educativos, las cuales gradualmente se han moldeado con fin de evidenciar más aspectos del PC.

El Surgimiento del Pensamiento Computacional a través de la Programación: el Caso de LOGO

El concepto del pensamiento computacional es relativamente reciente y tiene como su principal antecedente a LOGO, que se ha concebido, sin duda, como la primera herramienta que empleo programación sin hacer uso de líneas de código complejas. En principio LOGO consiste en controlar el movimiento de una tortuga por la pantalla de un computador mediante instrucciones que los jóvenes tecleaban, bajo la metáfora de enseñarle nuevas palabras a dicha tortuga (Ruiz, 1994). Esto se trae a colación debido a que esta plataforma se caracterizaba, entre otras cosas, por poseer una interfaz de programación visual, lo que incide en un entendimiento más asequible al usuario.

LOGO se caracterizó además por ser un programa que demostró tener implicaciones positivas en distintos aspectos tanto de aprendizaje como de desarrollo social. Por un lado estaban los beneficios en la enseñanza de matemáticas, en donde, por ejemplo en términos de geometría, sentido de espacialidad, proporciones y razones numéricas los estudiantes mostraron un mejor desempeño cuando usaban LOGO (Battista & Clements, 1988; Battista & Clements, 1991; Clements & Battista, 1992a; Clements & Battista, 1992b; revisado en Clements & Samara, 1997). Por el otro lado los estudiantes se veían involucrados en actividades con ciertos grados de dificultad, en donde se hacía necesario desarrollar competencias de resolución de problemas y negociación con los otros usuarios de LOGO (Clements & Samara, 1997).

Adicionalmente, la literatura también destaca el rol que cumplían los docentes guiando a los estudiantes en la adquisición de habilidades encaminadas a la resolución de problemas presentes en LOGO. Entre las habilidades que se destacan por la mediación del docente se incluyen las de planificar, el monitoreo cognitivo, decidir sobre la naturaleza del problema

y seleccionar una representación para resolverlo (Clements & Samara, 1997). La importancia de la mediación en el proceso de aprendizaje también tiene efecto en la transferencia a tareas que no necesariamente requieren de un ordenador. Es decir que, por ejemplo, en actividades artísticas se destacaba un mejor desempeño creativo tanto en el trabajo con el programa, como con actividades alternas que requerían de papel y lápiz (Vaidaya & McKeeby, 1984; citado en Clements & Samara, 1997).

Ahora, en cuanto a las ventajas en el aprendizaje de programación, o como se ha venido mencionando en este escrito, el pensamiento computacional, LOGO (aún en estos días) se destaca por mostrar el vínculo que existe entre los comandos y la modificación de la figura, lo cual implica una correspondencia visual muy apropiada para vincular las acciones (códigos) sobre los dibujos (geometría de la figura) (Clements & Sarama, 1997). Además de esto, los estudiantes que manejan LOGO pueden explorar la función de nuevos comandos y posteriormente modificar los códigos, lo cual motiva la experimentación. Aún con todo lo expuesto y a pesar de todas las ventajas mencionadas de este lenguaje de programación, también tiene ciertas limitaciones que son muy propias de sus propósitos y de la interfaz que maneja. En otras palabras es una interfaz restringida por el tipo y la cantidad de comandos que se pueden operar, lo cual lleva a una deficiencia en los conceptos de programación que se aprenden, que son de tipo secuencial, dejando de lado conceptos como las variables (Leron, 1985; Mc Allister, 1986; citado en Ruiz, 1994., pág. 114). Ahora, aquellos que llegan a manejar estos conceptos, lo hacen mecánicamente, sin mostrar interés por el proceso subyacente a dicho procedimiento, por lo tanto no hay comprensión de la naturaleza de las acciones y sus resultados (Leron, 1985; Burton & Magliaro, 1986; citado en Ruiz, 1994., pág. 115). Finalmente, otra de las falencias de LOGO es que los estudiantes no usan heurísticos para la resolución de problemas sino que

tienden a funcionar con ensayo y error (Ruiz, 1994). No obstante con el paso del tiempo han surgido nuevas herramientas basadas en el lenguaje de programación visual, las cuales han ido depurando las falencias de LOGO permitiendo medir otros aspectos esenciales del pensamiento computacional.

Programas Actuales Para el Desarrollo del Pensamiento Computacional

A medida que se ha avanzado en la comprensión del PC la literatura ha identificado dos características mínimas que deben tener las herramientas que en la actualidad están basadas en el lenguaje de programación visual. Estas características son: a) la visualización de la programación a través de bloques y b) la posibilidad de compartir productos en comunidades en línea (Maloney, et al., 2010).

En relación con la primera característica: se han desarrollado varias plataformas basadas en bloques. Algunas de éstas son por ejemplo Alice, AgentSheets, Greenfoot y Scratch, todas tienen en común que cuentan con un entorno virtual que facilita a los usuarios crear diversos proyectos acorde a sus intereses (Maloney, et al., 2010). Alice, por su parte es un software de programación en 3D que permite la creación de animaciones para contar historias, desarrollo de juegos interactivos o videos (Carnegie Mellon University, 2014). Una de las grandes ventajas de Alice, además de ser gratuito, es que es de fácil uso, puesto que funciona como una introducción a la programación orientada al objeto, en donde el usuario elabora escenarios y puede ver el progreso en el acto. Otras herramientas muy semejantes son AgentSheets y AgentCube, las cuales facilitan la elaboración de simulaciones para transmitir ideas y la creación de juegos. Tanto AgentSheets (que es la versión en 2D) como AgentCube (que es en 3D) propician la asimilación de conceptos de ciencia computacional, pensamiento algorítmico y computacional (Repenning, 2014). Por último Greenfoot, es una herramienta basada en el lenguaje de Java, óptima para el

desarrollo de juegos y simulaciones, que en si es una interfaz simple que combina la programación basada en texto y la visualización gráfica.

En cuanto a la posibilidad de compartir productos en las comunidades en línea, los cuatro ejemplos anteriores permiten a los usuarios subir sus trabajos en la web. Alice por su parte tiende más a la publicación de videos sobre los proyectos, AgentSheets si permite publicar en la web como Applets de Java al igual que Greenfoot, además esta última maneja un sitio en línea, donde los demás usuarios comentan e interactúan sobre los proyectos disponibles. Sin embargo, estos cuatro son dirigidos a una población más “adulta”, es decir, estudiantes que tengan cierta experiencia con los conceptos de programación de Java y que estén pasando a una etapa de mayor desempeño en computación (Maloney, et al., 2010, p. 13). En contraste, Scratch, se dirige a estudiantes más jóvenes, que están iniciándose en la programación; así, viene a ser el precursor de las otras plataformas ya mencionadas, puesto que su sistema permite usar la exploración y experimentación más intuitivamente, como cuando los jóvenes interactúan con LEGO (piezas de plástico que principalmente tienen forma de bloques). Así pues, Scratch cuenta con otras ventajas para la formación de estudiantes novatos en programación, como lo es el hecho de promover y facilitar la creación de una gama amplia de proyectos (no solo videojuegos y animaciones), además de resultar más fácil la publicación de éstos en el sitio web, permitiendo reforzar conocimientos con otros proyectos de los usuarios activos en la comunidad (Maloney, et al., 2010, Pp.14).

Scratch

En el año 2007 los investigadores del MIT (Massachusetts Institute of Technology) desarrollaron Scratch, una herramienta multimedia que implica arrastrar y soltar bloques, que corresponden a funciones y acciones para programar la conducta de ciertos objetos en

un ambiente virtual. Scratch permite emprender proyectos que van desde historias, presentaciones, animaciones, representaciones artísticas, juegos y hasta composiciones musicales (Resnick, et al., 2009). Entre las características de Scratch que hacen de esta plataforma sobresaliente sobre otras con los mismos propósitos, encontramos que está disponible en más de 40 idiomas, cuenta con una variedad de guías desarrolladas por miembros del grupo y sus respectivas traducciones. Además dispone de un espacio donde los usuarios de distintas partes del mundo interactúan con los proyectos de los demás usuarios, dándoles la oportunidad de crear y combinar proyectos propios o trabajar sobre unos ya existentes.

Entonces, Scratch toma ventaja en la construcción de usuarios más enterados y activos, cuyas necesidades se ven impresas en la “inteligencia colectiva”, la tercera pieza fundamental para Jenkins (2006), según la cual, al no haber usuarios omnisapientes, cada cual puede ofrecer su parte de conocimiento para la elaboración de algo más complejo y más serio. De esta manera, el sitio web de Scratch es un ejemplo claro de cómo darle forma a esa inteligencia colectiva o dicho de otra manera, de ofrecer un espacio para decantar las habilidades, intereses y conocimientos que unos poseen y otros carecen. Se trata de un escenario donde prima la interacción entre usuarios que se involucran en distintos niveles de desempeño con la plataforma, esto se ve en el tipo de proyectos compartidos y de los comentarios o aportes dados a otros proyectos, lo cual también puede ser una representación de las dimensiones del pensamiento computacional que ya se mencionaron.

El Rol de Scratch en el Aprendizaje del Pensamiento Computacional

Se sabe que el proceso de aprendizaje del pensamiento computacional, requiere de un agente de procesamiento de información para poder representar las soluciones a un problema determinado (Lee, et al., 2011), estos agentes pueden ser un humano, un

computador o ambos (Wing, 2006). En este caso, dicho agente es la combinación del aprendiz y el ordenador, específicamente mediante Scratch, el cual provee un contexto y una serie de herramientas conceptuales que se traducen en bloques con funciones asignadas.

El pensamiento computacional como lo plateo Jeannette Wing (2008) no se introdujo como la posibilidad de hacer que las personas pensarán como lo haría una computadora sino con la intención de fomentar el desarrollo de habilidades mentales para poder “computar” soluciones de problemas cotidianos y propios de las ciencias de la computación (Lu y Fletcher, 2009). Comúnmente cuando se inicia un proceso de aprendizaje ya sea en matemáticas o lenguaje se atiende a que los jóvenes adquieran las capacidades básicas de leer, escribir y habilidades cuantitativas básicas (sumar, restar, dividir, etc.) y más adelante (por lo general en escenarios de educación superior) se les enseña métodos demostrativos en matemáticas y análisis crítico en la lectura (Lu y Fletcher, 2009). En el caso de la ciencia computacional (como asignatura) y el pensamiento computacional (como habilidad) se puede decir que sucede lo mismo. En los escenarios curriculares tradicionales se recomienda enseñar aquellas habilidades de pensamiento ya descritas para luego poder enganchar a los estudiantes en la programación (Lu y Fletcher, 2009). Esto asumiendo las dificultades inherentes a los lenguajes de programación en su concepción clásica, donde los programas son un conjunto de vocabularios y símbolos.

De la misma manera, en el entorno educativo actual no se capacita al estudiante en la comprensión de los rudimentos de las matemáticas, del lenguaje ni tampoco de las tecnologías que tanto usan, específicamente aquellas cuya columna vertebral es la programación (Grover y Pea, 2013). Por el contrario se enseña a las nuevas generaciones a partir de currículos tradicionales que asumen limitaciones propias del lenguaje en el que se

presenta la programación (Lu y Fletcher, 2009). Por ejemplo según Kazimoglu et al (2012) los jóvenes quienes están aprendiendo a programar pueden presentar ciertas dificultades las cuales tienden a la adquisición de un conocimiento superficial y no desarrollan efectivamente habilidades de resolución de problemas. Esto puede estar en consonancia con los métodos y los contextos empleados para enseñar dichos conceptos, por ejemplo no es lo mismo instruir al estudiante en el manejo de Excel como procesador de datos que emplearlo para resolver problemas en distintas áreas, como ciencias o matemáticas.

Así, el problema puede tener origen primero, en el uso de los programas dirigidos solo para responder a las exigencias de la clase de informática sin que se extiendan a la resolución de problemas que se plantean en otros escenarios dentro y fuera de la escuela (Jaramillo, 2005), y segundo, que estos no tengan como propósito el aprendizaje de programación. Por su parte, Scratch busca dar respuesta a estas dos dificultades proporcionando una interfaz donde el estudiante puede crear proyectos en distintas áreas de conocimiento y sobre todo procura la inmersión en la programación sin ser demasiado complejo. Sin embargo, para los propósitos de este estudio no es de importancia el proceso en proyectos de artes, música, matemáticas, ni tampoco videojuegos. En este caso, la atención se dirige al nivel de análisis de los elementos lógicos que conforman cada uno de los proyectos mencionados anteriormente y para llegar a ellos es necesario comprender los *conceptos computacionales*, los cuales subyacen al programa.

Aprendizaje en Scratch: Construyendo Usuarios Productores de Programas

Investigaciones relacionadas a Scratch han mostrado que este programa permite que los estudiantes desarrollen habilidades de pensamiento computacional al mismo tiempo que entienden las relaciones existentes entre escribir y programar (Burke & Kafai, 2010). En la misma línea, se ha encontrado que el manejo de Scratch facilita el aprendizaje de otros

lenguajes de programación (Malan y Leitner, 2007). De este modo, se ha identificado que Scratch puede promover actitudes positivas hacia las matemáticas mediante la creación de juegos (Choi, Jung y Baek, 2013), además de conductas relacionadas con la creación de información en prácticas digitales como por ejemplo: el desarrollo de contenido, organización y presentación de la información (Koh. K. 2013). A un nivel social, se ha encontrado que Scratch favorece el entendimiento de las prácticas colaborativas de las comunidades en línea (Fields, Giang, y Kafai, 2013), la relación con la expresión artística creativa en medios digitales (Peppler y Kafai, 2005) y la construcción de una cultura digital más participativa (Jenkins, et al., 2006).

En general Scratch promete una configuración distinta en contraste con otras herramientas del mismo corte en cuanto a ciertos elementos en la programación y retroalimentación, que hacen el proceso de aprendizaje mucho más intuitivo y natural. En este sentido cuando un estudiante se embarca en el proceso de aprendizaje en Scratch tiene la oportunidad de identificar los bloques de comando que se diferencian por categorías asignadas a colores, además pueden aprender a encajar los bloques entre si identificando sus formas. A su vez, en el proceso de programación, son eliminados los mensajes de error con el propósito de hacer el aprendizaje semejante al de manipular componentes mecánicos o electrónicos en donde las piezas encajan de acuerdo a su forma, así los bloques de Scratch se acoplan de manera intuitiva (Maloney, et al., 2010). A diferencia de otras plataformas de programación, Scratch ofrece un flujo de la actividad continuo, es decir, si el estudiante quiere ver que hace cierta configuración de bloques no tiene que exportar su trabajo o ponerlo a correr en segundo plano, por el contrario puede ponerlo en marcha incluso si aún no está terminado, agregando o quitando bloques.

También se han desarrollado videojuegos con el fin de enseñar el pensamiento computacional, donde existen elementos de retroalimentación que son semejantes a los presentes en Scratch. Dichos juegos, entre los que están: Robocode, Colobot, Catacombs, Saving Serra, Elemental y Program your robot (Kazimoglu, Kiernan, Bacon y Mackinnon, 2012), explícitamente requieren la programación de agentes inteligentes u otros contenidos. El mecanismo de recompensa en estos videojuegos resulta un aspecto diferencial en comparación con Scratch, debido a que en este último se da a otro nivel de interacción, en donde el usuario ve ganancias en la implementación y disposición de bloques para obtener un resultado que va más allá que de algo inmediato, mientras que en estos juegos la recompensa se da por puntajes y logros superados. Así, Scratch posee los elementos suficientes para que el usuario desarrolle proyectos en donde su estímulo está dado por la consecución de un producto de programación, lo cual facilita la adopción de conceptos computacionales. Tales conceptos son esenciales porque se configuran a favor de que los jóvenes asuman un rol activo y creativo, que requiere una inmersión en el proceso de código, el cual Scratch presenta de una manera menos abstracta (Maloney, et al., 2010).

