Revista Mexicana de Análisis de la Conducta • Mexican Journal of Behavior Analysis
2018 | Núm. 2 • diciembre | No. 2 • December | Vol. 44, 152 - 172
http://dx.doi.org/10.5514/rmac.v44.i2.68539

NOTA TÉCNICA: DISEÑO Y CONDUCCIÓN DE EXPERIMENTOS CONDUCTUALES MEDIANTE EL USO DE SCRATCH

TECHNICAL NOTE: DESIGNING AND RUNNING BEHAVIORAL EXPERIMENTS WITH SCRATCH

Jorge A. Ruiz y Karina Bermúdez
Universidad Autónoma de Baja California

Resumen

El propósito del presente trabajo fue evaluar la utilidad del lenguaje de programación libre Scratch para diseñar y conducir experimentos conductuales en computadora. Scratch es un software de licencia libre, utilizable tanto en su versión online como en versión offline (Scratch 2.0), en sistemas operativos como Windows o Mac OS y dirigido principalmente a población infantil para promover el desarrollo de habilidades en programación mediante la creación de videojuegos, animaciones, entre otras posibilidades. La programación se basa en la construcción de conjuntos de bloques que controlan las acciones de los elementos que se disponen sobre un fondo y permiten la interacción con el usuario del algoritmo programado a través del

Jorge A. Ruiz y Karina Bermúdez, Facultad de Ciencias Administrativas y Sociales, Universidad Autónoma de Baja California, Unidad Valle Dorado.

La preparación del manuscrito fue apoyada por el financiamiento del Programa para el Desarrollo Profesional Docente para el Tipo Superior concedido al primer autor (folio UABC-PTC-613).

Dirigir correspondencia a: Jorge A. Ruiz, Facultad de Ciencias Administrativas y Sociales, UABC Unidad Valle Dorado, Boulevard Zertuche y Boulevard de los Lagos S/N, Fracc. Valle Dorado. C.P. 22890. Ensenada, Baja California, México. Teléfono: (646) 1766600; ext. 211. Correo electrónico: ruizvja@yahoo.com.

monitor de la computadora y mediante los dispositivos periféricos de ésta (teclado, ratón o incluso extensiones). Se programó un procedimiento que consistió en elegir entre dos opciones que resultaban en "ganar" o "perder" a lo largo de una serie de ensayos en los que la probabilidad de ganar siguió una distribución en forma de U invertida. El software permitió la programación de todos los eventos experimentales, así como el registro de las elecciones, latencias de los participantes, el número de ensayos exitosos, entre otras variables.

Palabras clave: Programación de experimentos, Scratch, software libre, computadora, humanos.

Abstract

The purpose of this paper was to evaluate the usefulness of Scratch to design and run computer-based behavioral experiments. Scratch is a free license software, usable both in its online and offline (Scratch 2.0) versions, with operativing systems like Windows or Mac OS and mainly aimed at children in order to promote the development of programming skills by creating video games and animations, among other possibilities. The programming is based on the construction of sets of blocks that control the actions of the elements that are arranged on a background and allow the interaction with the user through the computer monitor and peripheral devices like the keyboard, mouse or another extension. A procedure was programmed involving choosing between two options that resulted in "winning" or "losing" over a series of trials in which the probability of winning followed an inverted U-shaped distribution was programmed. The software allowed the programming of all the experimental events, as well as the ecording of choices, response latencies, number of successful trials, and other variables.

Keywords: Programming experiments, Scratch, free software, computer-based, humans.

La investigación en el análisis de la conducta se caracteriza por la manipulación de variables ambientales específicas con el objetivo de observar su efecto sobre la conducta. El avance de la tecnología orientada a la automatización y/o control del equipo experimental se ha reflejado en el desarrollo de equipo electrónico y lenguajes de programación que permiten cada vez mayor control ambiental. El uso de equipo electrónico en la conducción de experimentos tiene ciertas ventajas,

primero, permiten evidenciar relaciones ambiente-conducta que no son fáciles de observar a simple vista, segundo, se reduce el error humano en el registro y la presentación de eventos ambientales y tercero, permite la automatización (Escobar & Santillán, 2017).

Si bien, la utilización de equipo electrónico representa ventajas para conducir investigación, en algunas condiciones no es fácil de conseguir debido al precio elevado y a que en algunos países, como México, tiene que exportarse lo que implica un aumento en el costo al tener que pagar los impuestos correspondientes (Escobar & Pérez-Herrera, 2015, Ribeiro, Neto, Morya, Brasil & Pereira de Araújo, 2017).

Como solución a la situación, en los últimos años se han desarrollado dispositivos de bajo costo diseñados para la programación de contingencias ambientales y registro de conducta, por ejemplo, actualmente existen interfaces que utilizan un microcontrolador Arduino© con el lenguaje de programación Visual Basic. Las interfaces se comunican con dispositivos electrónicos, como computadoras y tabletas electrónicas y permiten la presentación de estímulos visuales y auditivos así como el registro de datos (Escobar y Pérez-Herrera, 2015). El funcionamiento de dichos dispositivos se ha probado en tareas de discriminación auditiva con primates (Ribeiro et al. 2017), en la investigación con roedores (Devarakonda, Nguyen & Kravitz, 2016, Escobar & Pérez-Herrera, 2015), e incluso con humanos (Schultz, 2018).

El desarrollo de dispositivos de bajo costo es una alternativa viable, respecto al equipo de laboratorio más costoso, no sólo porque emplea diseños eficientes de los dispositivos de control y registro sino también por la falta de dependencia de un código cerrado de programación que deba pagarse a alguna empresa (cf., Picanco & Tonneau, 2018). La alternativa más empleada ha sido el lenguaje de programación Visual Basic, debido a que es un código fácil de utilizar y es abierto al público (Cabello, Barnes-Holmes, O'Hora & Stewart, 2002). Dixon y MacLin (2003) publicaron un libro sobre Visual Basic dirigido especialmente a psicólogos conductuales, lo cual facilita la tarea de aprendizaje del programa. No obstante, es importante mencionar que los ejemplos que se desarrollan en el texto corresponden principalmente a programas de reforzamiento, lo cual podría representar una dificultad para los programadores novatos que quieran hacer uso del lenguaje en la programación de otro tipo de procedimientos experimentales.

