Cuaderno para el docente

Actividades para aprender a Program.AR

Segundo Ciclo de la Educación Primaria y Primero de la Secundaria

Presidencia de la Nación

Versión 2.0 | Mayo de 2015

Autores
Pablo Factorovich
Federico Sawady O'Connor

Edición
Ignacio Miller

Diseño Gráfico e ilustración
Alejandro Manfroni
Jaqueline Schaab

Fundación Sadosky

Coordinación Program.AR
Lic. Belén Bonello
Dr. Fernando Schapachnik
Lic. Pablo Factorovich

Educ.ar S.E.

Coordinación Contenidos
Cecilia Sagol

Coordinación Editorial
Inés Roggi

Coordinación Tecnológica
Juan Jolis

Coordinación Diseño
Juan Furlo

Factorovich, Pablo Matías
Actividades para aprender a Program.AR : segundo ciclo de la educación primaria y primero de la secundaria / Pablo Matías Factorovich y Federico A. Sawady O'Connor ; edición literaria a cargo de Ignacio David Miller ; ilustrado por Alejandro Manfroni y Jaqueline Schaab. - 1a ed. - Ciudad Autónoma de Buenos Aires : Fundación Sadosky, 2015.
E-Book. - (Program.AR)

ISBN 978-987-27416-1-7

1. Educación. 2. Capacitación Docente. I. Sawady O'Connor, Federico A. II. Miller, Ignacio David, ed. lit. III. Manfroni, Alejandro, ilus. IV. Schaab, Jaqueline, ilus. V. Título
CDD 371.1

Autoridades

Presidenta de la Nación
Dra. Cristina Fernández de Kirchner

Ministerio de Ciencia, Tecnología e Innovación Productiva
Dr. José Lino Barañao

Ministerio de Educación
Ministro de Educación
Prof. Alberto E. Sileoni

Fundación Sadosky
Director Ejecutivo
Lic. Santiago Ceria

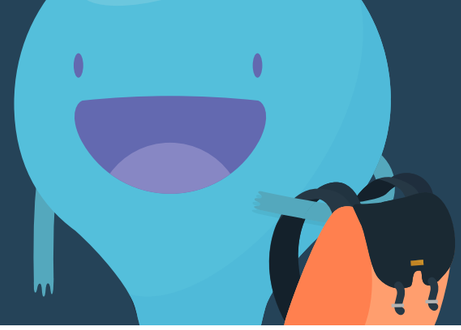
Secretario de Educación
Lic. Jaime Perczyk

Subsecretaría de Equidad y Calidad
Lic. Gabriel Brener

Educ.ar S.E.
Gerente General Educ.ar S.E.
Rubén D'Audia

Coordinación General
Mayra Botta

Prólogo



¿Por qué enseñar programación en la escuela argentina?

Esta pregunta no puede responderse sin antes analizar en el contexto en el que nace Program.AR, la iniciativa del Estado Nacional que se propone llevar la enseñanza y el aprendizaje de las Ciencias de la Computación a la escuela argentina. Este proyecto comenzó su gestación a mediados del año 2013, cuando las agencias gubernamentales que trabajan temas vinculados a educación y tecnología encararon la labor conjunta de analizar de qué manera la computación, su enseñanza y aprendizaje se implementaban en las aulas de nuestro país.

A partir de la creación del Programa Conectar Igualdad en el año 2010, se ha producido una llegada masiva de tecnología a las escuelas argentinas, concretamente de netbooks. Esto estuvo acompañado por la instalación de pisos tecnológicos que las conectan en red y por el desarrollo del Plan Nacional de Telecomunicaciones Argentina Conectada, que construye una plataforma digital de infraestructura y servicios.

La primera etapa del enorme desafío que implica alcanzar una distribución masiva de equipamiento en todo el país puede considerarse completa, más allá de que este requiere un trabajo sostenido en el tiempo y no puede darse por saldado debido a la natural rotación de los estudiantes dentro del sistema educativo y al natural deterioro del equipamiento entregado.