Evaluación del Pensamiento Computacional y Scratch

A pesar de las notables ventajas de Scratch para desarrollar el pensamiento computacional, en la actualidad el consenso sobre la forma de evaluarlo se encuentra vagamente estructurado. Los estudios que han realizado la medición de este tipo de pensamiento no se enfocan en las dimensiones clave del PC (Werner, 2012) además de ser ausentes las mediciones específicas de los efectos que el trabajo con Scratch tiene hacia éste. De hecho, la mayoría de las investigaciones señaladas anteriormente se concentran en dilucidar como ésta plataforma es funcional para el aprendizaje de ciertas áreas específicas como las artes, escritura y matemáticas (Burke, 2012, citado en Lye y Koh, 2014). Algunas

razones que justifican este panorama son: a) las dificultades que hay para hacer una medición de este tipo de conceptos en el aula, ya sea por la población, la disponibilidad de tiempo o por el tipo de instrumentos (Brennan y Resnick, 2012); y b) la tendencia de los docentes a descartar este tipo de aprendizaje por considerarlo de campos exclusivos de ciertas disciplinas y muy complicados para un usuario no entrenado (Yadav, Mayfield, Zhou, Hambruch, y Korb, 2014). Esta última afirmación, tiene que ver con que no hay una “cultura de programación” (Maloney, et al., 2004), que permita ir más allá del uso de software de consumo. Por lo cual, se puede decir que aparentemente no existe una cultura en la cual las prácticas digitales se transfieran a otros contextos de la cotidianidad de los estudiantes y además desarrollen habilidades como pensar creativamente, razonar sistemáticamente y trabajar colaborativamente, aunque estas sean consideradas fundamentales para el siglo 21 (Resnick, et al., 2009).

Ahora, si bien estas investigaciones mencionadas procuran destacar el pensamiento computacional a través de la resolución de problemas en áreas de aprendizaje que son esenciales para la educación formal, se hace evidente un vacío en las mediciones que den cuenta del desarrollo de los conceptos computacionales implícitos. En otras palabras, la revisión hecha para este estudio ha permitido reconocer que no se está indagando por el cómo los usuarios emplean y comprenden: las secuencias, los loops, los eventos, los paralelismos, los condicionales, los operadores y los datos, expresados como bloques en Scratch. De hecho, las conclusiones de dichos estudios no apuntan a mostrar si hay un desempeño, mejor o peor, en los conceptos computacionales que subyacen a cada programa elaborado, independiente del área, sino a mostrar cómo Scratch cumple su función como herramienta multimedia. Por ende, en este estudio se pretende evaluar el desarrollo de estos conceptos a partir del uso de Scratch y la incidencia de factores endógenos (p.ej.

complejidad del concepto) y exógenos (p.ej. el uso y acceso de TICs). Además, se pretende dar cuenta de cómo se pueden transferir esas nociones computacionales esenciales a otro dominio, a través de un instrumento isomorfo a las condiciones que presenta la interfaz de Scratch.

Transfer y Aprendizaje

El *transfer* suele ocurrir cuando el aprendizaje en un contexto y con un conjunto de herramientas y/o materiales tiene impacto en el desempeño en otro contexto con otros elementos (Perkins & Salomon, 1992). En otras palabras se puede hablar de este tipo de aprendizaje en muchos entornos cotidianos, en donde, básicamente la persona puede poner en marcha procesos que aprendió en un ambiente y ejecutarlos en otro que sea isomorfo, mejorando el desempeño en esa nueva situación (Luger y Bauer, 1978) (en el mejor de los casos), por ejemplo lo aprendido conduciendo un carro se puede transferir a una situación con un camión.

Bajo esta perspectiva, este tipo de aprendizaje es esencial en los entornos académicos, ya que en general los espacios educativos formales aspiran a conseguir un nivel de *transfer* alto de los temas de las clases a otros escenarios de la vida diaria, no obstante resulta ser una meta difícil de alcanzar (Perkins & Salomon, 1992). Los estudiantes pueden involucrarse en el desarrollo de ejercicios de matemáticas durante una clase y tener éxito en ello, pero cuando se les evalúa mediante una serie de ejercicios que combinan técnicas y los resultados no son los esperados, queda en evidencia que no hubo un buen transfer del conocimiento previo que el docente impartió. Se pueden encontrar varios ejemplos como éste y en múltiples áreas del aprendizaje. Sin embargo, para los objetivos de esta investigación se destacarán los ejemplos y características en escenarios asociados a la programación y el pensamiento computacional.

Antes de abordar ejemplos de *transfer* mediante el uso de herramientas de programación vale la pena destacar algunos aspectos y condiciones sobre el transfer en el aprendizaje. Para comenzar hay que destacar lo que Perkins & Salomon (1992) denominan “*transfer* cercano” y “*transfer* lejano”. El primero hace referencia al *transfer* que se da entre contextos similares, como por ejemplo cuando un estudiante realiza un examen en donde evalúan ejercicios combinados que son del mismo tipo que han realizado en sus tareas en casa. Por otro lado el “*transfer* lejano” se refiere a un tipo de *transfer* que requiere de mayor esfuerzo y abstracción por parte del estudiante, es decir plantea la necesidad de extrapolar dicho conocimiento a un escenario lejano y cotidiano.

De acuerdo a la literatura, se habla de unas condiciones para que se dé el *transfer*, que aunque no sean obligatorias sí han sido reportadas por investigadores en sus estudios: 1) *practica profunda y variada*, la cual se enfoca en el conocimiento práctico al realizar una actividad, no solo en repetidas ocasiones sino también en variedad de contextos, como lo expreso Luria (1976) en un experimento de alfabetización y educación en Rusia (como se cita en Perkins & Salomon, 1992, p. 6). 2) *Abstracción explícita*, que hace referencia a cuando los aprendices son capaces de abstraer atributos esenciales de una situación, como se puede observar en el estudio de Gick and Holyoak (1980, 1983) quienes presentaron a determinados sujetos una situación problema con una solución concreta y luego los enfrentaron a cierta situación análoga (como se cita en Perkins & Salomon, 1992, p. 6). 3) el *Auto monitoreo activo*, el cual, por su parte se centra en la actividad de auto monitorear el proceso de pensamiento que se da cuando se resuelve un problema específico como se dio en la investigación de Belmont et al. (1982), quienes afirmaron que estos procesos de auto monitoreo en niños/as les permitió a los investigadores reconocer si estaban aplicando la estrategia que habían aprendido previamente los participantes (como se cita en Perkins &

Salomon, 1992, p. 6). 4) *Conciencia despierta*, que se refiere a un estado de alerta tanto de las actividades en las que un sujeto se involucra como de su entorno, este estado de conciencia activa fomenta el auto monitoreo y la abstracción explícita. 5) Finalmente esta *usar una metáfora o analogía*, en donde el material nuevo (en este caso la prueba en LEGO) se estudia a la luz del material o situación previa (la prueba Scratch) pero modificando la forma en que se presenta dicha información (Perkins & Salomon, 1992).

El termino de *transfer* ha sido relacionado con los siguientes mecanismos psicológicos:

- a) la *abstracción* de elementos que son idénticos en su función pero que se presentan en modelos diferentes, de manera que en distintos contextos se puedan presentar los mismos procesos, luego está b) el *transfer por affordances* según el cual, de acuerdo con Greeno (1993), el aprendiz adquiere un esquema de acción que depende de *affordances* u oportunidades de acción, que si se presentan en la situación de *transfer* y el sujeto las puede identificar podría aplicar el esquema al nuevo contexto. Finalmente se habla del c) *transfer Low y High*, propuesto por Perkins & Salomon (1989, 1990, como se cita en Molina, 2002). De acuerdo a estos autores, para que suceda el *Low transfer* es necesario que se dé un automatismo en el procedimiento en cuestión (Molina, 2002), esto suponiendo que el estímulo de la situación *transfer* es similar al del contexto previo, este tipo de *transfer* se da junto a un *transfer* cercano. En cuanto al *High transfer* este obedece a un mecanismo más estructurado y requiere una gran abstracción del contexto de aprendizaje siendo más consciente que el anterior, por lo tanto, requiere más tiempo y esfuerzo mental (Perkins & Salomon, 1992).

Estas condiciones y mecanismos fueron puestos a prueba por varios investigadores, según los cuales se encontraron resultados positivos sobre todo en lenguajes de programación. Por ejemplo, el caso de la investigación de Clements and Gullo (1984) que

destacaron *transfer* positivo en ciertas medidas cognitivas mediante la programación con LOGO (como se cita en Perkins & Salomon, 1992, p. 5). Por otro lado, Goldenson (1996) describe en su artículo tres estudios de campo, en dos de ellos estudiantes de noveno grado aprendieron métodos de programación usando Karel the Robot (una interfaz de programación que se centra en un robot que se mueve por un plano cartesiano compuesto de calles y avenidas; éste posee una semántica simple, hace énfasis en la programación modular y en la abstracción procedural) quienes posteriormente se desempeñaron considerablemente mejor en una serie de tareas de escritura que otros estudiantes en un grupo control. El tercer estudio mostró que otro grupo de estudiantes que recibió un curso introductorio de programación en Karel se desempeñó significativamente mejor en unas tareas de mayor grado de dificultad usando otro lenguaje de programación llamado Pascal.

En síntesis, se puede distinguir dos tipos de transfer que vale la pena retomar, el reflexivo (*Low transfer*) el cual requiere que el conocimiento adquirido se automatice hasta tal punto que en situaciones que presenten estímulos y condiciones semejantes activen y faciliten la ejecución en tareas de *transfer*. El otro tipo es el *transfer* consciente (*High transfer*), obedece a un transfer que requiere de abstracción activa y exploración de posibles conexiones con otras áreas o escenarios cotidianos. En ambos casos se puede decir que en la realidad educativa no se dan situaciones de aprendizaje que promuevan el desarrollo adecuado de cada tipo de *transfer*. De esta manera, se hace necesario introducir en las aulas donde se imparte programación y ciencias computacionales, metas orientadas a lograr estos tipos de *transfer* ya que, como lo indica la literatura, a diferencia de otras áreas como la Historia (Perkins & Salomon, 1992), la programación en computación y a su vez, la adquisición de conceptos computacionales implícitos en dicha programación, tiene un

mayor espectro de *transfer* para resolución de problemas en otros dominios (Pea y Kurland, 1984).

En esta investigación se usó un instrumento con el fin de medir el *transfer* mediante una prueba de LEGO como se mencionó anteriormente, la cual consiste en los mismos ejercicios que realizaron los estudiantes con Scratch pero empleando piezas de LEGO a las que se les asignó correspondencias con los bloques manejados en la interfaz de Scratch. Esta prueba podría servir para elicitación un *Low transfer* de los conceptos computacionales descritos por Brennan & Resnick (2012) ya que la prueba que se aplica con LEGO no cuenta con los elementos para determinar si el *transfer* que se da repercute en otras áreas de conocimiento o de la cotidianidad.

Cabe aclarar que usar bloques de LEGO no solo hizo más práctica la forma de aplicar la segunda prueba, sino que también esta elección está soportada por literatura en donde han usado algunas de las presentaciones de LEGO para responder a interrogantes en el campo educativo. Una de las versiones más usadas es el kit LEGO Mindstorms que es popular por ser usado para desarrollar dispositivos de robótica en entornos escolares (Brandt y Colton, 2008). Este kit de LEGO facilita la introducción de los estudiantes en entornos de programación destinados para interactuar con los bloques con los que dispone. Por otro lado está el set de construcción LEGO WeDo, que además de haberse popularizado más que Mindstorms, es menos complejo. Si bien ambos se centran en el desarrollo proyectos de robótica, LEGO WeDo tiene una cantidad más reducida de elementos avanzados, cuenta con un software de trabajo más simple y está más orientado en la programación centrada en iconos en vez de las instrucciones en código o textuales (Mayerová, 2012). Este último cuenta también con compatibilidad con Scratch. No obstante, aunque estos estudios no dan cuenta de si LEGO en sus distintas presentaciones es indicado para propiciar *transfer* en

otros dominios, si dan cuenta de los usos y las ventajas que tiene en la introducción de temas de programación básicos, generando entusiasmo en los usuarios y más aún, son isomorfos a la interfaz de programación que usan. En este sentido usar bloques de LEGO clásicos, como se hace en este estudio, supone una buena alternativa puesto que, además de ser más accesible que LEGO Mindstorms y WeDo cumple con la condición de *transfer* de *usar una metáfora o analogía* presentando la información de los mismos ejercicios que se presentan en Scratch, pero mediante una interfaz compuesta de piezas de LEGO que suponen las mismas funciones y principios lógicos que en Scratch.

Como se ha podido apreciar el pensamiento computacional es una destreza que reúne una serie de formas de pensar para resolver problemas que se extienden más allá del uso de un computador. Este tipo de pensamiento requiere de unos conceptos lógicos los cuales dan cuerpo a los lenguajes de programación. Scratch, por su parte es una de las varias herramientas de programación centradas en un lenguaje de programación visual o centrada en el objeto, que pone a prueba el pensamiento computacional y sus distintas dimensiones. Una de estas dimensiones, tal vez la menos evaluada hasta el momento es la de los conceptos computacionales, la cual sin lugar a dudas constituye uno de los pilares del PC, no solo para su entendimiento sino también para su aprendizaje. Es por esto que en este estudio se pretende abordar el desarrollo de dichos conceptos relacionados a Scratch por parte de los estudiantes y a su vez determinar ciertos aspectos que influyen en el respectivo aprendizaje de los siete conceptos ya mencionados. Paralelamente se evaluará en qué medida esos conceptos lógicos computacionales se transfieren a otro dominio con características isomorfas.

Método

A continuación se presentará la metodología empleada para el desarrollo de esta investigación. Primero se describirán las características de la población con la que se contó. Segundo se definirán los instrumentos usados y sus propósitos. En tercer lugar se describirá el procedimiento que se llevó a cabo con cada uno de los instrumentos y los momentos en que se aplicaron. Como cuarto y último punto se hablará de la forma en que se codificó la información y la rúbrica que se empleó para ello.

Participantes

Para esta investigación se contó con estudiantes de dos colegios distritales de Bogotá que fueron el Colegio Distrital Manuel Cepeda Vargas y el Instituto Técnico Industrial Francisco José de Caldas. Los estudiantes del primer colegio se establecieron como grupo experimental, porque todos los sujetos de este colegio manejaron Scratch antes y durante la aplicación. El segundo colegio se tomó como grupo control, en donde los estudiantes que participaron en la prueba no manejaban Scratch. Se debe aclarar que el término experimental se usa acá para facilitar la comprensión de los resultados pero que el diseño de este estudio es cuasi-experimental ya que los grupos no fueron asignados aleatoriamente a las condiciones. La muestra total de los participantes fue de 103 estudiantes de los cursos octavo, decimo y once de ambas instituciones (\bar{x} edad = 14,9 años; DE = 1,6 años). Entre estos 75 eran hombres (\bar{x} edad = 14,9 años; DE = 1,6 años) y 31 eran mujeres (\bar{x} edad = 14,8 años; DE = 1,5 años).