En el contexto de la programación, siempre resulta bienvenida cualquier alternativa que cumpla con las características de facilitar la iniciación en el diseño de algoritmos sin requerir un nivel alto de conocimientos sobre el lenguaje en el que se va a programar, que abra la posibilidad de crear proyectos cada vez más comple-

jos, y que permita la creación de proyectos guiados por diversos tipos de interés y estilos de aprendizaje de los programadores (Guzdial, 2004). En este sentido, Scratch (Resnick et al. 2009) es un entorno de programación visual y multimedia que destaca por su énfasis original para el desarrollo de habilidades de programación en niños y jóvenes (Kalelioğlu & Gülbahar, 2014, Kordaki, 2012), lo cual hace que el entorno de programación sea muy intuitivo y requiera de un nivel prácticamente nulo de conocimientos de programación.

Si bien el objetivo original de los creadores de Scratch fue (y sigue siendo) dotar de una herramienta a niños para promover el aprendizaje y la solución de problemas a través de la programación, gracias a sus características, las cuales se describen en la siguiente sección, podría ser una herramienta útil y práctica para la programación de experimentos cuando no se cuenta con los recursos monetarios para adquirir la licencia de software especializado y principalmente porque requiere de un nivel inicial mínimo de conocimientos y habilidades para la programación (Marji, 2014).

En el presente trabajo se evaluó la utilidad de Scratch para la programación de experimentos conductuales y para la recolección automatizada de datos. En la sección de Prueba se describe un experimento en el cual se probó la viabilidad del uso del programa.

Acerca de Scratch

Scratch es un software creado por el Media Lab's Lifelong Kindergarten Group, del Instituto de Tecnología de Massachusetts (MIT, por sus siglas en inglés), con el propósito original de fomentar el desarrollo de la creatividad en niños (Resnick et al. 2009). El programa, en línea o en versión off-line, permite a los usuarios crear historias animadas, juegos, noticieros en línea, reportes de libros, tarjetas de felicitación, videos musicales, tutoriales, simulaciones y proyectos de arte, entre otras creaciones.

El software de Scratch y el sitio web de éste fueron lanzados al público en el año 2007, está disponible en 50 idiomas, para el sistema operativo Windows, Mac OS X y Linux y es de licencia libre (Maloney, Resnick, Rusk, Silverman & Eastmond, 2010). Para utilizar el programa en línea es necesario contar con una conexión estable a la Internet, algún navegador (Chrome 7 o posterior, Firefox 4 o posterior, o Internet Explorer 7 o posterior) y Adobe Flash Player versión 10.2 o posterior, instalados en la computadora. También es posible descargar la versión offline del software (Scratch 2.0), aunque está sólo puede instalarse en Windows y Mac OS,

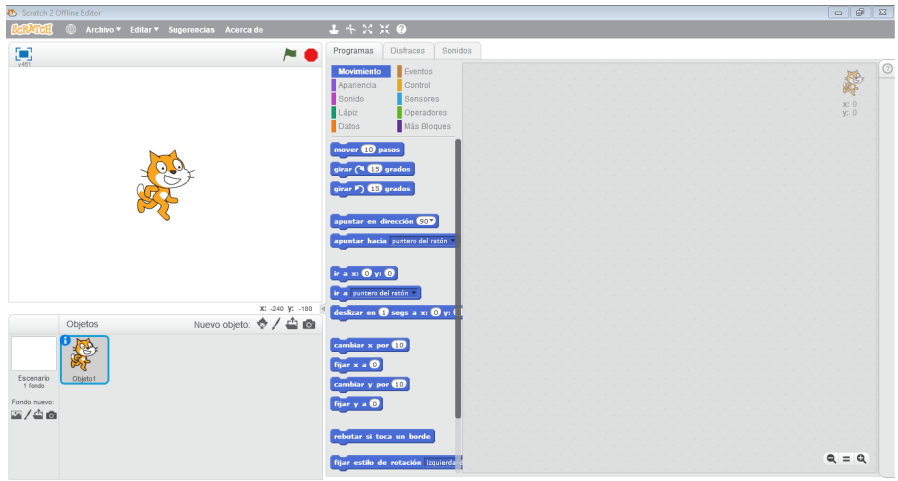

Figura 1. Imagen del área de trabajo de Scratch al iniciar el software.

para ello los requerimientos de sistema operativo son los indicados previamente. Para la versión sin conexión se requiere la instalación de una versión actualizada de Adobe AIR para los usuarios de Windows, y para los usuarios de Mac OS se debe instalar de manera específica la versión 20.

La Figura 1 muestra una captura de pantalla de la interfaz de programación de Scratch, tal como aparece por default al inicializar la versión offline Scratch 2.0. El menú principal cuenta con los botones que permiten ir al sitio web, cambiar el idioma del programa, el menú de Archivo, de Edición, sugerencias de tutoriales o guías para crear nuevos proyectos y la opción para conocer más acerca de Scratch. También incluye botones de acciones específicas para el uso de bloques de programación: duplicar, cortar, cambiar de tamaño, así como para acceder a la ayuda de bloques.

Durante la fase de programación la pantalla se divide en tres áreas. En la parte superior izquierda se muestra el escenario, en el que aparecerán todos los fondos y objetos que se incluyan en el programa diseñado. En la parte inferior izquierda, se muestran los fondos y objetos que están disponibles para el diseño del programa que se está desarrollando, los cuales deben ser creados por el programador, seleccionados del banco de imágenes de Scratch, o importados de archivos externos para que aparezcan como disponibles durante el proceso de programación (en formato de mapa de bits). En la parte de la derecha aparecen tres pestañas, de las cuales la primera corresponde al área de programación en la que se muestra una paleta