Paralelamente al desafío de la distribución de equipamiento, se encaró la tarea de incorporar el uso de tecnología en los procesos disciplinares de enseñanza y aprendizaje. Si bien en la actualidad existe una enorme multiplicidad de herramientas informáticas para incorporar de manera significativa el uso de tecnología en la enseñanza de diversos campos del conocimiento, esto requiere de una profunda labor de formación docente. Esta tarea se lleva adelante desde el Ministerio de Educación a través de ofertas de capacitación presenciales y virtuales y están, en su mayoría, centradas en la incorporación del uso de las nuevas tecnologías como recursos para el abordaje de contenidos curriculares. La incorporación de las TIC de manera transversal en el sistema educativo tal como la plantea el Programa Conectar Igualdad es fundamental en la construcción de una educación de calidad, actualizada y pertinente a los tiempos en los que se desarrolla.

No obstante lo antedicho, es importante diferenciar claramente entre una persona en condiciones de utilizar de manera hábil una determinada tecnología y una persona que comprende el funcionamiento de la tecnología que utiliza. No es lo mismo usar un buscador que preguntarse (y saber responder) cómo hace éste para encontrar en fracciones de segundo esas pocas páginas coincidentes con la búsqueda entre las miles de millones existentes. ¿Qué es un virus informático? Cuando entramos a una página “segura”, de esas que tienen un candadito en el navegador, ¿son realmente seguras? ¿Por qué? ¿Cómo hace una computadora para reconocer el habla y responder a una pregunta? ¿Cómo hace para poder recomendarnos cosas en base a nuestros gustos y preferencias? ¿Cuando mandamos un mail, cómo llega hasta la otra punta del planeta en segundos? ¿Cómo hacen las redes sociales para sugerirnos nuevos amigos? ¿Por qué es cierto que una computadora de 1 GHz puede ser más rápida que otra de 2 GHz? ¿Y qué es eso de los GHz? ¿Qué le pasa a una computadora cuando se “cuelga”? ¿Qué podemos decir de los robots? ¿Con qué mecanismos van a proteger los estudiantes sus datos personales y su privacidad online? ¿Esperamos que sigan recetas que no pueden analizar críticamente? ¿Cómo tomarán posición sobre el voto electrónico? ¿Sabrán deconstruir las opiniones de los “expertos”?

Estas preguntas son sólo un ejemplo de muchas otras que se plantean cuando se aplica una mirada curiosa al mundo actual. Y de cada una de ellas se ramifican varias otras que van apareciendo a medida que comenzamos a profundizar en la búsqueda de respuestas. En definitiva, se trata de explorar distintos aspectos de las Ciencias de la Computación.

El presente material inaugura una nueva línea de trabajo, complementaria a la anterior, ya que se toma a las nuevas tecnologías como el objeto de estudio y no como una herramienta al servicio de otros aprendizajes.

La educación argentina se enfrenta en la actualidad a los desafíos de la inclusión y la calidad: cómo incorporar a los sectores tradicionalmente excluidos a la escolaridad brindándoles herramientas y saberes socialmente pertinentes a través de una educación de calidad. Adicionalmente, la escuela tiene la misión de formar ciudadanos de pleno derecho, dotados de la capacidad de comprender, analizar y criticar el mundo en el que viven. La Iniciativa Program.AR sostiene que el mundo moderno no puede comprenderse cabalmente sin contar con un conjunto de herramientas que permitan decodificar la lógica de la tecnología que media en buena parte los vínculos que establecemos con el mundo en el que estamos insertos.

Para comprender y problematizar los saberes vinculados a la tecnología no alcanza con simplemente conocerlos, nombrarlos o estar en constante contacto con dispositivos tecnológicos. Las lógicas de funcionamiento de la tecnología no se revelan con su mero uso.

Luego de los primeros encuentros del equipo de trabajo de Program.AR resultó evidente que esta problemática requería del involucramiento de la sociedad en general y de diversos actores clave del sistema educativo tales como la comunidad docente, las autoridades educativas, la academia y el mundo del trabajo en particular. Es por esto que durante 2014 se llevaron adelante Foros Regionales de Debate en diversas Universidades Nacionales (Universidad Nacional de Quilmes, Universidad Nacional de Córdoba, Universidad Nacional del Nordeste, Universidad Nacional de Cuyo y Universidad