Instrumentos

La metodología usada involucró el uso de dos pruebas: la primera correspondió a siete ejercicios pre establecidos en Scratch los cuales fueron elaborados con el fin de representar cada uno de los conceptos computacionales concebidos (secuencias, evento, loops,

paralelismos, condicional y datos) (ver Anexo 1). Esta parte de la prueba se desarrolló en una sala con disposición de computadores (portátil o de escritorio) en los que se encontraba instalado el programa de Scratch versión 1.4. La segunda prueba consistió en la misma secuencia de ejercicios sobre conceptos computacionales pero en este caso los participantes usaron piezas de LEGO previamente seleccionadas (ver Anexo 2), a las que se les asignaron correspondencias con los bloques que se usan en Scratch (ver Anexo 3). Estas tareas fueron diseñadas lo más cerradas posible, es decir que la elección de bloques por parte de los participantes fue restringida a una combinación de bloques, de tal manera que si el estudiante usaba otro tipo de bloques para resolver el ejercicio (tanto en la prueba de Scratch como en la de LEGO) su calificación en la prueba disminuiría.

Adicionalmente se usó un instrumento en forma de encuesta para obtener una caracterización de los participantes en términos de información demográfica, tipo de acceso y uso de tecnologías, conocimiento de software académico y herramientas de programación previos (ver Anexo 4). También se les entregó a los estudiantes un consentimiento informado que debían traer debidamente firmado y aprobado por los padres o acudientes, y durante la prueba ellos firmaban un asentimiento informado (ver Anexo 5).

Procedimiento

El proceso de aplicación estuvo dividido en dos momentos. En el primer momento, (previo a la aplicación de la prueba) se recogieron los consentimientos informados de los estudiantes. Inmediatamente después se les entregó la encuesta y el asentimiento informado, los cuales devolvieron diligenciados antes del inicio de la prueba. Seguidamente se dieron las instrucciones de la prueba que consistían en crear una carpeta en el escritorio del computador en el que cada estudiante estaba y que la guardaran con sus nombres y apellidos. En esta carpeta guardaban los archivos que se nombraban como cada uno de los

ejercicios (secuencias, evento, loops, paralelismos, condicional y datos), como se puede apreciar en el Anexo 1. Los estudiantes tenían a su disposición como máximo una hora para realizar la prueba; sin embargo muchos acababan antes de ese tiempo. Cada uno de los participantes recibía una hoja que contenía las instrucciones y los ejercicios de la prueba y recibían el asentimiento informado que diligenciaban y devolvían antes de iniciar la prueba. Al finalizar la prueba, el investigador guardaba en una unidad de memoria extraíble USB la carpeta que contenía las respuestas a cada ejercicio por cada estudiante. Con el grupo control, este procedimiento cambiaba ligeramente ya que estos estudiantes, al no conocer el programa debían dedicar 5 minutos para revisar una presentación de cinco diapositivas que contenía información básica de la interfaz en la que iban a trabajar; una vez finalizaban su lectura iniciaban la prueba.

El segundo momento se desarrolló ocho días después de la aplicación en Scratch, ya que el tiempo que daba el docente con el grupo era máximo de hora y media. En esta segunda sesión los estudiantes recibieron una hoja que contenía las instrucciones concernientes a esta prueba y los mismos ejercicios de la prueba en Scratch, lo que cambiaba es que para resolver los ejercicios los participantes hicieron uso de fichas de LEGO previamente seleccionadas. Adicionalmente, se le entregó a cada uno unas hojas en donde podían apreciar las correspondencias entre las fichas y los bloques que se usaron en la prueba en Scratch. Por último, ellos recibieron una hoja de papel cuadriculado, tamaño carta en donde consignaban nombres y apellidos, edad, curso y fecha. Además esta hoja representaba el campo de área de programas en Scratch, de manera tal, que en ella los estudiantes colocaron las fichas que correspondían al programa del ejercicio que estuvieran realizando en forma de línea de tiempo (ver Anexo 8). Para poder capturar y sistematizar esta información se les pidió a los estudiantes que levantaran la mano cada vez que finalizaran

un ejercicio para que el investigador se acercará y le tomará una foto a la configuración de fichas.

Codificación de la Información

Una vez se recopilaron las dos fuentes de información para ambas pruebas (Scratch y LEGO) tanto para el grupo experimental y el grupo control, se organizaron los archivos y fotos calificándolas posteriormente mediante una rúbrica, la cual se presenta a continuación.

Tabla 1

Rúbrica de calificación

Valor	Descripción
0	No eligen las fichas o bloques adecuadamente.
1	Elige bien los bloques o fichas pero no los unifica, al menos con el 50% de los bloques. Es decir elige una buena cantidad de bloques o fichas pero no los pone en el orden correcto o los unifica, solo los escoge.
2	Agrupar subunidades de bloques o fichas pero no lleva a cabo la secuencia completa. Completa al menos el 80% de la secuencia del programa.
3	Hace la secuencia global pero comete errores. Es decir omite o incluye el uso de algún bloque o ficha que no afecta la secuencia global.
4	Hace bien la secuencia del programa y unifica los bloques o fichas de acuerdo al ejercicio.

Para poder codificar los productos de los estudiantes mediante esta rúbrica fue necesario emplear unas plantillas con las que se compararon los resultados tanto de Scratch, como los de LEGO. Dichas plantillas se pueden observar detalladamente en el Anexo 6 y 7. Una vez se codificaron los resultados de ambas pruebas y la información de la encuesta, toda la

información se organizó en una base de datos para poder correr análisis estadísticos en el paquete SPSS Statistics versión 21.0.

Resultados

En este apartado se expondrán los resultados de la aplicación de los instrumentos descritos anteriormente. En primera instancia se considerará el análisis de consistencia interna para determinar la fiabilidad de ambos instrumentos aplicados en esta investigación y además el análisis factorial a partir del cual se corrieron otros estadísticos para determinar la forma en que se relacionan dichos factores, que corresponden a agrupaciones de conceptos computacionales. Posteriormente se presentarán dichos análisis estadísticos que se corrieron con las dos pruebas sobre pensamiento computacional de este estudio (mediante Scratch y LEGO) y se destacarán los resultados obtenidos, los cuales permiten apreciar la forma en que se agrupan los conceptos computacionales y su relación con el hecho de manejar o no Scratch. Seguidamente se podrán apreciar los resultados obtenidos de la encuesta de uso y acceso a tecnologías y conocimiento de programas, en relación al grupo experimental y control, con el fin de establecer si hay o no relaciones entre estos datos y los resultados tanto de la prueba en Scratch como con la de LEGO. Finalmente se presentarán las correlaciones entre la prueba de Scratch y la prueba de LEGO con el fin de establecer hasta qué punto esta última prueba funciona como medida de *transfer* de los conceptos evaluados con la prueba en Scratch.

Análisis de Confiabilidad

Para determinar la confiabilidad de la información obtenida mediante los dos instrumentos se calculó el Alfa de Cronbach, distinguiendo entre el grupo experimental y el grupo control. De esta manera, se obtuvo un $\alpha=0.77$ para para las puntuaciones del grupo experimental y $\alpha=0.82$ para el grupo control, lo cual indica una buena confiabilidad de los resultados obtenidos.

Análisis Factorial

Dado que entre la prueba de Scratch y la prueba de LEGO hay en total 14 variables, que representan los conceptos computacionales, se decidió hacer un análisis factorial exploratorio con el fin de establecer las posibles interrelaciones entre las variables por cada instrumento y así determinar las configuraciones subyacentes. En otras palabras se hizo este análisis con el fin de determinar cómo se agrupan los conceptos computacionales, ya que si bien estos se describen en la teoría inicial de Brennan y Resnick (2012), las relaciones entre dichos conceptos nunca han sido testeadas empíricamente.

Tabla 2

Factores agrupados para la prueba de Scratch

Matriz de componentes rotados^a			
	Componente		
	1	2	3
Secuencias en Scratch	-,132	,702	,506
Eventos en Scratch	,250	,839	-,084
Loops en Scratch	,179	-,051	,842
Paralelismos en Scratch	,535	,599	-,202
Condicional en Scratch	,816	,067	-,090
Operadores en Scratch	,673	,303	,328
Datos en Scratch	,796	,091	,302
Método de extracción: Análisis de componentes principales.			
Método de rotación: Normalización Varimax con Kaiser. ^a			
a. La rotación ha convergido en 10 iteraciones.			

Como se puede observar en la tabla 2 de acuerdo a las ponderaciones de los puntajes de las variables en Scratch se forman tres factores: un primer factor que consiste en la

agrupación de eventos (0,70), secuencias (0,83) y paralelismos (0,59); el segundo factor está dado por condicional (0,81), operadores (0,67) y datos (0,79), y el último factor que corresponde a loops (0,84). Estos componentes se pueden agrupar por la dificultad que implican para los estudiantes. El primer factor corresponde a los ejercicios que resultaron más sencillos de elaborar mientras que el factor dos corresponde a los ejercicios más difíciles. El último factor podría indicar que el ejercicio de loops tenía un grado de dificultad intermedio para los estudiantes por lo cual no se agrupa con ninguno de los otros conceptos.

Tabla 3

Factores agrupados para la prueba de LEGO

Matriz de componentes rotados^a		
	Componente	
	1	2
Secuencias en lego	,810	,191
Eventos en lego	,749	,085
Loops en lego	,816	,108
Paralelismos en lego	,309	,676
Condicional en lego	-,106	,637
Operadores en lego	,241	,592
Datos en lego	,133	,740

Método de extracción: Análisis de componentes principales.

Método de rotación: Normalización Varimax con Kaiser.^a

a. La rotación ha convergido en 3 iteraciones.

En la tabla 3 se puede apreciar que para el caso de la prueba en LEGO solo se crean dos factores que se agrupan de la siguiente manera: el primer factor está conformado por secuencias (0,81) eventos (0,79) y loops (0,81), mientras que el segundo factor recoge a paralelismos (0,67), condicional (0,63), operadores (0,59) y datos (0,74). Al igual que en la

prueba de Scratch esta agrupación puede estar asociada al grado de dificultad que implican los ejercicios, así pues el primer factor contempla los conceptos más sencillos para los estudiantes y el segundo factor los más complejos. Además de la complejidad gradual entre cada ejercicio, los conceptos se pueden estar agrupando de acuerdo a su estructura lógica, es decir, el primer factor corresponde a aquellos conceptos que suponen cierta linealidad para su desarrollo, mientras que los del segundo factor implican una mayor capacidad de abstracción.

Diferencias Entre el Grupo Experimental y el Grupo Control en Desempeño

Dado que los datos de toda la población no se distribuyen normalmente fue necesario correr un análisis no paramétrico, empleando la prueba de muestras independientes de Mann-Whitney. Mediante este análisis se evaluó la hipótesis de que los estudiantes del grupo experimental y el grupo control difieren en sus resultados con respecto a los componentes extraídos del análisis factorial, por lo tanto se cruzaron los resultados de ambos grupos con respecto a ambas pruebas. Así mismo, se hace necesario presentar este mismo análisis estadístico pero aplicado a los resultados totales de toda la aplicación y de las pruebas por separado.

Análisis de Mann-Whitney para la media total de la aplicación: Prueba en Scratch y en LEGO

Estadísticos de prueba^a	
	MTotalPr
	b
U de Mann-Whitney	565,500
W de Wilcoxon	1511,500
Z	-5,085
Sig. asintótica (bilateral)	,000

a. Variable de agrupación: Grupo

Como se puede apreciar en la tabla 4 y 5 tanto para el resultado total que se obtiene de los puntajes de ambas pruebas [$Z = -5,08$, $p = 0,00$] como para los puntajes totales de cada prueba por separado: Scratch [$Z = -3,77$, $p = 0,00$] y LEGO [$Z = -5,27$, $p = 0,00$], evidencian un efecto significativo del uso sostenido de Scratch con respecto al grupo control. Dejando en evidencia que las diferencias que se van a presentar a continuación se obtienen desde lo más general hasta lo más específico de los resultados de este estudio.

Tabla 5

Análisis de Mann-Whitney para la media total de la prueba de Scratch y LEGO por separado

Estadísticos de prueba^a		
	MeTotalLe	MeTotalScr
	g	a
U de Mann-Whitney	770,500	537,500
W de Wilcoxon	1716,500	1483,500
Z	-3,779	-5,273
Sig. asintótica (bilateral)	,000	,000

a. Variable de agrupación: Grupo

En cuanto los resultados de este análisis no paramétrico en los factores obtenidos para la prueba de Scrtach, estos indican que hay diferencias significativas entre el grupo experimental y el grupo control. En otras palabras, se puede decir que hubo un efecto significativo del uso sostenido de Scratch para los estudiantes del grupo experimental con respecto a los del grupo control, en cuanto al factor 1 que agrupa los ejercicios de secuencias, eventos, paralelismos [$Z = -5,92$, $p = 0,00$], el factor 2 que agrupa a condicional, operadores, datos [$Z = -3,51$, $p = 0,00$] y también con respecto al ejercicio de loops, que corresponde al último factor [$Z = -2,41$, $p = 0,01$] (ver Tabla 6).

Tabla 6

Análisis de Mann-Whitney para los factores de la prueba de Scratch y LEGO

	Estadísticos de contraste ^a				
	Factor 1 Lego	Factor 2 Lego	Factor 1 Scratch	Factor 2 Scratch	Factor 3 Scratch
U de Mann-Whitney	1323,000	474,000	490,000	896,000	1141,000
W de Wilcoxon	3339,000	1420,000	1571,000	1977,000	2222,000
Z	-,209	-5,726	-5,926	-3,516	-2,410
Sig. asintót. (bilateral)	,834	,000	,000	,000	,016

a. Variable de agrupación: Grupo

Por otro lado, en cuanto a la prueba de LEGO se destaca que no hay diferencias significativas entre el grupo experimental y el grupo control con respecto al primer factor que agrupa los ejercicios de secuencias, eventos y loops [$Z = -0,20$, $p = 0,83$], lo cual indica que estos tres ejercicios se pueden desarrollar en LEGO sin que los estudiantes hayan manejado previamente Scratch. En cuanto al segundo factor, si hay diferencias significativas entre ambos grupos, de manera que los estudiantes que usaron previamente

Scratch tuvieron mejores puntajes en los ejercicios de paralelismos, condicional, operadores y datos a diferencia de aquellos que no han usado el programa [$Z = -5,72$, $p = 0,00$].

Tabla 7

Rangos promedio de los factores de las pruebas de Scratch y LEGO

Rangos				
	Grupo	N	Rango promedio	Suma de rangos
Factor 1 de Lego	0	43	54,23	2332,00
	1	63	53,00	3339,00
	Total	106		
Factor 2 de Lego	0	43	33,02	1420,00
	1	63	67,48	4251,00
	Total	106		
Factor 1 de Scratch	0	46	34,15	1571,00
	1	63	70,22	4424,00
	Total	109		
Factor 2 de Scratch	0	46	42,98	1977,00
	1	63	63,78	4018,00
	Total	109		
Factor 3 de Scratch	0	46	48,30	2222,00
	1	63	59,89	3773,00
	Total	109		

Finalmente, a partir de la Tabla 7, que muestra los rangos promedio de cada factor con respecto al grupo experimental y control, se puede destacar que los puntajes para los tres factores de la prueba en Scratch son más altos para el grupo experimental, al igual que el segundo factor de la prueba de LEGO. El primer factor de LEGO tiene puntajes muy semejantes entre ambos grupos, lo cual indica que no hay diferencias en la ejecución de los ejercicios de secuencias, eventos y loops por parte de los estudiantes que han o no usado previamente Scratch.

Las tres gráficas que se presentan a continuación reflejan los puntajes promedio entre el grupo experimental y el grupo control con respecto a los tres factores de Scratch. En estas se puede apreciar como los puntajes promedio soportan la afirmación de las diferencias significativas entre los grupos.