Tabla 1. *Categorías de bloques de programación y sus características.*

Categorías	Características
Movimiento	Consta de 14 bloques que permiten controlar la posición, orientación y desplazamiento de los objetos incluidos en el programa a través del escenario. Además de tres bloques que permiten insertar la posición en X y en Y, así como la dirección del objeto, como valores de variables requeridas o anidadas en otros bloques.
Apariencia	Consta de 16 bloques que permiten presentar texto en el escenario del programa, mostrar, esconder o cambiar fondos y disfraces de los objetos, así como su tamaño y color. Contiene tres bloques que permiten insertar el número de un disfraz específico, un fondo o el tamaño de un objeto como valores de variables requeridas o anidadas en otros bloques.
Sonido	Consta de 11 bloques que permiten controlar la presentación de sonidos, de la biblioteca de Scratch, grabados por el programador o importados de otras fuentes. Contiene dos bloques que permiten insertar el volumen de un sonido y el tempo como valores de variables requeridas o anidadas en otros bloques.
Lápiz	Consta de 11 bloques que permiten hacer trazos de diferentes colores y tamaños como si se escribiera sobre el escenario del programa.
Datos	Consta de dos botones que permiten la creación de una variable y la creación de una lista o arreglo. El programador puede crear y nombrar las variables y las listas que requiera para el desarrollo del programa. Cuando se crea una variable, se hacen disponibles cuatro bloques que permiten fijar y cambiar valores de la variable, así como mostrarla en el escenario del programa u ocultarla. Cuando se crea una lista, se activan nueve bloques que permiten añadir, borrar, insertar o reemplazar elementos dentro de la lista, "leer" los elementos en la lista, mostrarla en el escenario u ocultarla. También es posible importar listas completas con valores de variables que podrían ser pertinentes para la programación, así como exportar las listas que el programa genera. Scratch usa un formato de columna tanto para las listas que son importadas como para las que son exportadas, sin embargo, cuando se crea una "lista de listas" están son organizadas a manera de matriz, con cada lista ocupando un renglón diferente y los elementos de cada lista ocupando las columnas.
Eventos	Consta de ocho bloques que cumplen la función de iniciar un programa (o parte de él). Se colocan siempre en la parte superior de cualquier conjunto de bloques y dependen de inputs provenientes de los dispositivos conectados a la computadora (como los botones del teclado), o bien, de inputs generados durante el funcionamiento del programa (como cuando se cumple alguna instrucción particular). Dos de los ocho bloques se pueden colocar en algún punto diferente al inicio del bloque y permiten la conexión con otros conjuntos de bloques a través del "envío de mensajes".
Control	Consta de 11 bloques que permiten controlar principalmente relaciones condicionales entre objetos, variables o bloques. Dos de los bloques tienen la peculiaridad de ser bloques de finalización de conjuntos de bloques y cumplen con la función de detener procesos en operación.

Sensores	Consta de 20 bloques que cumplen con la función de registrar eventos o condiciones como inputs para ser tomados como valores en otros bloques de programación. Destaca el bloque Cronómetro, el cual puede ser usado en cualquier momento en el que se requiera muestrear el tiempo de ocurrencia de un evento (con una resolución de milésimas de segundo).
Operadores	Consta de 17 bloques que permiten realizar diversas operaciones aritméticas entre valores de las variables involucradas en el programa, el uso de operadores booleanos y la selección aleatoria de números dentro de un rango definido por el programador.
Más bloques	Consta de dos botones que permiten la creación de nuevos bloques y la conexión con dispositivos periféricos que permiten el control de señales de entrada y de salida. El programador puede crear y nombrar los nuevos bloques que requiera para el desarrollo del programa. Cuando se crea un nuevo bloque, se hace disponible un nuevo bloque de inicio, cuya función será definida por el usuario colocando los bloques necesarios para desarrollar el algoritmo deseado para ese nuevo bloque. Al añadir una extensión, Scratch permite la comunicación entre el programa desarrollado y dispositivos como LEGO WeDo 1.0, LEGO WeDo 2.0 y PicoBoard. Dependiendo de la extensión conectada, se activan un número diferente de bloques (en color gris) para establecer el control de los inputs y outputs regulados por tales dispositivos.

de bloques de programación y el espacio en el que se van colocando los bloques para desarrollar el algoritmo deseado. La segunda pestaña muestra los "disfraces" o apariencias que puede tomar cada uno de los objetos incluidos en el programa y que aparecen disponibles en el área inferior izquierda mencionada anteriormente. En la tercera pestaña se muestra el espacio en el que se designan los sonidos que podrían utilizarse en el programa, los cuales pueden ser elegidos de la biblioteca de Scratch, pueden ser grabados por el programador o incluso puede importarlos de otros archivos.

La sección de la plataforma de Scratch más relevante para el desarrollo de programas es la paleta de bloques, junto con el área de escritura. El programa incluye 10 categorías de bloques de construcción con diferentes funciones e identificados con diferentes colores: Movimiento (azul), Apariencia (violeta), Sonido (rosa), Lápiz (verde oscuro), Datos (naranja), Eventos (café), Control (amarillo), Sensores (azul claro), Operadores (verde claro), Más bloques (púrpura). En la Tabla 1 se describen las funciones básicas de cada categoría de bloques. Los bloques tienen la forma de piezas de rompecabezas y se pueden organizar de manera secuencial en el área de escritura para crear uno a más algoritmos independientes o relacionados, los cuales colectivamente controlan lo que ocurre en el escenario. La programación consiste

en seleccionar los bloques que encajan de manera automática creando instrucciones con una sintaxis apropiada. Las áreas de programación están siempre visibles mientras se elabora el programa, aunque se puede ocultar todo cuando el programa está en operación y mostrar únicamente el escenario.

Prueba

Para probar la utilidad del programa en el diseño de experimentos conductuales y recolección automatizada de datos, se llevó cabo una réplica sistemática del procedimiento propuesto por Daugherty y Quay (1991) para estudiar la persistencia de la respuesta en niños con desórdenes conductuales. En su versión original los participantes recibían 10 fichas al inicio del experimento, intercambiables por algún premio al finalizar su participación. La tarea experimental consistía en elegir a lo largo de una serie de 100 ensayos entre jugar o abandonar la tarea. En cada ensayo se presentaba una puerta cerrada y si el participante elegía jugar, el resultado al abrir la puerta podía ser ganar o perder una ficha. Por cada bloque de 10 ensayos la probabilidad de ganar disminuía en un 10 % (i.e., de 0.90 a 0.00). Las puertas que resultaban en ganar o perder se presentaban en un orden aleatorio a través de cada bloque de ensayos. La variable dependiente en este procedimiento es el número total de fichas acumuladas al momento de abandonar la tarea. Posteriormente, Miller, DeLeon, Toole, Lieving y Allman (2016) encontraron que esta tarea es confiable para medir la persistencia de la conducta en apuestas e incluso es sensible a la manipulación de variables como la entrega contingente o no contigente de fichas al inicio de la tarea.

Método

Participantes

Participaron 10 mujeres, con un rango de edad entre los 19 y 22 años. Su participación fue voluntaria y no se compensó de ninguna manera su participación en el experimento.