Nacional de la Patagonia Austral) y en 2015 el debate continuará recogiendo distintas voces en todas las regiones de nuestro extenso territorio.
El debate que proponemos se encuentra en apogeo en distintos países del mundo. Podríamos reseñar experiencias interesantes, también podríamos detallar buenas y malas prácticas llevadas adelante en tierras lejanas. Sin embargo, aún no existe un modelo probado y considerado unánimemente exitoso, esto puede deberse a que las Ciencias de la Computación y su didáctica son disciplinas jóvenes, terrenos en plena construcción. Asimismo, aún cuando existiera un modelo exitoso, el consenso acerca de la inconveniencia de importar un diseño curricular o una experiencia extranjera de manera acrítica ya es absoluto. El desafío implica pensar y diseñar estrategias que partan de nuestras particularidades como nación soberana.
Durante el desarrollo de los foros de debate hemos realizado talleres sobre programación y robótica para adolescentes, cuyas experiencias y actividades se recogen en este cuadernillo, pensado como una herramienta para los docentes de nivel primario que deseen llevar este tipo de enseñanzas a sus clases. Profundizando este sentido, durante 2015 se comenzará a trabajar en la capacitación de los docentes a través de diversas Universidades Nacionales. Encarar estos desafios de manera paralela a la difusión, el debate y la construcción de una comunidad de conocimiento resulta fundamental para abarcar el problema de manera integral.
Para encarar el enorme desafío de actualizar y profundizar los contenidos vinculados a la computación es necesario invertir esfuerzos en la capacitación de los docentes. Desarrollar experiencias de capacitación que generen capacidades a nivel local resulta fundamental para sostener un criterio federal y respetar la autonomía de cada provincia al momento de incorporar en la enseñanza local rasgos culturales propios de cada región. Este material forma parte de la línea de trabajo vinculada a la formación, que no sólo capacita formadores sino que también busca disponibilizar material para que los docentes puedan llevar al aula este nuevo enfoque para la enseñanza de informática. Este cuadernillo está orientado a docentes del último ciclo del nivel primario.
Nos enfrentamos al desafío de construir una escuela moderna y en condiciones de interpelar a los alumnos a través de un uso profundo y significativo de la tecnología. Esta tarea requiere de una actualización permanente y de un constante desafío al cuerpo docente. Indudablemente, el éxito en estos objetivos tributará a la edificación de la educación inclusiva y de calidad por la que todos trabajamos.

El equipo de Program.AR

fundación SADOSKY Investigación y Desarrollo en TIC
educ.ar
CONECTAR IGUALDAD_ WWW.CONECTARIGUALDAD.GOB.AR
Presidencia de la Nación | Ministerio de Ciencia, Tecnología e Innovación Productiva
Presidencia de la Nación | Ministerio de Educación
ANSES

Índice

Introducción

La programación es una de las áreas más importantes de las ciencias de la computación. Estas abordan el estudio de las computadoras desde los principios lógicos y formales que hacen posible su funcionamiento, es decir, más allá de los detalles que presentan las tecnologías concretas que utilizamos a diario. Como disciplina, la programación está orientada al desarrollo de una serie de habilidades de abstracción y operacionalidad. El primer tipo de habilidades incluye técnicas como la simplificación de problemas, la definición de soluciones generales aplicables a problemas similares y la asignación de nombres significativos a las distintas partes de una solución. El segundo tipo de habilidades, las operacionales, supone la definición de soluciones en términos de un conjunto de pasos que deben ejecutarse en un orden determinado para alcanzar un objetivo.

El presente Cuaderno pretende ser una herramienta que sirva de guía a los docentes en la enseñanza de algunos de los principios básicos de programación a los alumnos del segundo ciclo de la educación primaria y primero de secundaria. Está organizado en cinco partes, en cada una de las cuales se aborda un concepto fundamental de programación, a través de actividades ideadas específicamente.

La primera parte se centra en la forma en que se comportan las computadoras al ejecutar programas y en la manera en que se deben expresar las instrucciones al programar. Además, en esta parte se introducirá el concepto de repetición, que permite reducir de forma considerable el código escrito. La segunda parte trata el uso de alternativas condicionales, es decir, instrucciones cuya ejecución está supeditada al cumplimiento de una condición, lo que posibilita la inclusión de la toma de decisiones en los programas, de manera que el funcionamiento de estos dependa de cierta información sobre el contexto en el que se ejecutan. La tercera parte recupera y reformula los conceptos de repetición y condición trabajados en las partes anteriores a través de un nuevo tipo de repetición: la repetición condicional. La cuarta parte aborda la realización de programas más generales mediante un mecanismo conocido como parametrización. Por último, la quinta parte, importante para motivar y concretizar el aprendizaje de los alumnos, procura indagar en algunos aspectos de la interactividad por medio de la programación de juegos.