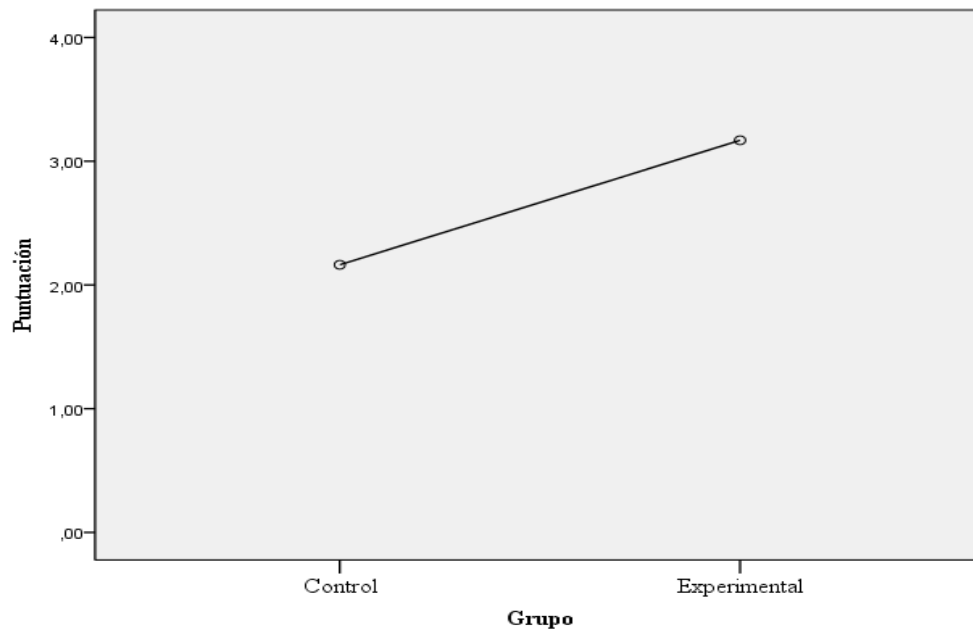


Figura 1. Grupo experimental vs grupo control –Factor 1 Scratch

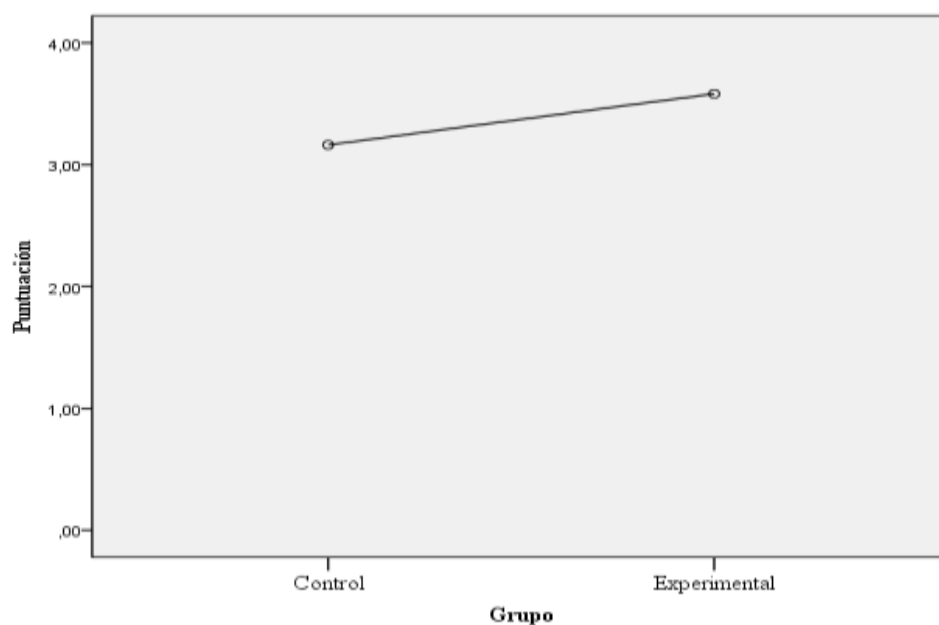


Figura 2. Grupo experimental vs grupo control –Factor 2 Scratch

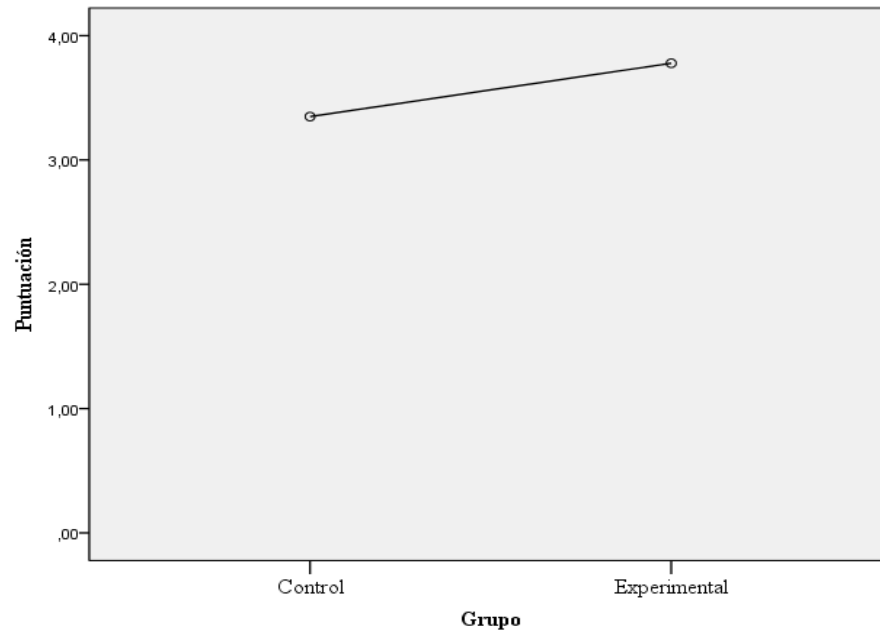


Figura 3. Grupo experimental vs grupo control –Factor 3 Scratch

Las dos siguientes gráficas representan los dos factores para LEGO y el comportamiento de los puntajes promedio correspondientes para ambos grupos de estudio. Como se puede observar para el factor 1 de LEGO, los puntajes son homogéneos en ambos grupos mientras que para el segundo factor es evidente como los puntajes son superiores en el grupo experimental.

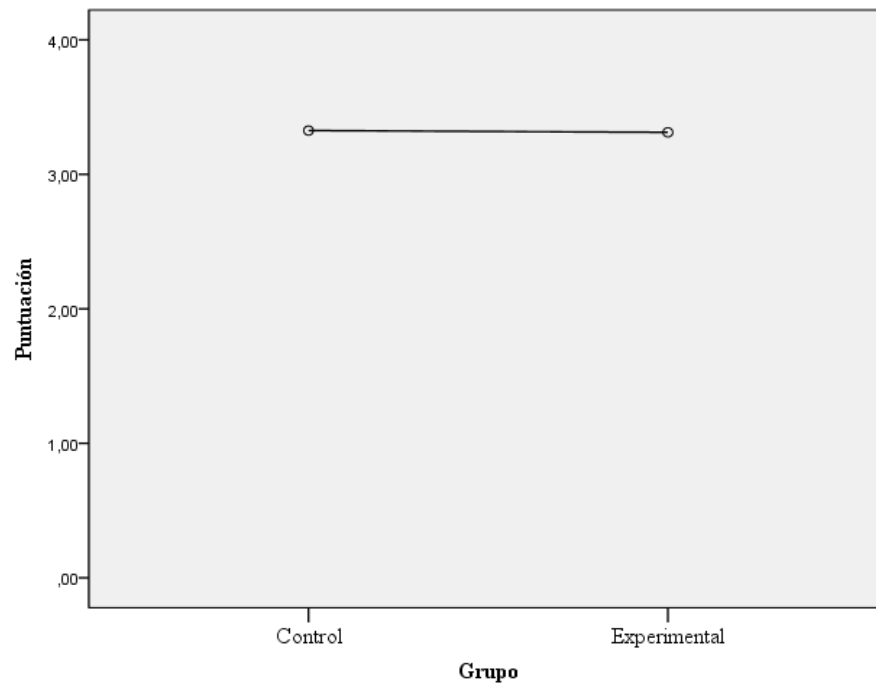


Figura 4. Grupo experimental vs grupo control –Factor 1 LEGO

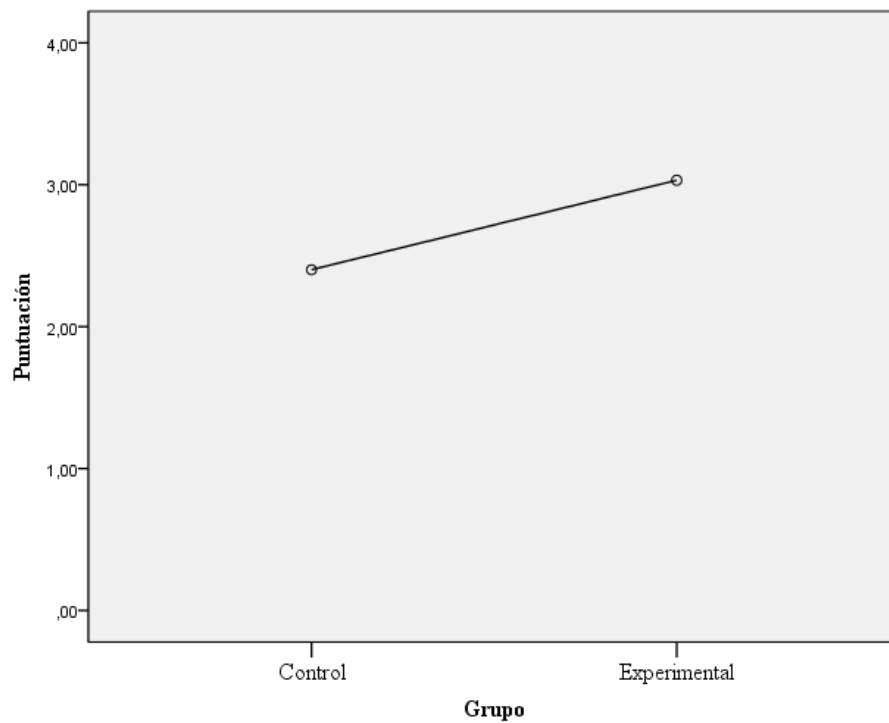


Figura 5. Grupo experimental vs grupo control –Factor 2 LEGO

Encuesta Sobre Uso y Acceso a Tecnologías y Conocimiento de Programas

Todos los estudiantes que participaron en esta investigación realizaron una encuesta que indagó sobre la disposición de recursos tecnológicos en casa, sobre el uso que le daban a esos recursos y finalmente sobre el conocimiento que tenían sobre ciertos programas a nivel educativo. El objetivo de esta encuesta fue el de (además de recoger información demográfica de los participantes) obtener una caracterización sobre variables de acceso y uso de tecnologías para ver en qué medida influían en los resultados de las pruebas sobre los conceptos computacionales. Para efectos de realizar un análisis de las relaciones que hay entre los aspectos de la encuesta y los grupos asignados en la investigación, se decidió agrupar los totales de cada sección de ítems de la siguiente manera.

Tabla 8

Agrupación de variables de la encuesta de uso, acceso y conocimiento de programas

Categoría que agrupa	Ítems agrupados	Observaciones
Acceso a tecnologías	Desktop PC, laptop, Tablet, internet, smartphone	
Uso dado a las tecnologías	Videojuegos, navegar en internet, hacer tareas, escuchar música, leer, revisar el correo electrónico, ver películas.	Todas las respuestas eran de sí o no.
Programas que conocen	Paint, Power point , Word, Excel, LOGO, Photoshop	

Una vez se calcularon estos totales por variable de agrupación se hizo un análisis mediante la prueba de muestras independientes Mann-Whitney. Esta prueba permitió establecer que si hay diferencias significativas entre grupos en el acceso a tecnologías [$Z = -3,37$, $p=0,01$] y los programas que conocen [$Z = -2,30$, $p=0,02$], lo cual puede estar en consonancia con el hecho que el grupo control correspondía a un colegio que, además de contar con mejores recursos tecnológicos también le facilita a los estudiantes variedad de programas relacionados con diseño y electrónica. Por otro lado, los estudiantes del grupo control reportan tener mayor disposición de dispositivos electrónicos en casa lo cual puede estar relacionado con diferencias en los niveles socioeconómicos de la zona de donde son los participantes, que es la misma en donde está ubicado el colegio en donde se llevó a cabo la toma de datos. El grupo control corresponde a la localidad de Engativá en donde el 75% de la población son de estrato 3 (Secretaría de Hábitat, 2011) mientras que el grupo experimental corresponde a la localidad de Kennedy en donde el 61% de la población no supera el estrato 2 (Observatorio de Culturas, 2008).

Tabla 9

Análisis de Mann-Whitney para las variables agrupadas de la encuesta.

Estadísticos de prueba^a			
	Uso de tecnologías	Acceso a tecnologías	Programas que conocen
U de Mann-Whitney	1143,500	841,000	1024,000
W de Wilcoxon	3159,500	2857,000	3040,000
Z	-1,388	-3,377	-2,303
Sig. asintótica (bilateral)	,165	,001	,021

a. Variable de agrupación: Grupo

En cuanto al uso de la tecnología se puede observar que no hay diferencias significativas [$Z = -1,38$, $p = 0,16$]. En este caso esto puede estar orientado a que, a pesar de contar con los recursos tecnológicos, los estudiantes del grupo control usan los computadores, tabletas y laptops para casi lo mismo que los estudiantes del grupo control. Como se puede observar en las siguientes graficas las diferencias de medias son más marcadas para el acceso a tecnologías y programas que conocen (Figura 6 y 8) mientras que para el uso dado a las tecnologías las medias son muy cercanas (Figura 7).