Instrumentos y materiales

Se empleó una computadora de escritorio (marca Dell, modelo OPTIPLEX 7010, Procesador CORE i7, memoria RAM 8 GB) para programar y presentar los estímulos visuales de la tarea experimental a través de un monitor de 20 pulgadas

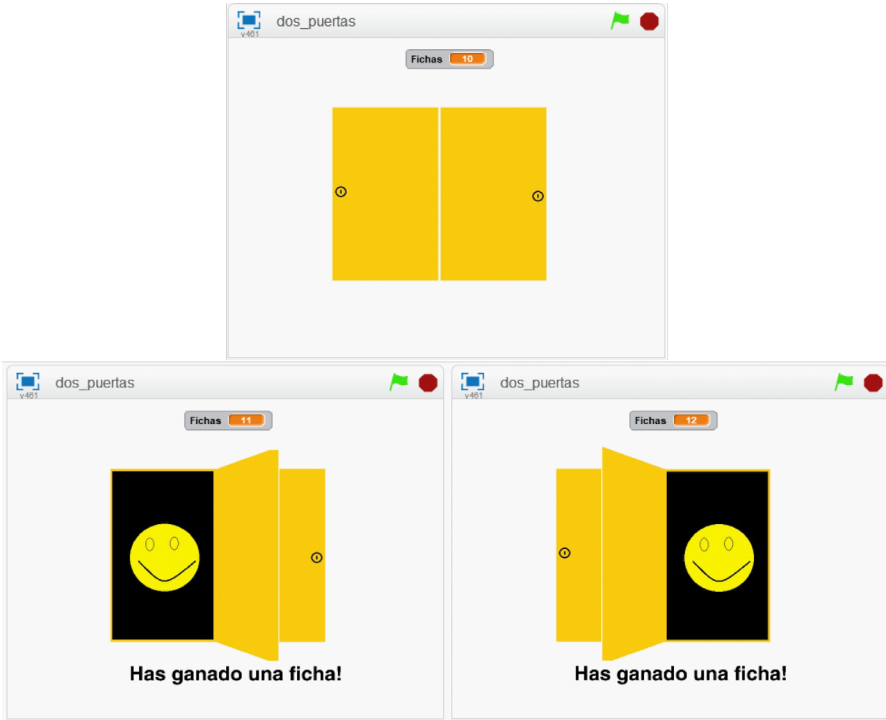

Figura 2. Escenarios posibles a lo largo del experimento de prueba, iniciando por la elección entre dos puertas cerradas (imagen superior) y con ejemplos de los posibles resultados al abrir la puerta de la izquierda (imagen inferior izquierda) o la puerta de la derecha (imagen inferior derecha).

(marca Dell, modelo E2011Hc, resolución: 1600 x 900). Las respuestas de los participantes se registraron mediante la presión de las teclas A, B, flecha izquierda y flecha derecha del teclado (marca DELL, modelo KB212-B).

La tarea experimental se programó usando Scratch 2.0. Los fondos y disfraces mostrados durante la operación del programa fueron diseñados por los autores del trabajo mediante el uso de Paint.

Procedimiento

Se diseñó una tarea semejante a la versión original de Daugherty y Quay (1991), con la diferencia de que se presentaron dos puertas (en lugar de una) en cada ensayo. El motivo de la modificación del procedimiento original obedece a propósitos relacionados con otras investigaciones en curso, en las que resulta relevante esta

Tabla2. Ganancias y pérdidas a través de los 10 bloques de ensayos.

Bloque	Ensayos	Ganados	Perdidos	Saldo	Acumulado
	0				10
1	1 a 10	9	1	8	18
2	11 a 20	8	2	6	24
3	21 a 30	7	3	4	28
4	31 a 40	6	4	2	30
5	41 a 50	5	5	0	30
6	51 a 60	4	6	-2	28
7	61 a 70	3	7	-4	24
8	71 a 80	2	8	-6	18
9	81 a 90	1	9	-8	10
10	91 a 100	0	10	-10	0

manipulación pero se requería probar en primer lugar la utilidad de Scratch para programar la tarea. En la Figura 2 se muestran capturas de pantalla que ilustran lo que aparecía en el monitor ante los participantes. En el panel superior se muestra la imagen que aparecía al inicio de cada ensayo, i.e., "dos puertas cerradas". Si los participantes elegían jugar, debían presionar la *flecha izquierda* del teclado para "abrir" la puerta izquierda (panel inferior izquierdo), o bien, presionar la *flecha derecha* para abrir la otra puerta (panel inferior derecho). Si los participantes elegían abandonar el juego, debían presionar la tecla *Escape* para salir de la tarea experimental. Para distinguir en el escenario cuando el participante "ganaba" o "perdía", al abrirse las puertas se presentaba durante 500 ms una imagen que simulaba una "cara sonriente" o una "cara triste", acompañadas de la leyenda "Has ganado una ficha!" o "Has perdido una ficha!", respectivamente. En cuanto finalizaba la presentación de la cara y leyenda correspondiente, se volvía a presentar inmediatamente la imagen de las dos puertas cerradas.

En la Tabla 2 se muestran el número de fichas que los participantes podían ganar o perder en cada bloque de ensayos, así como el saldo a favor (o en contra) y el número acumulado de fichas al finalizar cada bloque. La probabilidad de ganar (o

perder) estuvo determinada por la proporción de ensayos exitosos programados en cada bloque y fue independiente de la puerta que el participante elegía abrir (i.e., el resultado en cada puerta era el mismo en cada ensayo).

Las sesiones se llevaron a cabo de manera individual en un cubículo de 1.5 m x 1.5 m, iluminado con luz artificial y aislado de los sonidos del exterior. A todos los participantes se les informó el objetivo del estudio y se les indicó que su participación era únicamente para probar el funcionamiento de un lenguaje de programación para desarrollar tareas conductuales en computadora, por lo que su ejecución no se juzgaría como correcta o incorrecta ni con ninguna otra etiqueta. Todas las voluntarias firmaron un consentimiento informado para poder usar los datos de su ejecución para los fines de este trabajo. A todos los participantes se les indicaron las instrucciones del procedimiento de manera verbal, de la manera más parecida al procedimiento de Miller et al. (2016), ya que Daugherty y Quay (1991) no describieron con claridad cuáles fueron las indicaciones a sus participantes. Los valores del intervalo entre ensayos y de la duración de los estímulos se eligieron de manera arbitraria debido a que en los estudios anteriores no se proporcionó dicha información.

Diseño del programa

El programa se diseñó sobre un fondo blanco, el cual se creó a partir de la herramienta *Fondo nuevo* de Scratch coloreando de blanco todo el plano del escenario. El fondo tuvo tres versiones, uno completamente blanco, otro con la leyenda "Has ganado una ficha!" y otro con la leyenda "Has perdido una ficha!".

Sobre el fondo se presentó un objeto (la imagen de dos puertas cerradas, en color amarillo) que podía tomar la apariencia de alguno de cinco disfraces: 1) puertas cerradas, 2) puerta izquierda cerrada y derecha abierta con una cara sonriente, 3) puerta izquierda cerrada y derecha abierta con una cara triste, 4) puerta izquierda abierta con una cara sonriente y derecha cerrada y, 5) puerta izquierda abierta con una cara triste y derecha cerrada.