Para facilitar la planificación y el trabajo en clase, los contenidos se presentan en forma de secuencias didácticas, en las que la realización de actividades se integra con la problematización de los conceptos involucrados en ellas; a esto sigue un grupo de actividades que funcionan como ejercicios, que permiten reforzar y profundizar lo aprendido. Como parte de la tarea de planificación, queda a criterio de cada docente la determinación de la cantidad de clases que requiere el desarrollo de cada secuencia didáctica; no obstante, de manera general, convendrá tener en cuenta que, en la mayoría de los casos, serán necesarias al menos dos clases por secuencia.

Propósitos generales

- Propiciar la reflexión acerca de la utilidad de los programas para representar ideas y resolver problemas.
- Indagar en la noción de que las computadoras sirven para ejecutar programas y realizan lo que el programa indique.
- Incentivar la creación de programas por parte de los alumnos, de manera que no se limiten a ser usuarios de aplicaciones realizadas por terceros.
- Estimular la confianza de los alumnos mediante el uso y la ejecución de programas diseñados por ellos mismos.
- Promover la reflexión crítica y el trabajo colaborativo a través de la detección y corrección de errores de los programas propios y ajenos.
- Trabajar con conceptos relacionados con las ciencias de la computación para desarrollar habilidades de pensamiento computacional.

Contenidos

- Noción de programa y autómata. Comandos (acciones) y valores (datos).
- Estrategia de división en subtareas. Planificación de la solución de un problema de programación. Identificación de subproblemas. Procedimientos. Denominación y síntesis del objetivo de un procedimiento.
- Identificación de patrones.
- Parámetros.
- Repeticiones simples. Alternativas condicionales. Repeticiones condicionales.
- Uso de las herramientas Scratch y Lightbot.
- Procesamiento de estructuras lineales y diseño de actividades de programación típicas, incluyendo la selección de operaciones primitivas del sistema de cómputo elemental o autómata.
- Metodología para la corrección de errores del programa a partir del análisis de la diferencia entre lo que se espera del programa y lo que este efectivamente hace.

Estrategias didácticas

- Actividades individuales y de a pares, con y sin computadoras.
- Exposiciones breves del docente.
- Socialización en clase de las soluciones de los ejercicios.

Estrategias de evaluación

- Resolución de actividades.
- Participación en clase.
- Exposición de soluciones de ejercicios.

Herramientas

- Actividades sin computadora.
- Scratch (2.0).
- Lightbot 1.

Sobre Scratch

Scratch es un lenguaje de programación diseñado para el aprendizaje de las herramientas básicas de programación. Los usuarios pueden crear sus propias historias interactivas, animaciones, juegos y música, y compartirlas en la web. Se puede acceder a él y utilizarlo gratuitamente ingresando en **http://scratch.mit.edu**
En la propuesta que aquí se presenta, se trabajará a partir de actividades creadas ad hoc en Scratch. En cada actividad se plantea un problema específico, que los alumnos deberán resolver programando. Copiando en la barra del navegador la dirección que se indica en cada caso (o haciendo clic sobre el título de la actividad de Scratch, en la versión digital de este *Cuaderno*), se puede acceder a una versión *on line* de la actividad.
Si bien todas las actividades de Scratch se diseñaron utilizando los personajes, objetos y fondos que provee el sitio, se debe tener presente que se pueden reemplazar por otros sin que ello implique una modificación de la actividad misma. En muchas ocasiones resultará conveniente cambiar algunos de esos elementos, en particular los personajes, eligiendo otros con los que están más familiarizados los alumnos con el fin de incentivar el aprendizaje.*

Sobre Lightbot

Lightbot es un juego de ingenio en el que el usuario debe programar para pasar de un nivel al siguiente. La dificultad va aumentando a medida que se pasa de nivel. Se utilizará esta herramienta para que los alumnos den sus primeros pasos en el desarrollo de programas que resuelven problemas. Deberán secuenciar acciones y, dado que el espacio para programar es limitado, abstraer subtareas a partir de tareas que se repiten.