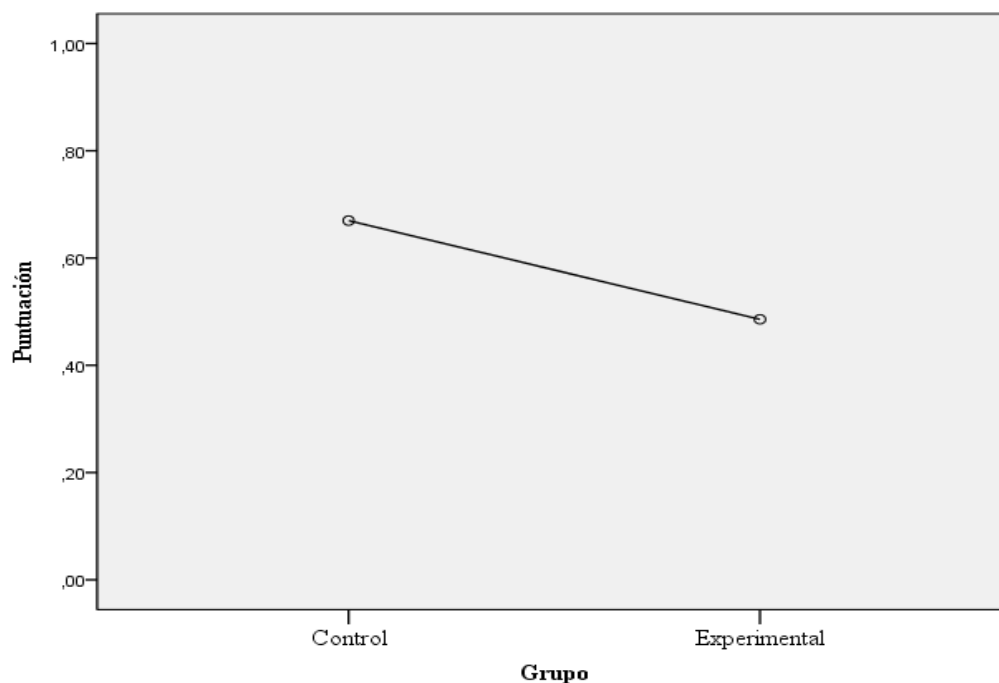


Figura 6. Grupo experimental vs grupo control –Acceso a tecnologías

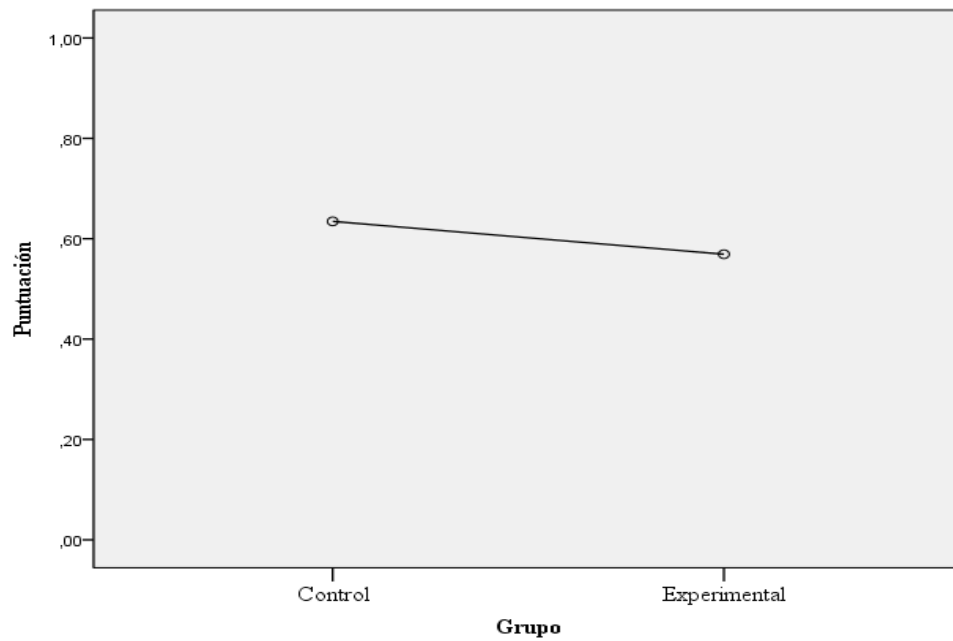


Figura 7. Grupo experimental vs grupo control –Uso dado a las tecnologías.

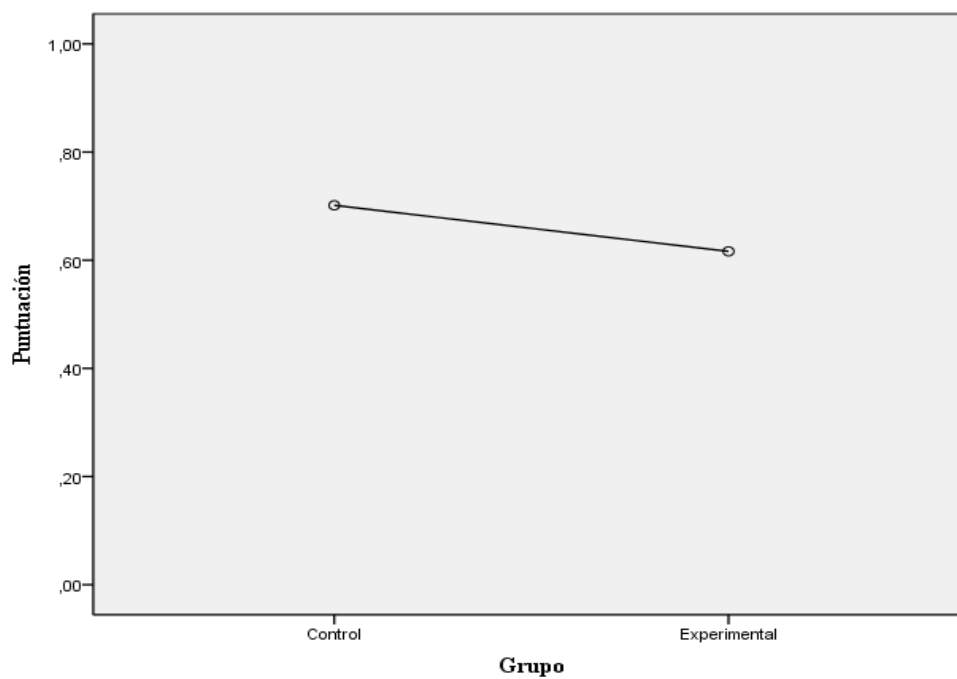


Figura 8. Grupo experimental vs grupo control –Programas que conocen.

No obstante, para evaluar si efectivamente estas variables están influyendo en los resultados de la prueba en los grupos experimental y control se hizo un análisis mediante un modelo lineal univariante. Este permitió evidenciar que las variables de uso de tecnologías, acceso a las tecnologías y programas que conocen los estudiantes no están explicando la variación observada mediante los puntajes totales de la prueba ($F=0.88$, $p > 0.05$).

Tabla 10

Análisis mediante el modelo lineal univariante entre el total general de ambas pruebas y las covariables.

Pruebas de efectos inter-sujetos					
Variable dependiente: Promedio total de la prueba					
Origen	Tipo III de suma de cuadrados	gl	Media cuadrática	F	Sig.
Modelo corregido	7,653 ^a	4	1,913	9,927	,000
Intersección	34,717	1	34,717	180,122	,000
Acceso a tecnologías	,013	1	,013	,069	,793
Uso de tecnologías	,009	1	,009	,048	,828
Programas que conocen	,715	1	,715	3,711	,057
Grupo	5,331	1	5,331	27,660	,000
Error	19,467	101	,193		
Total	1035,444	106			
Total corregido	27,120	105			

a. R al cuadrado = ,282 (R al cuadrado ajustada = ,254)

De igual manera tampoco se observan estadísticos con niveles críticos para la prueba de Scratch ($F=0.40$, $p > 0.05$) ni la de LEGO ($F=0.33$, $p > 0.05$), por lo cual se puede afirmar que la relación entre las covariables y cada una de las pruebas no está afectado o alterando

los resultados del grupo experimental y el control. El valor de R al cuadrado indica que los tres efectos incluidos en el modelo (Acceso a tecnologías, Uso de tecnologías y programas que conocen) están explicando un 30% en el caso de la prueba de Scratch, un 15% para la prueba de LEGO y un 28% en cuanto al total de la aplicación de ambas pruebas lo cual es muy bajo.

Tabla 11

Análisis mediante el modelo lineal univariante entre el total de la prueba de Scratch y las covariables.

Pruebas de efectos inter-sujetos					
Variable dependiente: Promedio total de la prueba de Scratch					
Origen	Tipo III de suma de cuadrados	gl	Media cuadrática	F	Sig.
Modelo corregido	13,718 ^a	4	3,430	10,998	,000
Intersección	31,180	1	31,180	99,992	,000
Acceso a tecnologías	,424	1	,424	1,358	,247
Uso de tecnologías	,002	1	,002	,006	,939
Programas que conocen	1,305	1	1,305	4,184	,043
Grupo	9,992	1	9,992	32,045	,000
Error	31,494	101	,312		
Total	1103,939	106			
Total corregido	45,212	105			

a. R al cuadrado = ,303 (R al cuadrado ajustada = ,276)

En la tabla 11 y 12 se expresan los resultados del análisis lineal univariante en donde se puede apreciar cómo cada una de las categorías obtenidas a través de la encuesta no explican el efecto de las pruebas realizadas para medir los conceptos computacionales, lo

cual está en consonancia con los resultados obtenidos con el análisis no paramétrico de Mann-Whitney. De esta manera las ventajas en acceso, uso y conocimiento en programas que favorecen al grupo control no son las mismas que para el grupo experimental, por lo cual las diferencias en las pruebas realizadas son atribuibles al uso o no de Scratch.

Tabla 12

Análisis mediante el modelo lineal univariante entre el total de la prueba en LEGO y las covariables.

Pruebas de efectos inter-sujetos					
Variable dependiente: Promedio total de la prueba de LEGO					
Origen	Tipo III de suma de cuadrados	gl	Media cuadrática	F	Sig.
Modelo corregido	3,734 ^a	4	,933	4,644	,002
Intersección	38,444	1	38,444	191,272	,000
Acceso a tecnologías	,176	1	,176	,875	,352
Uso de tecnologías	,055	1	,055	,273	,603
Programas que conocen	,302	1	,302	1,500	,223
Grupo	2,122	1	2,122	10,559	,002
Error	20,300	101	,201		
Total	983,184	106			
Total corregido	24,034	105			

a. R al cuadrado = ,155 (R al cuadrado ajustada = ,122)

Análisis de Correlación Entre la Prueba de Scratch y la de LEGO

Para poder establecer si la prueba de LEGO funcionó como medida de *transfer* de los *conceptos computacionales* entrenados en Scratch a otro dominio isomorfo, se realizó un análisis de correlación de Spearman el cual indica que hay una relación significativa,

moderada y directamente proporcional entre el desempeño en la prueba de Scratch y la prueba de LEGO ($R_S=0,55$, $p < 0.01$). Lo anterior indica que la prueba de LEGO cumple su función como herramienta de medida del *transfer* de los *conceptos computacionales* desarrollados en un dominio de programación a un dominio físico que permite abstraer el tipo de interfaz, bloques y elementos lógicos.

Tabla 13

Correlación de Spearman entre la prueba de Scratch y la prueba de LEGO

Correlaciones				
			Promedio total en LEGO	Promedio total en Scratch
Rho de Spearman	Promedio total en LEGO	Coefficiente de correlación	1,000	,555**
		Sig. (bilateral)	.	,000
		N	106	106
	Promedio total en Scratch	Coefficiente de correlación	,555**	1,000
		Sig. (bilateral)	,000	.
		N	106	106

**. La correlación es significativa en el nivel 0,01 (bilateral).

Los resultados descritos anteriormente dan cuenta de la evaluación del desarrollo de los conceptos computacionales a partir de dos instrumentos. Éstos evidencian que, además de haber un mejor desempeño en el uso de los conceptos computacionales para los estudiantes quienes han recibido formación con Scratch, también hay un patrón en la forma en que se agrupan dichos conceptos. Se agrupan de tal manera que reflejan aspectos que inciden en el aprendizaje de los mismos. Además, no se deja lugar a posibles variables de acceso y uso de tecnologías que generen ruido en el desempeño de los estudiantes en ambas pruebas. Finalmente se establece que la relación entre la prueba de Scratch y la de LEGO es

suficiente como para sustentar el transfer de los conceptos computacionales a través de una tarea isomorfa.

Discusión

Los resultados de este estudio sugieren que el uso de la herramienta de programación Scratch no solo fomenta el desarrollo de los conceptos computacionales descritos por Brennan y Resnick (2012) sino que también dejan ver que, la adquisición de dichos conceptos esta cernida a procesos de aprendizaje en los cuales incide tanto la formación continuada con el programa, como aspectos específicos de los conceptos de programación. De tal manera, por un lado, se destaca que los participantes del grupo experimental al estar involucrados en actividades de programación desde el curso octavo con Scratch, gradualmente adquirieron elementos importantes de pensamiento computacional y, por lo tanto, mostraron una mejor ejecución en los ejercicios planteados y a su vez en el manejo de los conceptos. Por el otro lado, los conceptos computacionales abordados se agrupan de tal manera que reflejan no solo grados de dificultad, sino aspectos de abstracción e integración de información que no se podrían apreciar en un usuario novato.

Para comenzar, los resultados de las diferencias en los totales de la aplicación general, de la prueba de Scratch y la de LEGO por separado, indican que el desempeño de los estudiantes está significativamente asociado al uso de Scratch. En otras palabras, los conceptos computacionales que se plasmaron en los ejercicios de estas pruebas se han ido adquiriendo por el uso sostenido en el que se han involucrado los estudiantes del grupo experimental. Además, estas dos pruebas muestran cómo Scratch desarrolla las habilidades de pensamiento que fomentan la adquisición de los conceptos computacionales y también permiten evidenciar cómo se agrupan dichos conceptos. En este sentido se puede decir que esta agrupación está asociada a la dificultad que implican unos ejercicios sobre otros. Sin

embargo, también se puede afirmar que los conceptos de secuencias, evento, loops y paralelismos corresponden a una unidad conceptual que no se ha documentado, en donde estos primeros se dan por cadenas de acción de carácter lineal mientras que los conceptos de condicional, operadores y datos son nociones que implican estructuras con múltiples caminos.

No obstante queda en evidencia que no hay diferencias significativas en la prueba de LEGO para los tres primeros ejercicios, lo cual indica que éstos pueden ser desarrollados sin tener un precedente específico con la interfaz de Scratch, que en cierta medida se podría explicar porque la solución para estos ejercicios en LEGO reside en la modificación del primer ejercicio cambiando uno o dos bloques al igual que en la prueba de Scratch. Aun así, en este caso las guías de correspondencias limitan la cantidad de opciones que pueden ser distractores en el caso de Scratch. Además de esto se puede decir que en el caso de LEGO, es más sencillo para los estudiantes demostrar que son capaces de entender conceptos que van desde seguir una serie de pasos para que algo suceda (secuencias), pasando por identificar que hay acciones o situaciones que tienen causalidad sobre otras (eventos) y finalmente que hay formas de hacer que algo se repita las veces que se desee sin volver a hacer la programación desde el principio una y otra vez (loops o bucles). Lo cual podría indicar que este tipo de conceptos se pueden adquirir bajo otro tipo de herramientas o contextos cotidianos que no se indagaron en esta investigación. Mientras que, los conceptos de programación que implican un mayor nivel de abstracción y trabajo en la interfaz de Scratch (paralelismos, condicionales, operadores y datos), si reflejan un mejor desempeño por parte de los estudiantes del grupo experimental, dando a entender que aunque se evalúen estos conceptos con la prueba análoga en LEGO los resultados serán mejores para aquellos que lleven determinado tiempo involucrados con Scratch.

En cuanto a los resultados obtenidos por la encuesta de uso, acceso y conocimiento de programas en relación al grupo experimental y control, la cual se usó para determinar si estas variables de control estaban influyendo en el desempeño mostrado por los participantes en ambas pruebas, se evidenciaron diferencias significativas en acceso y programas que conocen los estudiantes pero no en el uso que le dan a las TIC. Los estudiantes del grupo experimental mostraron tener más acceso a recursos tecnológicos (computador de escritorio, laptop, Tablet, contar con internet y Smartphone) y conocimiento de software (paquete de office, LOGO, Photoshop y otros) que los estudiantes del grupo control. Esto se puede atribuir a la diferencia que hay entre niveles socioeconómicos de la zona donde viven y donde se ubica el colegio donde las diferencias de acceso a tecnologías dependen de los ingresos que se calculan por localidad (Secretaría de Planeación Distrital, 2013). Por otro lado, el uso dado a las tecnologías por los estudiantes en ambas instituciones es homogéneo, lo cual es congruente con el hecho de que las localidades de menores ingresos usan el computador para estudiar casi en la misma proporción que las localidades de altos ingresos (Secretaría de Planeación Distrital, 2013) y además esta igualdad en el uso también se podría atribuir a tendencias acordes a la edad y el género (OECD, 2015). Por otro lado vale la pena destacar que el colegio del grupo control, como ya se mencionó previamente, es un colegio técnico, por lo cual cuenta con una mejor infraestructura tecnológica y además los estudiantes pueden capacitarse en temas de electrónica, mecánica y diseño gráfico.