En la Figura 3 se muestra una captura de pantalla del área de escritura, en la que aparecen siete conjuntos de bloques completos y la parte de un bloque considerablemente más largo que los demás, pero cuya función se describe a continuación. El conjunto de bloques más largo se iniciaba al presionar la bandera verde visible en el escenario del programa, coloca al objeto en una posición específica dentro del plano y establece al fondo completamente blanco y a las puertas cerradas como el inicio de la tarea. Limpia los valores de las variables y listas involucradas en la tarea, estableciendo como valor inicial del número de fichas igual a 10. Posteriormente,

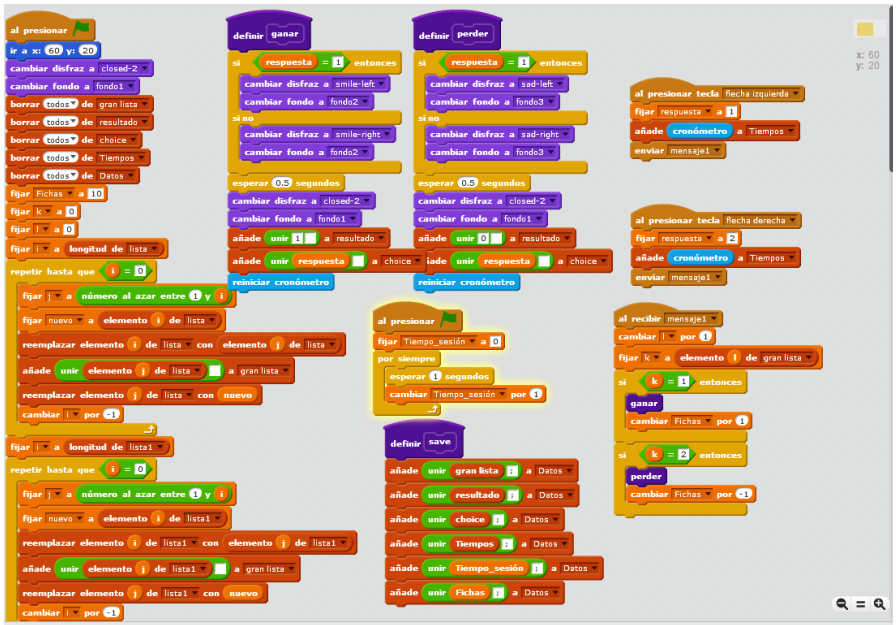

Figura 3. Imagen del área de escritura en la que se muestra parte del algoritmo programado para operar el procedimiento descrito.

se activaba un algoritmo repetitivo para leer una serie de 10 listas de valores previamente importadas de una serie de archivos de texto, las cuales tenían una longitud de 10 elementos configurados de tal manera que la primera lista contenía nueve valores "1" y un valor "2", la segunda lista contenía ocho valores "1" y dos valores "2", y así sucesivamente hasta el caso de la décima lista que contenía 10 valores "2". El algoritmo repetitivo hace una extracción aleatoria de los elementos de cada lista cada vez que se inicia el programa, de tal manera para cada participante se puede generar aleatoriamente una "gran lista" de 100 elementos que permite una configuración de éxitos y fracasos como la descrita en el procedimiento.

Dos conjuntos de bloques, prácticamente iguales, sirvieron para asignar el valor 1 o 2 a la variable "respuesta" en función de si el participante presionaba la tecla "izquierda" o la tecla "derecha", para registrar la latencia respecto al resultado anterior (o respecto al inicio del programa, en el caso de la primera respuesta) y para enviar un mensaje a otro conjunto de bloques. Ese conjunto de bloques se encargaba de hacer la lectura del elemento correspondiente de la "gran lista" de éxitos y fracasos (valores 1 y 2), para entonces establecer mediante dos bloques condicionales si el

participante ganaba una ficha o si perdía una ficha, además de reiniciar el cronómetro. Los resultados de ganar y perder se crearon a partir de la herramienta *Nuevos bloques*, para diseñar un bloque de ganar y otro bloque para perder, ambos con la misma estructura pero diferente función. El bloque de ganar cumplía con la función de evaluar si se había activado la tecla izquierda, para entonces presentar la cara sonriente del lado izquierdo (junto con la leyenda "has ganado una ficha") o si no (i.e., se había activado la tecla derecha), se presentaba la cara sonriente del lado derecho (junto con la leyenda "has ganado una ficha!"). Después de 500 ms se volvía presentar el fondo blanco y las dos puertas cerradas, además de que se añadían el valor 1 a una nueva lista nombrada como "resultado" y el valor de la "respuesta" registrada a otra lista nombrada "choice". El bloque de perder hacía básicamente lo mismo que el anterior, con la diferencia de que se especificaba la presentación de la cara triste del lado izquierdo o derecho (según la respuesta registrada) y de la leyenda "has perdido una ficha!".

Otro conjunto de bloques, creado mediante la herramienta de Más bloques, sirvió para unir los valores registrados de las listas generadas en una nueva lista nombrada como Datos, la cual incluyó las listas: gran lista (éxito o fracaso programado al azar), resultado (éxito o fracaso registrado por el programa después de la respuesta), choice (respuesta en la tecla izquierda o en la tecla derecha), tiempos (lista de las latencias registradas), tiempo de la sesión (duración registrada en segundos), y número de fichas acumuladas al abandonar la tarea. Una vez que finalizaba el experimento para cada participante la lista Datos era exportada como archivo de texto (**.txt*), para posteriormente hacer el copia de esos datos en formato de tabla en una *Hoja de Cálculo* de Excel. El programa se encuentra disponible en línea a través de la liga https://scratch.mit.edu/projects/264193316/.

Resultados y Discusión

En la Figura 4 se muestra la captura de pantalla de un ejemplo de cómo aparecen los datos al ser importados a una hoja de cálculo de Excel (filas 1 a 6, de la columna A a la columna CV) desde el archivo creado a partir del bloque Guardar (diseñado por los autores). En la misma imagen se muestra en la esquina inferior derecha las casillas que hay que activar al momento de abrir el archivo de texto desde la aplicación Excel (separación por espacios y por punto y coma) para que al momento de abrir el archivo de texto, muestre cada valor en una celda diferente (y no a todos los valores en una sola celda).