* Para saber cómo reemplazar o modificar los personajes de las actividades de Scratch, consúltese la página 115 de este cuaderno.

<Program.AR/>

AUTÓMATAS COMANDOS Y PROCEDIMIENTOS

En esta parte se trabajará sobre algunas nociones elementales de programación. Al respecto, interesa destacar que un programa puede ser entendido como un conjunto de instrucciones (correspondientes a comandos) destinadas a ser ejecutadas por un autómata: la computadora. De manera más específica, estas instrucciones son descripciones de soluciones a problemas. Programar implica, por lo tanto, delimitar problemas para poder luego formular las soluciones adecuadas. Para ello, se propone emplear una estrategia llamada *división en subtareas*, que consiste en dividir una acción en otras acciones más pequeñas, incluidas en aquella.

Secuencia didáctica 1

Las computadoras hacen todo al pie de la letra

En esta primera secuencia didáctica no se utilizarán computadoras, sino que, a partir de la representación de las reglas de los programas informáticos, se trabajará con los alumnos en el tipo de tareas que realiza una computadora y se prestará atención al modo en que se deben indicar esas tareas.

Objetivos

- Comprender el carácter específico de las instrucciones empleadas en programación.
- Distinguir entre instrucciones generales y específicas.
- Introducirse en algunos de los conceptos propios del ámbito de la programación.
- Establecer relaciones entre el orden de las acciones programadas y los resultados obtenidos.

Desarrollo

Actividad

Como primer acercamiento para que los alumnos reflexionen acerca del modo en que trabajan las computadoras, se les puede proponer que imaginen que el docente es un autómata, es decir, una máquina que (como las computadoras) sigue al pie de la letra las indicaciones que se le dan.
Los alumnos deberán darle, con precisión y en el orden adecuado, las instrucciones básicas para realizar una tarea y cumplir un objetivo; por ejemplo, dirigirse, desde un rincón del aula, hacia la puerta y salir. En este caso, algunas de esas instrucciones básicas (llamadas **primitivas**) podrían ser las siguientes: "*dar un paso hacia delante*", *"girar hacia la derecha", "girar hacia la izquierda", "abrir puerta".*

Actividad

En un segundo momento de la clase, se les puede sugerir a los alumnos que piensen y formulen acciones acotadas, para que las realice algún compañero; por ejemplo: *"levantar brazo derecho"*, *"pestañear"*, *"mirar a un determinado compañero"*, "*levantarse de la silla"*, etcétera. A continuación, se propondrá a un alumno que, en el rol de autómata, ejecute una serie de instrucciones, como la siguiente: "*levantarse de la silla", "levantar el brazo derecho", "cerrar los ojos", "abrir la boca", "sentarse en la silla".*
Al término de estas actividades, se puede introducir la noción de **comando**, como la instrucción que un programador o usuario le da a una computadora, y hacer que los alumnos reflexionen acerca de la especificidad de las instrucciones formuladas. Las siguientes consignas pueden servir para orientar la reflexión.

Actividad

1. ¿Qué pasaría con el autómata si las acciones que se le hubieran indicado fueran muy generales o muy complejas (por ejemplo, "caminar" o "dibujar una casa")?, ¿las podría haber ejecutado?

2. ¿Se modificó el resultado final de la secuencia al cambiar el orden de las acciones?

3. ¿Qué ocurriría en los siguientes casos?

- Se le ordena al autómata que se siente en una silla, pero no hay ninguna silla.
- Se le dice al autómata que levante el brazo derecho y ya lo tiene levantado.
- Se le pide al autómata que levante el brazo derecho mientras tiene levantado el brazo izquierdo, y no se especifica si antes debe bajar este brazo o no.

Para profundizar en el carácter específico de las instrucciones que se utilizan en programación, el docente asumirá nuevamente el rol de autómata y le solicitará a un alumno que lo guíe paso a paso para que vaya al pizarrón y escriba una palabra determinada (por ejemplo, el nombre del alumno). El docente/autómata solo podrá escribir la palabra cuando ésta le sea dictada letra por letra, pero esto no se le dirá al alumno al comienzo, sino luego de que haya realizado varios intentos. De este modo, por ejemplo, si el alumno dicta el nombre todo junto, se le contestará "*No entiendo*"; si el docente/autómata no está en el pizarrón, ante la solicitud del alumno de que escriba, dirá "*No puedo escribir en el aire*".