Ahora bien, estos resultados permiten visibilizar que la infraestructura tecnológica, los programas que conocen los estudiantes y los usos que dan a las tecnologías no necesariamente son determinantes para el desarrollo de los conceptos computacionales. Así, aunque el colegio del grupo experimental cuenta con cierta desventaja tecnológica, ha

logrado instaurar el uso de Scratch como una herramienta de aprendizaje que incluso los ha posicionado por fuera de las aulas, llevándolos a participar en el III Premio Scratch 2014 (Cali) y gran parte de este proceso de formación depende del acompañamiento docente y la instrucción continuada en Scratch desde el grado octavo hasta el grado once. De hecho los resultados permiten establecer que hay ciertos conceptos computacionales que se pueden desarrollar sin necesidad de recibir capacitación en plataformas de programación o contar con los mejores recursos tecnológicos. Este desempeño en conceptos de secuencias, eventos y loops puede estar relacionado a que se presentan y se pueden identificar con mayor frecuencia en otros dominios, como la escritura, los deportes, otras disciplinas e incluso en los videojuegos.

Por otro lado, si bien es cierto que hay diferencias significativas en el acceso y uso que los estudiantes de ambos colegios dan a las tecnologías, los resultados también indican que esas diferencias no están explicando el efecto que se da por las pruebas de Scratch y LEGO. Es decir que la ejecución de los participantes en ambas pruebas no está siendo influenciada por las condiciones de infraestructura tecnológica ni mucho menos por los programas que manejan. De tal manera, se puede asumir que estas pruebas cumplen su objetivo de evaluar el desarrollo de los conceptos computacionales descritos tanto en una condición de programación con la interfaz de Scratch como en una condición isomorfa con bloques de LEGO.

Con respecto a si la prueba de LEGO constituyó un modelo análogo para considerar el *transfer* de un dominio de programación en computadores a un dominio físico, el cual representa las condiciones de la interfaz de Scratch, como los elementos lógicos que emplea en sus bloques de construcción y a su vez los conceptos inherentes asignados a dichos bloques; los resultados indican que hay una relación directa entre la prueba de Scratch y la

prueba de LEGO. La segunda prueba, que empleó las fichas de LEGO como elementos isomorfos, sirvió como medida para resaltar el *transfer* reflexivo o *low transfer* (Perkins & Salomon, 1989, 1990) que se da sobre todo en el grupo experimental. Las características previas del grupo experimental indican que el conocimiento sobre el programa y el uso de sus bloques se han replicado hasta cierto punto, permitiendo que al presentarse un estímulo semejante al de la interfaz de Scratch les fuera posible desarrollar de manera eficaz los ejercicios en LEGO.

Al considerar las condiciones que de acuerdo a la literatura suelen acompañar un proceso de *transfer*, se podría destacar que, en este caso, los estudiantes del grupo experimental ya habían tenido una *práctica profunda y variada* con Scratch mediante talleres de programación en la clase de tecnología. Además se asume que los participantes realizaron una *abstracción explícita* del entorno novedoso en LEGO, el cual se presentó como una *metáfora o analogía* de lo que habían desarrollado en Scratch previamente (Perkins & Salomon, 1992). Otras condiciones que se mencionan y pueden ser relevantes no son medidas en esta investigación como lo son el *auto monitoreo* y la *conciencia despierta* ya que su medición es compleja y se escapa del alcance de este estudio (Perkins & Salomon, 1992). De hecho estos últimos elementos mencionados se asocian con el tipo de *transfer* que en la mayoría de entornos educativos formales se quiere alcanzar y es el *transfer* consciente o *high transfer* (Molina, 2002). Este tipo de *transfer* precisa de sujetos entregados a una dinámica de abstracción que trasciende el contexto del ejercicio y el programa como tal, proporcionándoles la posibilidad de ver las relaciones de lo aprendido en un entorno de programación para resolver problemas en situaciones cotidianas o de otras disciplinas. Este concepto de *transfer* consciente es consistente con el usuario productor activo (Resnick y Monroy, 2008) porque implica a un usuario con la capacidad de transferir

sus destrezas en programación a otros dominios y además compartirlas entre pares, con el fin de aportar a la resolución de problemas que emergen en otros escenarios académicos y/o cotidianos.

Si bien en este estudio no se exploró el tipo de *transfer* consciente que resulta tan relevante para propósitos educativos, se dio un avance considerable en la formulación de una prueba que además de evaluar el desarrollo de los conceptos computacionales descritos por Brennan y Resnick (2012), también logra dar un salto significativo en la identificación de una medida de *transfer* que puede ser el primer paso para alcanzar el *transfer* consciente. Para complementar estos resultados acerca de los conceptos computacionales sería recomendable, en futuros estudios, evaluar el tipo de *transfer* consciente o *High transfer* mediante una prueba en otro lenguaje de programación o una tarea que permita resolver un problema en el contexto de otra área o disciplina. Además sería esencial evaluar las otras dos dimensiones del pensamiento computacional. Así, se podrían hacer entrevistas semiestructuradas a muestras de los participantes con el fin de ahondar en la dimensión de *perspectivas computacionales* con el fin de establecer como el uso de Scratch ha mediado en la forma en que los estudiantes se conciben a sí mismos, a los demás y a la tecnología con la que interactúan. Adicional a esto, para identificar cuáles de las *prácticas computacionales* realizan los estudiantes cuando hacen sus proyectos en Scratch, se podría plantear una prueba para poder determinar cuáles de las cuatro prácticas computacionales identificadas por Brennan y Resnick (2012) emplearon para sus proyectos.

Para concluir, cabe destacar que, aunque se requiere de investigaciones que ahonden en las otras dimensiones del pensamiento computacional, los hallazgos del presente estudio indican las potenciales bondades de las pruebas aplicadas para medir y discernir las diferencias entre los conceptos computacionales acotados por Brennan y Resnick (2012).

Tales conceptos, a su vez, cobran importancia en el surgimiento de usuarios productores en la nueva realidad digital, aquellos que no solo manejan muy bien los dispositivos y el software de última generación sino que también conocen su funcionamiento y están dispuestos a desarrollar, extrapolar y compartir su conocimiento en programación en distintas áreas de aprendizaje y de la cotidianidad.

Es importante considerar que el desarrollo de los conceptos computacionales (y las otras dimensiones) no surgen por la sola interacción con el programa. Es esencial un proceso guiado por docentes que entiendan y valoren la importancia que cobra en la actualidad las habilidades en programación y además que se preocupen porque estas habilidades se puedan potenciar y trasladar a otros dominios. Esto implica dinámicas graduales en donde los estudiantes vayan adquiriendo la capacidad de resolver problemas, aplicando la lógica subyacente a la programación en computadores. Sin embargo hay mucho camino por recorrer, ya que, como menciona Goldenson (1996) y Jaramillo (2005), aún persiste una tendencia en las escuelas (probablemente más en las públicas) de abandonar las clases sobre programación y enfocarse en la enseñanza de temas que tienen que ver con procesadores de palabras, hojas de cálculo, bases de datos y software por el estilo.

Referencias

- Brandt, A. M., & Colton, M. B. (2008). Toys in the classroom: LEGO MindStorms as an educational haptics platform. *IEEE*, 389-395. Doi: 10.1109/HAPTICS.2008.4479982.
- Brennan, K., and Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the American Educational Research Association (AERA) annual conference*.
- Burke, Q., and Kafai, Y.B. (2012). The writers' workshop for youth programmers: Digital storytelling with Scratch in middle school classrooms. *Proceedings of the 43rd ACM technical symposium on Computer Science Education, ACM 2012*.
- Carnegie Mellon University. (2014). Alice: An Educational Software that teaches students computer programming in a 3D environment. Recuperado de <http://www.alice.org/index.php>
- Choi, B., Jung, J. & Baek, Y. (2013). In what way can technology enhance student learning? : A preliminary study of technology supported learning in mathematics. In R. McBride & M. Searson (Eds.), *Proceedings of Society for Information Technology & Teacher Education International Conference*, Chesapeake, VA: AACE. 3-9.
- Clements, D. H., and Sarama, J. (1997). Research on Logo: A Decade of Progress. Co-published simultaneously in *Computers in Schools* (The Haworth Press, Inc.), 14, (1/2), 9-46.
- Corporación Colombia Digital. (2014). Una mirada a la juventud colombiana en el siglo XXI. Recuperado de <https://www.colombiadigital.net/opinion/columnistas/conexion/item/4947-una-mirada-a-la-juventud-colombiana-en-el-siglo-xxi.html#a3>
- DANE. (2015). Boletín Técnico: Indicadores Básicos de Tenencia y Uso de Tecnologías de la Información y Comunicación –TIC en Hogares y Personas de 5 y más años de edad 2014. Bogotá D.C.
- Fields, D. A., Giang, M. & Kafai, Y. B. (2013). Understanding collaborative practices in the Scratch online community: Patterns of participation among youth designers. In N. Rummel, M. Kapur, M. Nathan, & S. Puntambekar (Eds), *CSCL 2013 Conference Proceedings, International Society of the Learning Sciences: Madison, WI*. 1, 200-207.
- García, F., Portillo, J., Romo, J., & Benito, M. (2007). Nativos digitales y modelos de aprendizaje. In SPDECE. Recuperado de: <http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-318/Garcia.pdf>

- Gee, James Paul. (2008) "Learning and Games." *The Ecology of Games: Connecting Youth, Games, and Learning*. Edited by Katie Salen. The John D. and Catherine T. MacArthur Foundation Series on Digital Media and Learning. Cambridge, MA: The MIT Press, 21–40. Doi: 10.1162/dmal.9780262693646.021
- Goldenson, D. (1996). *Why Teach Computer Programming? Some Evidence About Generalization and Transfer*. Lecture, Minneapolis.
- Grover, S. y Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, 42 (1), 38-43. DOI: 10.3102/0013189X12463051
- Ipsos-Napoléon Franco. (2010). Primer gran estudio continuo de Ipsos-Napoléon Franco sobre el nivel de digitalización de los colombianos y cómo las nuevas tecnologías están impactando sus vidas. Recuperado de <http://www.slideshare.net/DiegoMolanoVega/encuesta-de-consumo-digital>
- Ipsos-Napoléon Franco. (2012). Percepción, usos y hábitos frente a las Tecnologías de la Información y la Comunicación. Recuperado de <http://www.slideshare.net/alfreakm/percepcin-usos-y-hbitos-frente-a-las-tecnologas-de-la-informacin-y-la-comunicacin>.
- Jaramillo, P. (2005). USO DE TECNOLOGÍAS DE INFORMACIÓN EN EL AULA. ¿QUÉ SABEN HACER LOS NIÑOS CON LOS COMPUTADORES Y LA INFORMACIÓN? *Revista de Estudios Sociales*. 20, 27-44.
- Jenkins, H. (2006). *Convergence Culture: la cultura de la convergencia de los medios de comunicación*. Paidós. Barcelona.
- Jenkins, H. Clinton, K., Purushotma, R., Robison, A., & Weigel, M. (2006). "Confronting the challenges of participation culture: Media education for the 21st century." White Paper. Chicago, IL: The John D. and Catherine T. MacArthur Foundation.
- Kazimoglu, C., Kiernan, M., Bacon, L y Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. *Elsevier*. 47 (2012), 1991-1999.
- Koh, K. (2013). Adolescents' information-creating behavior embedded in digital media practice using Scratch. *Journal of the American Society for Information Science and Technology*. 64(9), 1826-1841.
- Lee, I., Martin, F., Denner, B., Coulter, B., Allan, W., Erickson, J. et al. (2011). Computational Thinking for Youth in Practice. *Standard Articles*, 2 (1), 32-37.

- López, A. M. I., & de la Llata Gómez, D. E. (2010). Niños nativos digitales en la sociedad del conocimiento: acercamientos conceptuales a sus competencias. *Razón y palabra*, 72, 26-24.
- Lu, J. J., & Fletcher, G. H. (2009, March). Thinking about computational thinking. In *ACM SIGCSE Bulletin*, 41, (1), 260-264. Doi: 10.1145/1508865.1508959
- Malan, D.J., & Leitner, H.H. (2007). Scratch for budding computer scientists. *ACM SIGCSE Bulletin*, 39, 223–227.
- Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004). Scratch: a sneak preview [education]. In *Creating, Connecting and Collaborating through Computing Proceedings. Second International Conference on IEEE*, 104-109.
- Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. (2010). The scratch programming language and environment. *ACM Trans. Comput. Educ*, 10, 4. DOI = 10.1145/1868358.1868363. <http://doi.acm.org/10.1145/1868358.1868363>.
- Martín, J. A. J. (2010). La era digital: nuevos medios, nuevos usuarios y nuevos profesionales. *Razón y palabra*, 15 (71).
- Mayerová, K. (2012). Pilot activities: LEGO WeDo at primary school. In *3rd International Workshop, Teaching Robotics, Teaching with Robotics*. 32-39.
- Ministerio de Tecnologías de la Información y las Comunicación. (2014). Estadísticas de formación en TIC. Recuperado de <http://goo.gl/Hg2fOu>
- Molina, E.C. (2002). El Proceso del Transfer: Revisión y Nuevas Perspectivas. *EduPsykhé*, 1 (1), 69-95.
- Morales, & Morales, R. (2014). Lenguajes de programación: ¿qué son y para qué sirven? *Colombiadigital.net*. Recuperado el 7 de junio de 2016 desde: <https://colombiadigital.net/actualidad/articulos-informativos/item/7669-lenguajes-de-programacion-que-son-y-para-que-sirven.html>
- OECD (2015). How Students' Use of Computers has Evolved in Recent Years. PISA and OECD (Eds), *Students, Computers and Learning: Making the Connection* (pp. 49-73). OECD PUBLISHING. Doi: <http://dx.doi.org/10.1787/9789264239555-en>
- Olabe, X. B., Basogain, M. Á. O., & Basogain, J. C. O. (2015). Pensamiento Computacional a través de la Programación: Paradigma de Aprendizaje. *Revista de Educación a Distancia*, 46(6), 1-33. Doi: 10.6018/red/46/6
- Pea, R.D., & Kurland, M.D. (1984). On the cognitive effects of learning computer programming. *New Ideas Psychol*, 2 (2), 137-168.

- Peppler, K., & Kafai, Y. (2005). Creative coding: The role of art and programming in the K-12 educational context.
- Perkins, D. N., & Salomon, G. (1992). Transfer of learning: Contribution to the International Encyclopedia of Education. Oxford, England: Pergamon Press. Retrieved, 3(3), 07.
- Repenning, A. (2014). AgentSheets. Recuperado de <http://www.agentsheets.com/index.html>
- Resnick, M. (2012). Mother's Day, warrior cats, and Digital Fluency: Stories from the Scratch Online Community. Proceedings of the Constructionism 2012 conference. Athens, Greece.
- Resnick, M., Kafai, Y., y Maeda, J. (2003-2007). A Networked, Media-Rich Programming Environment to Enhance Technological Fluency at After-School Centers in Economically-Disadvantaged Communities. Information Technology Research, 1-14.
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K. et al. (2009). Scratch: Programming for All. Communications of the acm, 52(11), 60-67. Doi:10.1145/1592761.1592779.
- Ruiz, J. (1994). Implicaciones educativas del lenguaje LOGO. Comunicación, Lenguaje y Educación, 21,111-118.
- ScratchEd team. (2011). Creative computing: a desing-based introduction to computational thinking. Draft-20110923.
- Secretaría de Cultura, Recreación y Deportes: Observatorio de Culturas. (2008). Localidad de Kennedy: Ficha Básica. Bogotá. D.C: Autor.
- Secretaría Distrital de Planeación. (2011). Diagnostico Localidad de Engativá: Sector Hábitat. Bogotá. D.C: Autor.
- Secretaria Distrital de Planeación. (2013). Boletín No.52: TIC (Tecnologías de la Información y de las Comunicaciones) en Bogotá, D.C. Bogotá, D.C.
- Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition. University of Southampton. Recuperado de: <http://eprints.soton.ac.uk/356481/>
- Werner, L., Denner, J. y Campe, S. (2012). The Fairy Performance Assessment: Measuring Computational Thinking in Middle School. SIGCSE'12.
- Wilson, C., Sudol, L., Stephenson, C., and Stehlik, M. (2010). Running on Empty: The Failure to Teach K-12 Computer Science in the Digital Age. ACM. Recuperado de: <http://www.acm.org/runningonempty/fullreport.pdf>.