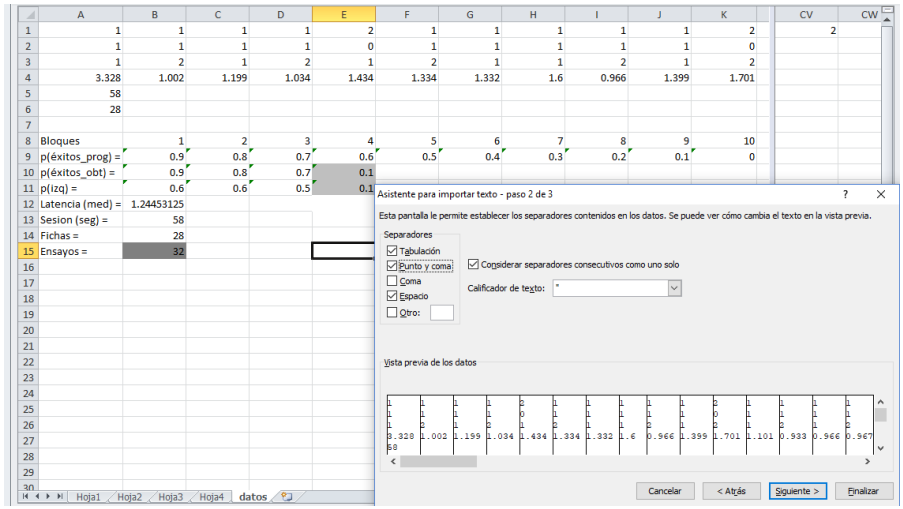

Figura 4. Imagen de los datos recolectados por el programa al ser exportados como archivo de texto e importados a una hoja de cálculo de Excel (primeras 6 filas), además de resultados que se pueden extraer a partir del análisis de los datos (filas 9 a 15).

En la Fila 1 de la imagen mostrada en la Figura 4 aparecen 100 valores que corresponden a la lista producida por el programa para indicar que cada ensayo debería resultar en un éxito (dígito 1) o en un fracaso (dígito 2), i.e., es la "gran-lista". La Fila 2 muestra el tipo de resultado obtenido después de la respuesta del participante, la cual debe coincidir con la "gran-lista" en cuanto a la ocurrencia del dígito 1 en ambas filas o el dígito 2 y 0 en la primera y la segunda fila. En la Fila 3 aparecen las respuestas izquierda (dígito 1) o derecha (dígito 2) en cada ensayo. En la Fila 4 se muestra el valor de la latencia (en segundos) para cada ensayo. En la Fila 5 se muestra la duración de la sesión (en segundos) y en la Fila 6 el número de fichas acumuladas al abandonar la tarea. Mientras que la Fila 1 siempre tiene una extensión de 100 columnas (correspondientes a los 100 ensayos programados), las Filas 2, 3 y 4 pueden tener una extensión igual o menor a 100 columnas (en función del número de ensayos jugados antes de abandonar la tarea), aunque la extensión de éstas tres filas siempre debe coincidir.

En las Filas 9 a 13 de la imagen que se muestra en la Figura 4 se incluyen ejemplos del tipo de información que puede obtenerse al analizar los datos recolectados. Por ejemplo, en las Filas 9 y 10 se muestra la proporción de ensayos exitosos programada en cada bloque de 10 ensayos y la proporción de fichas ganadas en cada uno

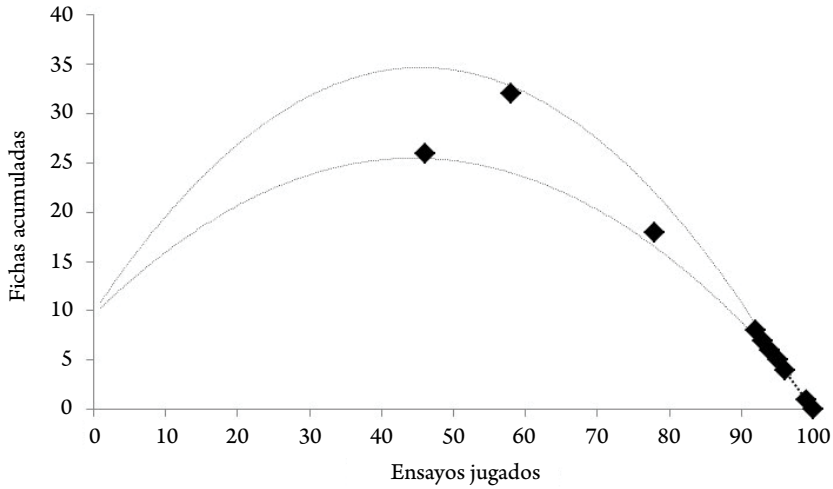

Figura 5. Número de fichas acumuladas en función del número de ensayos jugados antes de abandonar el juego para cada uno de los participantes. Las líneas punteadas señalan el número mínimo y el número máximo de fichas en cada ensayo de acuerdo con las diferentes secuencias de resultados posibles.

de esos bloques. Es evidente que la proporción de ensayos exitosos programados y el número de fichas obtenidas son consistentes entre sí y son consistentes con la lógica de la tarea programada, i.e., una disminución del 10 % en la proporción de ensayos exitosos a través de cada bloque de 10 ensayos. Nótese que el valor de la celda E10 no corresponde con el de la celda E9, esto se debe a que se abandonó la tarea antes de completar el cuarto bloque de ensayos (véase la celda B10 en donde se indica el número de ensayos jugados). En la Fila 11 se muestra la proporción de respuestas en la tecla izquierda, i.e., $R_{izq}/(R_{izq} + R_{der})$, para cada bloque de ensayos. En la Fila 12 se muestra el promedio de la latencia a lo largo de todos los ensayos jugados (también podría calcularse para cada bloque de ensayos). En las Filas 13 y 14 se vuelven a mostrar los valores de la duración de la tarea y el número de fichas acumuladas.

Si bien se contó con la posibilidad de registro de diferentes datos, por motivos de espacio y de énfasis en la prueba del funcionamiento de algoritmo programado, se muestra en la Figura 5 únicamente el número de fichas acumuladas en función de los ensayos jugados para cada participante. Así mismo, se proyectan en el plano dos líneas punteadas que describen el mínimo y máximo número esperado de

fichas acumuladas en cada uno de los 100 ensayos, de acuerdo con la probabilidad decreciente de ganar a lo largo de la tarea. Se encontró que dos de las participantes abandonaron el juego en el ensayo 45 y en el 60, lo cual corresponde con el momento óptimo para hacerlo en el sentido de que es justo cuando se ha acumulado el mayor número de fichas antes de comenzar a perderlas con una probabilidad cada vez mayor. El resto de participantes eligieron seguir apostando hasta llegar al Bloque 10, en el que finalizaron con menos de las 10 fichas que se les habían otorgado al inicio de la tarea, una de ellas incluso jugó hasta perder todas las fichas.