Cierre

Como cierre de esta secuencia, se sugiere trabajar con los alumnos en la asignación de nombres (lo suficientemente descriptivos y específicos) a diversas tareas que luego combinarán para formar distintas secuencias. En este sentido, se les puede pedir que describan los pasos que siguen al utilizar un programa, desde encender la computadora hasta la ejecución del comando que abre el programa. También se puede proponer que identifiquen, entre las actividades que realizan cotidianamente, las que involucran una serie de pasos o acciones y que indiquen el nombre de cada acción (por ejemplo, la actividad *"cepillarse los dientes"* involucra las subtareas *"agarrar el cepillo"*, *"tomar el dentífrico"*, *"aplicar el dentífrico sobre el cepillo"*, entre otras) e, inversamente, que agrupen varias actividades en otras sucesivamente más abarcadoras (en el caso de "cepillarse los dientes", "asearse" y esta, a su vez, en, por ejemplo, *"vivir un día entero"*).

Secuencia didáctica 2

Presentación de Scratch

En esta secuencia se trabajará con Scratch mediante un proyecto diseñado en ese lenguaje, que permite programar un autómata (representado en este caso por un gato). Las instrucciones u órdenes que debe realizar el autómata se organizan gráficamente en una serie de bloques. Cada bloque corresponde a una acción que debe realizar; cuando se los encastra verticalmente, forman una **secuencia de comandos**, es decir, una tarea que se hace en un orden establecido desde el primer bloque hasta el último (visualmente, desde arriba hacia abajo).

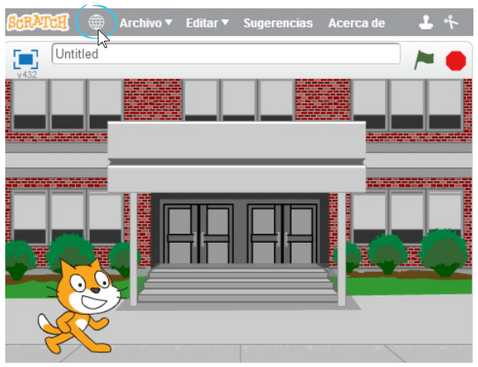
Objetivos

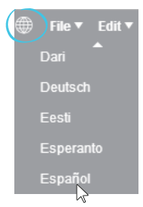
• Comprender que un comando es una acción que genera un efecto (reproducir un sonido, pintar a una imagen, mover un objeto, etcétera) y que las secuencias de comandos permiten encadenar (o componer) comandos de forma ordenada.
• Comprender que un programa es una secuencia de comandos y que ejecutar o *correr* un programa consiste en hacer que el autómata produzca un efecto al interpretarlo.
• Organizar secuencias de comandos ajustadas a un fin específico.
• Distinguir entre instrucciones primitivas y procedimientos.
• Identificar tareas nuevas y crear los procedimientos adecuados.

Desarrollo

http://scratch.mit.edu/projects/42291768/#editor

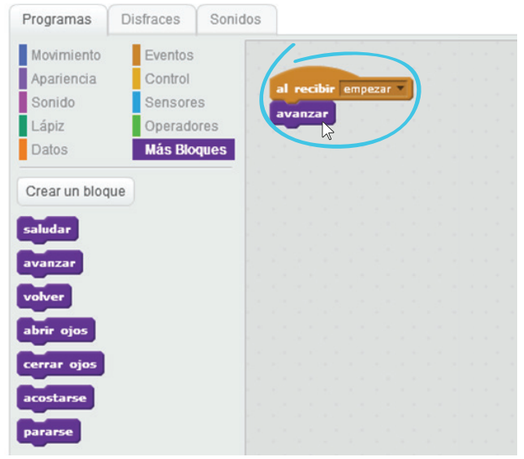
Luego de presentar Scratch, se les indicará a los alumnos que ingresen al programa *El gato en la calle*, disponible **on line***. Si está en inglés, se puede cambiar el idioma de la interfaz, haciendo clic en el botón que se encuentra en el menú de la parte superior de la pantalla.