Yadav, A., Mayfield, C., Zhou, N., Hambruch, S., and Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Trans. Comput. Educ.* 14, 1. DOI:<http://dx.doi.org/10.1145/2576872>

Anexos

ANEXO 1

Prueba en Scratch TPC

Estimado participante siga las siguientes instrucciones para realizar los ejercicios que se encuentran en la parte inferior de esta hoja:

Antes:

- 1. Escuche las instrucciones por parte del aplicador de la prueba.**

Durante:

- 1. Por favor una vez inicie con los ejercicios anote la hora de inicio que diga el aplicador en esta misma hoja en la parte final.**
- 2. Lea cuidadosamente el ejercicio**
- 3. Desarrollé los ejercicios**
- 4. Una vez finalice CADA UNO de los ejercicios guardé su respuesta así:
Ejemplo: Archivo/Guardar como/ (Nombre del ejercicio y las iniciales de su nombre completo) SecuenciasLASR y guárdelo en la carpeta que dice Scratch en el escritorio del computador. OJO: Verifique en la carpeta en cuestión que se estén guardando correctamente sus ejercicios**
- 5. Si no entiende algún ejercicio o no sabe cómo hacerlo guarde lo que haya hecho como se indica en el paso 4.**
- 6. Si tiene alguna duda pregúntele al aplicador de la prueba.**

Después:

- 1. Una vez termine todos los ejercicios anúnciele al aplicador y este verificará que todo esté completo.**
- 2. Anote en esta misma hoja la hora de finalización de todos los ejercicios.**

Ahora comience la prueba:

Secuencias:

Haz que al presionar la bandera verde, el gato Scratch se mueva 50 pasos, espere un segundo y gire 180 grados, por último haz que diga: Hola!

Evento:

Haz que, al hacer click sobre el gato Scratch este se mueva 50 pasos, espere un segundo, gire 180 grados y por último haz que diga: Soy Scratch!

Loops:

Haz que al presionar la bandera verde el gato Scratch repita 5 veces estos movimientos: que se mueva 50 pasos, espere un segundo y gire 180 grados, por último haz que diga: Soy Scratch!

Paralelismos:

Haz dos bloques grandes siguiendo estos pasos:

Haz un primer bloque en donde al hacer click en la bandera verde el gato se mueva 100 pasos y diga: “Somos Scratch!” durante dos segundos y detén el programa.

En un segundo bloque aparte haz que inicie también al tocar la bandera verde, pero en este haz que aumente el efecto de un solo gato a varios gatos, como si fuera un mosaico y detén el programa.

Condicional:

Haz que al presionar la bandera verde el gato Scratch repita su movimiento hasta que se presione la tecla de espacio. En este caso haz que el gato se mueva sí y solo sí la tecla “flecha derecha” se presiona y que la cantidad de pasos que dé sea de 5. Por último detén el programa.

Operadores:

Haz que al presionar la bandera verde, el gato Scratch pregunte: ¿Cuántos pasos doy? Seguidamente, haz que el gato se mueva de acuerdo a la respuesta a esa pregunta (que debe ser un número) la cual se suma a otro valor que tú escojas entre 5 y 100, para esto usaras el bloque de operadores de suma.

Datos:

Haz que al presionar la bandera verde, el gato se mueva por siempre a una velocidad de 5 pasos. Para hacer esto debes crear una variable que se llame velocidad, una vez la crees debes poner ese bloque en donde deben ir la cantidad de pasos. Por último debes fijar la velocidad del gato Scratch en 5.

No olvide anotar la hora de finalización.

Gracias por participar

Hora de inicio	Hora de finalización

ANEXO 2

Prueba en Lego TPC

Estimado participante siga las siguientes instrucciones para realizar los ejercicios que se encuentran en la parte inferior de esta hoja:

Antes:

1. Usted recibirá tres cosas aparte de este formato: una guía de equivalencias en lego, una hoja de cálculo que debe marcar con nombre completo, edad, sexo, curso y fecha y por último una bolsita con fichas de lego.
2. Por favor cuente que hayan__ fichas, si están incompletas infórmele al aplicador.
3. Escuche las instrucciones de parte del aplicador

Durante:

7. Por favor una vez inicie con los ejercicios anote la hora de inicio que diga el aplicador en la hoja de cálculo.
8. Lea cuidadosamente el ejercicio.
9. Tenga en cuenta las hojas con las equivalencias de las fichas de lego que le entregue el aplicador para resolver los ejercicios.
10. Desarrollé los ejercicios
11. Una vez finalice CADA UNO de los ejercicios levante la mano para que el aplicador de la prueba se acerque y le tome una foto a su trabajo, además usted deberá explicarle brevemente la secuencia que está representando con las fichas de lego.
12. Si tiene alguna duda pregúntele al aplicador de la prueba.

Después:

1. Una vez finalice todos los ejercicios guarde todas las fichas verificando que estén completas.
2. Ponga la hora de finalización al final de la hoja de cálculo.
3. Anúnciele al aplicador que ha terminado y entréguele: este formato, la guía de equivalencias, la hoja de cálculo y la bolsita con las fichas.

Ahora comience la prueba:

Secuencias:

Haz que al presionar la bandera verde, el gato Scratch se mueva 50 pasos, espere un segundo y gire 180 grados, por último haz que diga: Hola!

Evento:

Haz que, al hacer *click* sobre el gato Scratch este se mueva 50 pasos, espere un segundo, gire 180 grados y por último haz que diga: Soy Scratch!

Loops:

Haz que al presionar la bandera verde el gato Scratch repita 5 veces estos movimientos: que se mueva 50 pasos, espere un segundo y gire 180 grados, por último haz que diga: Soy Scratch!

Paralelismos:

Haz dos bloques grandes siguiendo estos pasos:

Haz un primer bloque en donde al hacer click en la bandera verde el gato se mueva 100 pasos y diga: “Somos Scratch!” durante dos segundos y detén el programa.

En un segundo bloque aparte haz que inicie también al tocar la bandera verde, pero en este haz que aumente el efecto de un solo gato a varios gatos, como si fuera un mosaico y detén el programa.

Condicional:

Haz que al presionar la bandera verde el gato Scratch repita su movimiento hasta que se presione la tecla de espacio. En este caso haz que el gato se mueva sí y solo sí la tecla “flecha derecha” se presiona y que la cantidad de pasos que dé sea de 5. Por último detén el programa.

Operadores:

Haz que al presionar la bandera verde, el gato Scratch pregunte: ¿Cuántos pasos doy? Seguidamente, haz que el gato se mueva de acuerdo a la respuesta a esa pregunta (que debe ser un número) la cual se suma a otro valor que tú escojas entre 5 y 100, para esto usaras el bloque de operadores de suma.








Datos:

Haz que al presionar la bandera verde, el gato se mueva por siempre a una velocidad de 5 pasos. Para hacer esto debes crear una variable que se llame velocidad, una vez la crees debes poner ese bloque en donde deben ir la cantidad de pasos. Por último debes fijar la velocidad del gato Scratch en 5.









No olvide anotar la hora de finalización en la hoja de cálculo y entregar las fichas.


Gracias por participar

ANEXO 3¹

Formato de equivalencias en LEGO (Versión 1)	
Inicio: Bandera	
Inicio: tocando objeto	
Repetir hasta que ...	 Dos piezas
Condicional	 Dos piezas
Movimiento (10, 20, 30, 40 y 50)	  20 pasos x 5 piezas 5 pasos x 5 piezas
Detener programa	

¹ Este formato tiene 17 versiones correspondientes a los 17 paquetes de fichas disponibles. En cada paquete y formato de correspondencias cambian los colores de las fichas más no las fichas como tal.

Por siempre	 <p>Dos piezas</p>
Variable	
Fijar variable	
Esperar	
Unidades de tiempo	 <p>1 segundo x dos piezas</p>
Girar	
Aumentar efecto	
Efectos: Azul: Pixelar Blanco: Desvanecer Verde: Mosaico	

Rebotar si está tocando borde	
Decir	
Mover (activador)	
Operador: suma Usa esta ficha como un +: Ejemplo:    -10	
Sensores (teclas): Tecla de espacio.	
Sensores (teclas): Arriba, abajo, izquierda, derecha (Úsala como una flecha indicando la dirección del ejercicio).	
Pregunta	
Respuesta	

ANEXO 4

Encuesta sobre uso, acceso de tecnologías y conocimiento sobre programas académicos.

1. DATOS GENERALES (Información de la persona que responde el formulario)	
1.1. Nombre completo:	
1.2. Ubicación (Localidad):	
1.3. Sexo:	
Masculino () Femenino ()	
1.4. Edad:	1.5. Fecha:
1.6. Nivel de escolaridad:	1.7. Nombre de la institución:
Colegio () Universitario ()	1.8. Grado o semestre:
2. DATOS SOBRE CONSUMO DE TECNOLOGÍA	
2.1. Tipo de insumos tecnológicos con que dispone. Marque con una x una o varias de las siguientes categorías:	
<input type="checkbox"/> 1. Computador de escritorio <input type="checkbox"/> 2. Portátil <input type="checkbox"/> 3. Tablet <input type="checkbox"/> 4. Internet en casa <input type="checkbox"/> 5. Smartphone	
2.2. ¿Cuánto tiempo dedica a usar el computador en casa?	
<input type="checkbox"/> 1. Entre 1 y 3 horas a la semana <input type="checkbox"/> 2. Entre 4 y 9 horas a la semana <input type="checkbox"/> 3. Más de 9 horas a la semana	
2.3. Para que usa el computador en casa (seleccione una o varias de las siguientes):	
<input type="checkbox"/> 1. Jugar Videojuegos <input type="checkbox"/> 5. Leer <input type="checkbox"/> 2. Navegar en internet <input type="checkbox"/> 6. Revisar el correo electrónico <input type="checkbox"/> 3. Hacer tareas <input type="checkbox"/> 7. Ver o descargar películas <input type="checkbox"/> 4. Escuchar o descargar música <input type="checkbox"/> 8. Otro: _____	
2.4. Selecciona cuál de los siguientes programas usa o ha usado en el computador (ya sea en el colegio/ universidad o en la casa). Marque con una x una o varias de las siguientes categorías:	
<input type="checkbox"/> 1. Paint <input type="checkbox"/> 1. Otro(s): _____ <input type="checkbox"/> 2. Power Point <input type="checkbox"/> 3. Word <input type="checkbox"/> 4. Excel	

<input type="checkbox"/> 2. LOGO <input type="checkbox"/> 3. Scratch <input type="checkbox"/> 4. Photoshop
2.5. Usa o ha usado alguna herramienta de programación: <input type="checkbox"/> 1. Si <input type="checkbox"/> 2. No

2.6. Si su respuesta fue afirmativa, ¿Cuál (es) programa (s) usa o uso?:
2.7. Conoce algún lenguaje de programación: <input type="checkbox"/> 1. Si <input type="checkbox"/> 2. No
2.8. Si su respuesta fue afirmativa, ¿Cuál (es) lenguaje (s) sabe?:
2.9. Ha tomado cursos o clases en programación: <input type="checkbox"/> 1. Si <input type="checkbox"/> 2. No
2.10. Si su respuesta fue afirmativa, ¿Cuál (es) curso (s) o clase (s) ha tomado?:

¡¡GRACIAS!!

ANEXO 5

Consentimiento informado y asentimiento informado

PROYECTO: EFECTOS SOBRE EL PENSAMIENTO COMPUTACIONAL DE EXPERIENCIAS DE APRENDIZAJE EN AMBIENTES VIRTUALES

(Padres de Familia)

La presente investigación tiene como objetivo evaluar empíricamente los efectos del uso sostenido de una herramienta multimedia llamada Scratch y las tres dimensiones del pensamiento computacional que se han identificado: conceptos computacionales, prácticas computacionales y perspectivas computacionales. A continuación se relacionan algunos aspectos generales sobre la investigación. Es muy importante que usted lea con atención y comprenda cada uno de los numerales de modo que, si no está de acuerdo con ellos, lo manifieste en el apartado final del formato.

Procedimiento. Las actividades que realizará su hijo (a), si usted autoriza su participación, serán: 1) una encuesta sobre conocimiento y uso de herramientas de programación 2) un instrumento que consiste en una serie de ejercicios en Scratch, 3) otro instrumento donde realizarán los mismos ejercicios que en Scratch pero esta vez con bloques de LEGO, y por último 4) una entrevistas sobre percepciones y perspectivas en computación y tecnologías. La aplicación se realizará en las instalaciones del colegio Francisco José de Caldas con la supervisión y acompañamiento del docente y el investigador del proyecto. El proyecto cuenta con el aval de la Universidad Nacional de Colombia y la participación de su hijo (a) es totalmente voluntaria.

Riesgos. La participación de su hijo (a) en la investigación no representará ningún tipo de riesgo para su bienestar dado que no se aplicarán intervenciones y por tanto no se alterará ningún aspecto sensitivo de su conducta.

Beneficios. Dado que el objetivo de la investigación consiste brindar una evaluación empírica sobre el pensamiento computacional y sus respectivas dimensiones, no se identifica ningún beneficio directo a corto plazo para su hijo (a), sin embargo, con su participación se hará una contribución para la comprensión de esta habilidad (pensamiento computacional) que todos pueden desarrollar bajo las condiciones de formación adecuadas y también impulsar el uso de programas como Scratch en la enseñanza de la educación básica y media.

Confidencialidad. Se garantizará la confidencialidad respecto a la identidad de su hijo (a). La información obtenida se utilizará con fines académicos y será manejada solamente por el investigador principal y los colaboradores. En el momento de reportar los datos en revistas científicas y/o congresos académicos, se mantendrá la confidencialidad.

Resultados. Los resultados se mantendrán bajo reserva del equipo investigador.

Inquietudes. En caso de que tenga alguna duda referida a la investigación o a los componentes éticos en investigaciones psicológicas puede comunicarse con el investigador que atenderá oportunamente sus inquietudes, a los siguientes datos de contacto:

Luis Alfredo Sánchez Ruiz

Investigador

iluvatar14@gmail.com

Cel. 3213858835

Autorización

Conozco el propósito del proyecto: **“Efectos sobre el pensamiento computacional de experiencias de aprendizaje en ambientes virtuales”**. Entiendo que la participación de mi hijo(a) en el mismo es voluntaria. He revisado la información contenida en este documento. He tenido la oportunidad de aclarar las inquietudes frente al proyecto con los investigadores responsables.

He leído y entendido la información presentada en este formato de consentimiento.

Si está o no de acuerdo con la participación de su hijo(a) en esta actividad, por favor marque a continuación con una X:

Estoy de acuerdo: _____ No estoy de acuerdo: _____

Entiendo que si manifiesto mi acuerdo con lo anterior, mi hijo(a) podrá participar en esta actividad.

Padre/ madre:

Nombre: _____

Firma: _____

CC: _____ Fecha: _____

Investigador:

Nombre: _____

Firma: _____

CC: _____ Fecha: _____

Asentimiento Informado

Esta es una invitación para que participes en nuestra investigación sobre evaluación de los efectos del uso de Scratch en el desarrollo del pensamiento computacional. Para esto, deberás responder unas preguntas con la mayor sinceridad posible y luego realizar los ejercicios propuestos por el

investigador de acuerdo a las instrucciones que él te indique y de manera individual. Por último participarás en una breve entrevista. La información que nos des será confidencial, lo que quiere decir que solo la conoceremos las personas que estamos a cargo de esta investigación.