En el estudio de Daugherty y Quay (1991) los participantes eran niños que habían sido clasificados en una de cinco categorías (con desórdenes de la conducta, con déficit de atención, con desórdenes de la conducta y déficit de atención e hiperactividad, con trastorno de ansiedad y un grupo control). Los autores encontraron que los niños con desórdenes conductuales y con problemas de atención jugaron un mayor número de veces (abrieron entre 56.1 y 71.2 puertas), mientras que los niños del grupo control lo hicieron menos veces (34.9 puertas en promedio). Miller et al. (2016) contaron con la participación de estudiantes universitarios, lo cual permite una comparabilidad más directa entre los resultados encontrados en la presente Prueba y los reportados por tales autores. Ellos encontraron que un grupo de 13 estudiantes que recibieron 20 fichas de manera no contingente a su conducta antes de la tarea experimental jugaron 93.2 veces en promedio (con un promedio de fichas acumuladas de 17.8), mientras que otro grupo de 13 estudiantes que tuvieron que participar en una tarea previa para ganar las 20 fichas que pudieron apostar en la tarea experimental jugaron un menor número de veces (46.1 en promedio) logrando acumular un mayor número de fichas (25.6 en promedio).

En la presente Prueba se encontraron resultados semejantes a los del grupo de estudiantes que recibieron de manera no contingente las fichas para jugar en el experimento de Miller et al. (2016), lo que permite asumir que el procedimiento actual (con la inclusión de una segunda puerta) es comparable con el procedimiento diseñado por tales autores.

Respecto a la efectividad de Scratch para diseñar e implementar la tarea experimental es posible concluir que se pudo recrear un procedimiento semejante a la versión original de Daugherty y Quay (1991). Fue posible generar series aleatorias de ensayos de "ganar" y "perder" por cada uno de los 10 bloques de 10 ensayos que cumplieran con las características del diseño experimental. De manera complementaria, los resultados obtenidos sobre las ejecuciones de los participantes mostraron que, independientemente del número de fichas jugadas, el acumulado de éstas siem-

pre se mantuvo dentro de los rangos esperados de acuerdo a un funcionamiento adecuado del algoritmo programado.

A la fecha se han programado otros procedimientos que también han mostrado la utilidad de Scratch para el diseño y conducción de experimentos conductuales en computadora para evaluar la elección entre alternativas para apostar en máquinas tragamonedas virtuales con frecuencias de reforzamiento condicionado diferentes (Ruiz & Bermúdez, 2017) y también para evaluar la resistencia a la extinción en función de la frecuencia de reforzamiento condicionado (Bermúdez & Ruiz, 2018), destacando de esta manera la posibilidad de programar procedimientos de no sólo de ensayo discreto, sino también de operante libre.

Conclusión

El propósito del presente trabajo fue evaluar la utilidad del lenguaje libre Scratch para la programación y conducción de experimentos conductuales. La presentación organizada de los eventos experimentales y la distribución de las fichas acumuladas por parte de los participantes muestran que este lenguaje permitió la programación adecuada del procedimiento descrito, así como el registro de las variables dependientes pertinentes a la situación experimental. Aunque en los estudios realizados por Daugherty y Quay (1991) y Miller et al. (2016) se emplearon lenguajes de programación como MacIntosh Plus y Visual Basic, para los que se cuenta con guías bien desarrolladas para la programación de tareas experimentales (e.g., Dixon & MacLin, 2003), los resultados obtenidos en la prueba llevada a cabo en este trabajo sugieren que Scratch podría ser una opción viable para la programación de un procedimiento semejante.

Desde hace algunos años, existen alternativas comerciales que fueron especialmente diseñadas para la programación de experimentos con humanos, como es el caso de E-Prime y SuperLab (Haxby et al. 1993; Ragozzine, 2002; Taylor & Marsh, 2017). Sin embargo, no siempre se cuenta con la posibilidad de pagar la cantidad de dinero que cuestan las licencias de estos programas, 995 dólares en el caso de E-Prime(https://pstnet.com/products/e-prime/) y 695 dólares en el caso de SuperLab (https://cedrus.com/store/), por lo que la posibilidad de contar con un lenguaje libre, que requiere conocimientos mínimos sobre programación y con una serie versátil de herramientas, abre la oportunidad para desarrollar herramientas para la investigación y apoyar las labores de docencia en laboratorios y escuelas sin la necesidad de invertir mucho dinero.

Existen otras alternativas de uso libre para la conducción de experimentos con humanos, como es el caso del Lenguaje de Construcción de Experimentos en Psicología (PEBL, por sus siglas en inglés [Mueller, 2011]) y PsychoP (enfocado principalmente a la conducción de tareas psicofísicas [Peirce, 2007]). Estas opciones cuentan con una serie de procedimientos pre-programados como la estimación de longitudes, tareas de reconocimiento, medición de tiempos de reacción, entre otras, en los cuales se pueden modificar valores de los parámetros de las variables involucradas en cada experimento. Si el usuario requiere de la programación de algún procedimiento no incluido, puede acceder al código base y crear el algoritmo necesario, sin embargo, para llevar a cabo esa tarea el investigador debe tener cierto dominio de lenguajes de programación como C++ o Python. La ventaja de Scratch, nuevamente, es que se requiere un mínimo conocimiento de programación para utilizarlo e incluso, podría ser útil para acercar a los estudiantes y programadores novatos al uso de esos lenguajes de programación (o a otros).

Aunque Scratch no es un lenguaje de programación creado para el diseño y conducción de experimentos, su aplicación en este campo tiene ciertas ventajas cuando no se cuenta con los recursos monetarios para adquirir software especializado. Las principales son su fácil uso, no requiere tener mucho conocimiento sobre programación debido a que el diseño es didáctico e intuitivo, permite la recolección de datos a través de cualquier medio de entrada de la computadora (teclado, mouse, monitor sensible al tacto, entrada de audio) y, además, es gratuito (Maloney et al. 2010). En este sentido, Scratch facilita la conducción de experimentos y recolección de datos sin la necesidad de instrumentos de registro de respuestas costosos, por ejemplo, los Response Pads que comercializa el mismo grupo que produce SuperLab, cuyo precio es de 499 dólares por unidad (https://cedrus.com/store/). No obstante, debe reconocerse que usar equipo periférico como el teclado de la computadora también puede representar ciertas desventajas respecto a instrumentos diseñados específicamente para el registro de respuestas, en el sentido de que hay mayor posibilidad de que el participante emita una respuesta no esperada (e.g., que presione una tecla inoperativa).