Tus padres ya nos han dado su autorización para que tú participes en nuestra investigación, y también queremos contar con la tuya. Si estás de acuerdo, puedes firmar este formato.

En caso de que no quieras darnos tu autorización, lo entenderemos y no tendrás ningún tipo de consecuencia por no hacerlo.

Si tienes alguna inquietud sobre la información que te acabamos de presentar, puedes aclararla de inmediato con la persona encargada.

Nombres y apellidos completos

Firma del estudiante

Firma del investigador

ANEXO 6

Plantilla de ejercicios en Scratch

Secuencias:

Haz que al presionar la bandera verde, el gato Scratch se mueva 50 pasos, espere un segundo y gire 180 grados, por último haz que diga: Hola!



Evento:

Haz que, al hacer **click** sobre el gato Scratch este se mueva 50 pasos, espere un segundo, gire 180 grados y por último haz que diga: Soy Scratch!



Loops:

Haz que al presionar la bandera verde el gato Scratch **repita 5 veces** estos movimientos: que se mueva 50 pasos, espere un segundo y gire 180 grados, por último haz que diga: Soy Scratch!

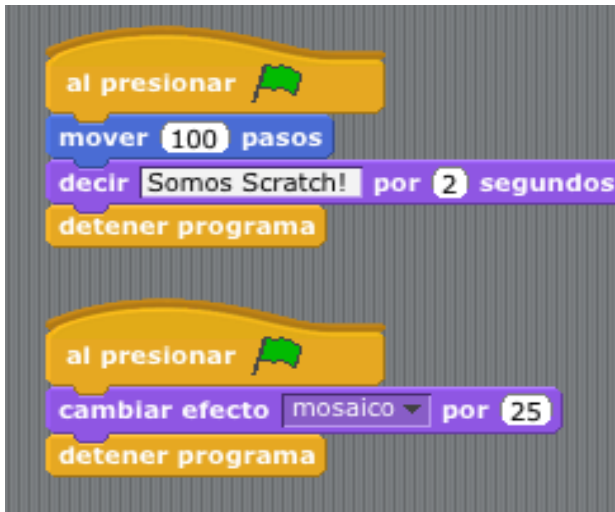


Paralelismos:

Haz dos bloques grandes siguiendo estos pasos:

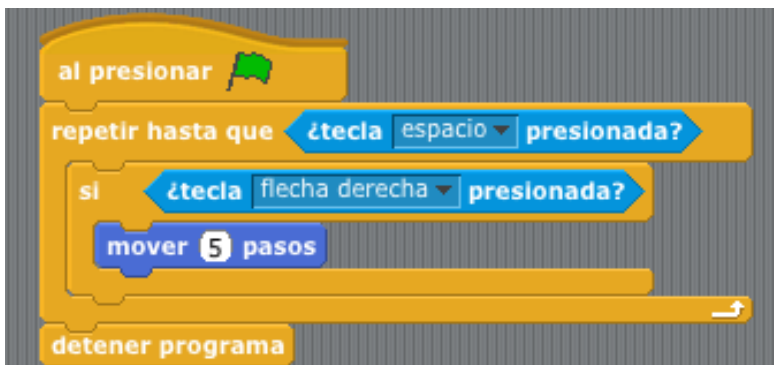
Haz un primer bloque en donde al hacer click en la bandera verde el gato se mueva 100 pasos y diga: "Somos Scratch!" durante dos segundos y detén el programa.

En un segundo bloque aparte haz que inicie también al tocar la bandera verde, pero en este haz que **aumente el efecto** de un solo gato a varios gatos, como si fuera un **mosaico** y detén el programa.



Condicional:

Haz que al presionar la bandera verde el gato Scratch **repita** su movimiento **hasta que** se presione la tecla de espacio. En este caso haz que el gato se mueva **sí** y solo sí la tecla "flecha derecha" se presiona y que la cantidad de pasos que dé sea de 5. Por último detén el programa.



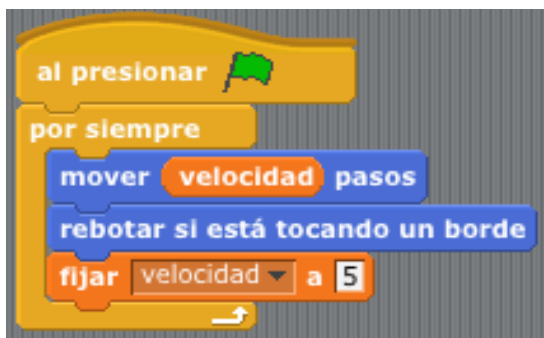
Operadores:

Haz que al presionar la bandera verde, el gato Scratch pregunte: ¿Cuántos pasos doy? Seguidamente, haz que el gato se mueva de acuerdo a la **respuesta** a esa pregunta (que debe ser un número) la cual se suma a otro valor que tú escojas entre 5 y 100, para esto usaras el **bloque de operadores de suma**.



Datos:

Haz que al presionar la bandera verde, el gato se mueva **por siempre** a una velocidad de 5 pasos. Para hacer esto debes **crear una variable** que se llame velocidad, una vez la crees debes poner ese bloque en donde deben ir la cantidad de pasos. Por último debes **fijar la velocidad** del gato Scratch en 5 y detener el programa.

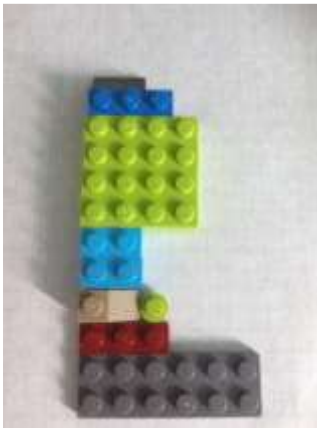


ANEXO 7

Plantilla de ejercicios en LEGO

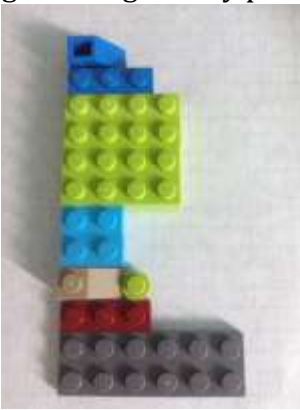
Secuencias:

Haz que al presionar la bandera verde, el gato Scratch se mueva 50 pasos, espere un segundo y gire 180 grados, por último haz que diga: Hola!



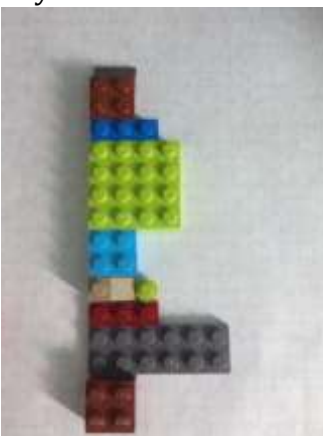
Evento:

Haz que, al hacer **click sobre el gato** Scratch este se mueva 50 pasos, espere un segundo, gire 180 grados y por último haz que diga: Soy Scratch!



Loops:

Haz que al presionar la bandera verde el gato Scratch **repita 5 veces** estos movimientos: que se mueva 50 pasos, espere un segundo y gire 180 grados, por último haz que diga: Soy Scratch!

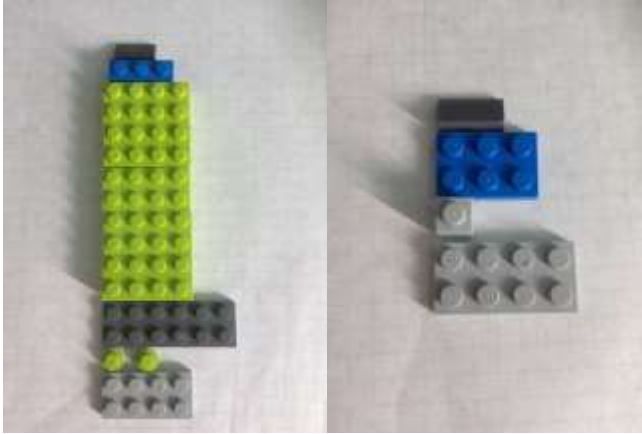


Paralelismos:

Haz dos bloques grandes siguiendo estos pasos:

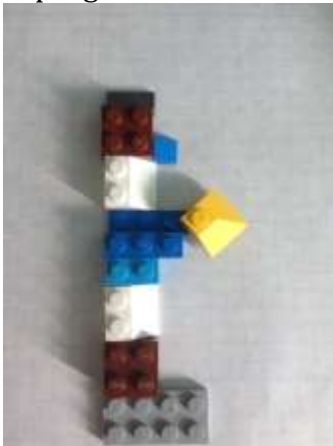
Haz un primer bloque en donde al hacer click en la bandera verde el gato se mueva 100 pasos y diga: "Somos Scratch!" durante dos segundos y detén el programa.

En un segundo bloque aparte haz que inicie también al tocar la bandera verde, pero en este haz que **aumente el efecto** de un solo gato a varios gatos, como si fuera un **mosaico** y detén el programa.



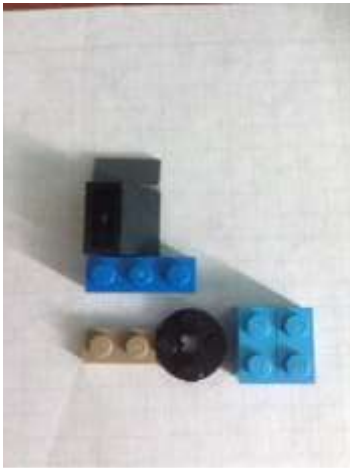
Condicional:

Haz que al presionar la bandera verde el gato Scratch **repita** su movimiento **hasta que** se presione la tecla de espacio. En este caso haz que el gato se mueva **sí** y solo sí la tecla “flecha derecha” se presiona y que la cantidad de pasos que dé sea de 5. Por último detén el programa.

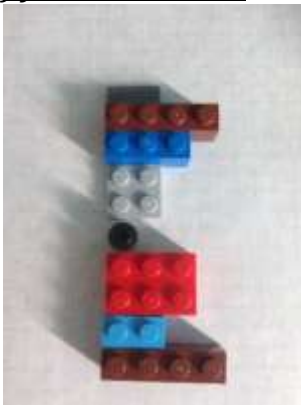


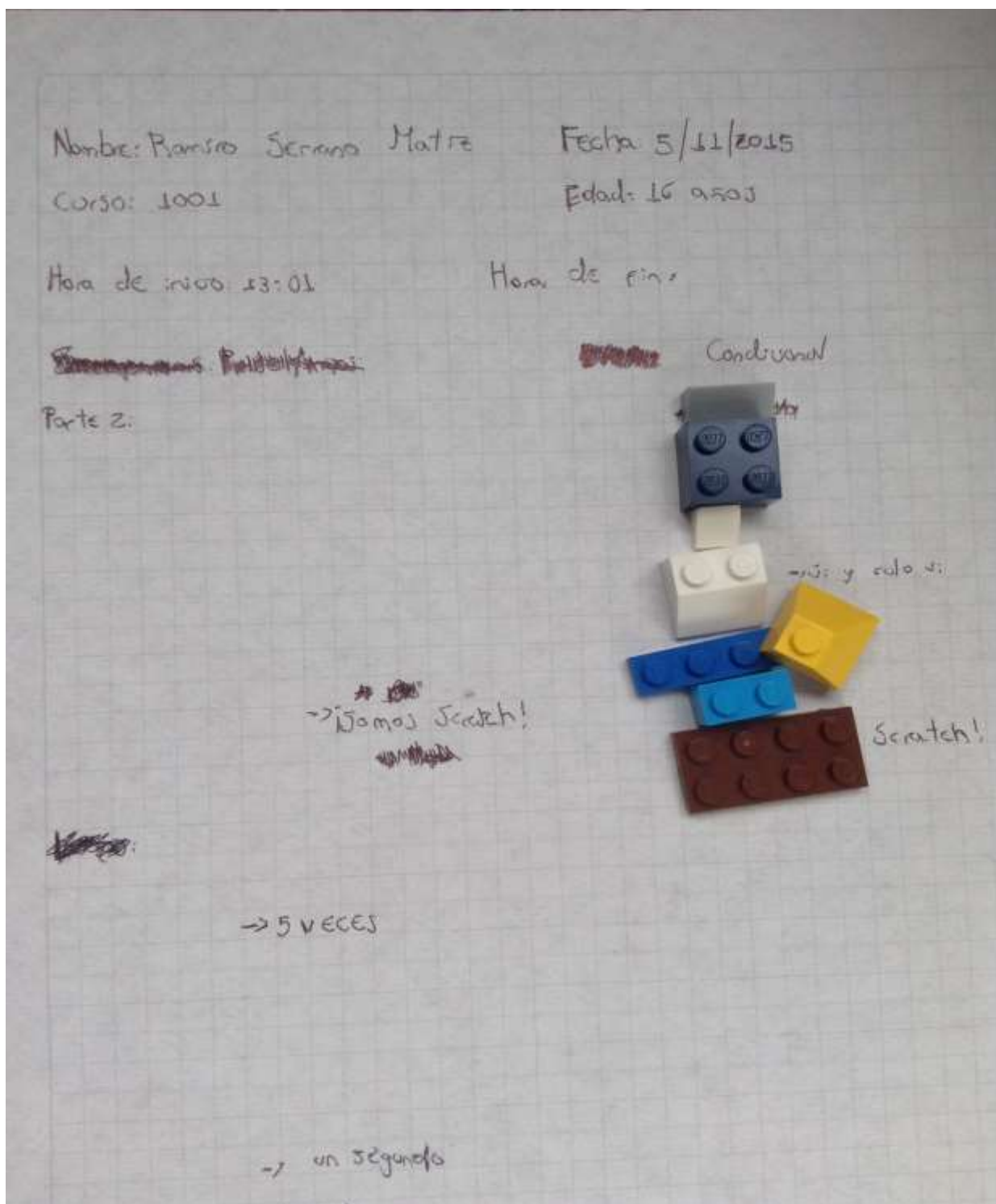
Operadores:

Haz que al presionar la bandera verde, el gato Scratch pregunte: ¿Cuántos pasos doy? Seguidamente, haz que el gato se mueva de acuerdo a la **respuesta** a esa pregunta (que debe ser un número) la cual se suma a otro valor que tú escojas entre 5 y 100, para esto usaras el **bloque de operadores de suma**.

**Datos:**

Haz que al presionar la bandera verde, el gato se mueva **por siempre** a una velocidad de 5 pasos. Para hacer esto debes crear una **variable** que se llame velocidad, una vez la crees debes poner ese bloque en donde deben ir la cantidad de pasos. Por último debes **fijar la velocidad** del gato Scratch en 5.

**ANEXO 8****Ejercicio de la prueba en LEGO**



Reseña biográfica del aspirante

² Este ejercicio corresponde a un estudiante de la prueba en LEGO del grupo experimental. El ejercicio que se está mostrando es el de condicional.

(Bogotá, Colombia 1986) Psicólogo de la Universidad Nacional de Colombia aspirante a magister de la maestría en psicología, línea Educación, cognición y Medios. En los últimos años se ha desempeñado como estudiante auxiliar de la Universidad Nacional de Colombia en diversos proyectos, entre estos se puede mencionar que participó como coordinador y editor de tres artículos investigativos desarrollados para el CIER, actualmente se encuentra en la construcción y redacción de uno. También fue facilitador en un diplomado que se desarrolló entre la Universidad Nacional de Colombia y la Secretaria de educación distrital (SED).