Otra característica importante de Scratch es que permite la conexión directa mediante la herramienta Extensiones (disponible en el categoría Más bloques) con tarjetas como PicoBoard (https://www.picocricket.com/picoboard.html), la cual cuenta con un sensor de sonido, un sensor de luz, un botón, un sensor deslizante y cuatro terminales libres en las que se puede conectar cualquier tipo de sensor para enviar señales al programa mediante una conexión USB. Esta característica

da oportunidad de que el programador pueda crear sus propios medios de registro de respuestas y conectarlos con Scratch.

Como anotación final, debido a que el propósito original de los creadores de Scratch no está encaminado al desarrollo de procedimientos experimentales en laboratorios de investigación psicológica, se debe tener en cuenta que algunos algoritmos podrían requerir cierto grado de involucramiento por parte del programador novato para encaminar las herramientas básicas de Scratch hacia su utilidad en la programación de tareas un tanto más sofisticadas y como un medio de aproximación hacia el aprendizaje de lenguajes de programación que en primera instancia podrían parecer menos intuitivos pero cuya utilidad ha sido demostrada (e.g., Dixon & MacLin, 2003).

Referencias

Bermúdez, K. & Ruiz, J. A. (2018). Efecto de la frecuencia de reforzamiento condicionado sobre la resistencia a la extinción en humanos. Trabajo presentado en el XXVIII Congreso Mexicano de Análisis de la Conducta, Boca del Río, Veracruz, México.

Cabello, F., Barnes-Holmes, D., Stewart, I. & O'Hora, D. (2002). Visual Basic in the Experimental Analysis of Human Behavior: A brief introduction. *Experimental Analysis of Human Behavior Bulletin, 20*, 17-20.

Daugherty, T. K. & Quay, H. C. (1991). Response perseveration and delayed responding in childhood behavior disorders. *The Journal of Child Psychology and Psychiatry, 32*, 453-461. DOI 10.1111/j.1469-7610.1991.tb00323.x

Devarakonda, K., Nguyen K., & Kravitz, A. (2016). ROBucket: A low cost operant chamber based on the Arduino microcontroller. *Behavior Research Methods, 48,* 503-509. DOI: 10.3758/s13428-015-0603-2

Dixon, M. R. & MacLin, O.H. (2003). *Visual Basic for Behavioral Psychologists*. Reno, NV: Context Press.

Escobar, R. & Pérez-Herrera, C. (2015). Low-Cost USB Interface for operant research using arduino and visual basic. *Journal of the Experimental Analysis of Behavior, 103*, 427-435. DOI 10.1002/jeab.135

Escobar, R. & Santillán, N. (2017). Nuevas tecnologías aplicadas a la investigación operante: fotoceldas de bajo costo con la interfaz Arduino. *Revista Mexicana de Análisis de la Conducta, 43*, 242-253. DOI 10.5514/RMAC.V43.I2.62315

Guzdial, M. (2004). Programming environments for novices. En S. Fincher & M. Petre (Eds.) Computer Science Education Research (pp. 127-154). Abington, UK: Taylor and Francis.

Haxby, J., Parasuraman, R., Lalonde, F. & Abboud, H. (1993). SuperLab: General-purpose Macintosh software for human experimental psychology and psychological testing. *Behavior Research Methods, Instruments, & Computers, 25,* 400-405. DOI: 10.3758/BF03204531.

Kalelioğlu, F. & Gülbahar, Y. (2014). The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners Perspective. *Informatics in Education, 13,* 33-50.

Kordaki, M. (2012). Diverse categories of programming learning activities could be performed within Scratch. *Procedia-Social and Behavioral Sciences, 46,* 1162-1166. DOI: 10.1016/j.sbspro.2012.05.267

Maloney, J., Resnick, M., Rusk, N., Silverman, B. & Eastmond, E. (2010). The Scratch Progamming Language and Environment. A*CM Transactions on Computing Education, 10 (4)*, Article 16. DOI: 10.1145/1868358.1868363

Marji, M. (2014). *Learn to program with Scratch: A visual introduction to programming with games, art, science, and math*. San Francisco, CA: No Starch Press, Inc.

Miller, J. R., DeLeon, I. G., Toole, L. M., Lieving, G. A., & Allman, M. J. (2016). Contingency enhances sensitivity to loss in a gambling task with diminishing returns. *The Psychological Record, 66*, 301-308. DOI 10.007/s40732-016-0172-5

Mueller, S. T. (2011). The PEBL Manual. Programming and Usage Guide for The Psychology Experiment Building Language PEBL Version 0.12. Recuperado de: *https://sourceforge.net/p/pebl/code-0/653/tree/trunk/doc/.../PEBLManual0.12.pdf*

Peirce, J. W. (2007). PsychoPy – Psychophysics software in Python. *Journal of Neuroscience Methods, 162*, 8-13. DOI 10.1016/j.jneumeth.2006.11.017

Picanco, C.R. & Tonneau, F. (2018). A low-cost platform for eye-tracking research: using Pupil© in behavior analysis. *Journal of the Experimental Analysis of Behavior, 106*, 1-14. DOI 10.1002/jeab.448

Resnick, M. et al. (2009). Scratch: programming for all. *Communicatiosn of the ACM, 52*, 60-67. DOI 10.1145/1592761.1592779

Ragozzine, F. (2002). SuperLab LT: Evaluation and Uses in Teaching Experimental Psychology. *Teaching of Psychology, 29,* 251-254. DOI: 10.1207/S15328023TOP2903_13.

Ribeiro, M. V., Neto, J. F. R., Morya, E., Brasil F. L. & Pereira de Araújo, M. F. P. (2017). OBAT: An open-source and low cost operant box for auditory discriminative task. *Behavior Research Methods, 50*, 816-825. DOI: 10.3758/s13428-017-0906-6.

Ruiz, J. A. & Bermúdez, K. (2017). El efecto de "casi ganar" sobre la preferencia para jugar entre dos máquinas tragamonedas. Trabajo presentado en el XXVII Congreso Mexicano de Análisis de la Conducta, Aguascalientes, Aguascalientes, México.

Schultz, B. G. (2018). The Schultz MIDI Brenchmarking Toolbox for MIDI interfaces, percussion pads, and sound cards. *Behavior Research Methods,* 1-31. DOI: 10.3758/s13428-018-1042-7.

Taylor, P. J. & Marsh, J.E. (2017). E-Prime (Software). *The International Encyclopedia of Communication Research Methods* (eds. J. Matthes, C. S. Davis y R. F. Potter). DOI: 10.1002/9781118901731.iecrm0085.

Recibido Junio 27, 2018 /
Received June 27, 2018
Aceptado Noviembre 22, 2018 /
Accepted November 22, 2018