* TAMBIÉN PUEDEN BAJARSE TODOS LOS ESCENARIOS DEL SIGUIENTE LINK Y USARSE SIN CONEXIÓN A INTERNET:
http://programar.gob.ar/descargas/actividades.zip

En esta primera parte, se sugiere trabajar solo con los bloques de acciones de la opción *Más bloques*, que se encuentra en la pestaña *Programas*. Haciendo clic en esa opción, se accede a la lista de acciones. Para que el autómata ejecute una acción, se arrastra el bloque correspondiente hacia el editor (el sector de la derecha) y se lo coloca debajo del bloque que dice *al recibir empezar.* Por ejemplo, en este caso, los alumnos seleccionarán el bloque *avanzar.*



A continuación, seleccionarán sucesivamente los bloques *saludar y volver*. El primer bloque irá debajo de *avanzar* y el segundo debajo de *saludar*. Así, se formará una secuencia de tres acciones para que las realice el gato.

Haciendo clic en la banderita verde que se observa en la parte superior de la pantalla se ejecutará el programa. El botón rojo sirve para detener el programa en cualquier momento. Para volver a ejecutar la secuencia, basta con hacer clic nuevamente en la banderita verde.

Actividad

Luego de explicar cómo formar secuencias de acciones en el programa, se les puede solicitar a los alumnos que organicen otra secuencia utilizando los bloques anteriores; por ejemplo:

Para insertar un bloque, se lo arrastra desde el menú y se suelta entre los bloques en el lugar en el que se lo quiere agregar. Si se quiere eliminar un bloque (o varios) de la secuencia, simplemente hay que arrastrarlo con el mouse hacia la izquierda, al sector de la opción *Más bloques*, donde se encuentra la lista de los bloques.
Los alumnos también podrán explorar otros modos de ordenar los bloques anteriores; por ejemplo: *saludar, avanzar, volver, saludar, avanzar o avanzar, saludar, avanzar, saludar, volver*. De este modo, observarán que distintas secuencias producen resultados diferentes.

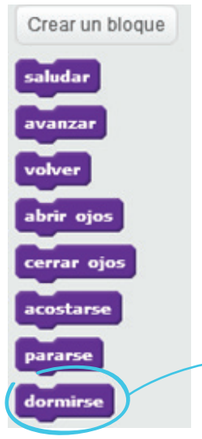
Actividad

Tras estas primeras aproximaciones, se puede abordar la organización de una secuencia formada por otras acciones; por ejemplo, la secuencia correspondiente a la tarea de dormirse. Para ello, en la misma opción *Más bloques*, se dispone de otros bloques:

De entre estos bloques, los alumnos seleccionarán los que consideren adecuados para armar la secuencia. Se espera que elijan el segundo y el tercer bloque, de manera que la secuencia *dormirse* se visualice así:

Hasta ahora, se han utilizado bloques que corresponden a acciones primitivas, es decir, acciones básicas preestablecidas, pero también pueden crearse nuevos bloques, es decir, nuevas acciones. A estas nuevas acciones se las llamará **procedimientos**. Por lo tanto, los procedimientos son nuevas acciones que el usuario le explica a la computadora cómo realizarlas.
Siguiendo con el ejemplo anterior, puede crearse un nuevo bloque que represente la actividad de dormirse y que pueda ejercitarse directamente. Para ello, se presiona, dentro de la opción *Más bloques*, el botón *Crear un bloque*. Al hacerlo, se abre una ventana, con un bloque en blanco, en el cual se escribe el nombre del nuevo bloque (en este caso, *dormirse*).

Una vez que se ha completado el nombre del bloque, se presiona *"OK"*. En el editor aparece un nuevo bloque.



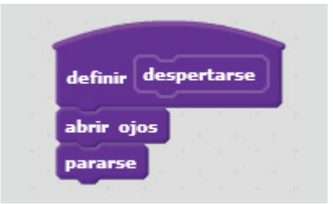
En el nuevo bloque se indica que es necesario definir en qué consiste la nueva tarea. Si no se indica nada, el gato no hará nada cuando se le pida ejecutar el bloque *dormirse*. Los alumnos pueden comprobar que esto es así, arrastrando el bloque *dormirse* de la lista de la izquierda, ubicándolo debajo del bloque *al recibir empezar* y haciendo clic en la banderita verde. El gato no efectuará ninguna acción.

Actividad

Teniendo en cuenta lo realizado, los alumnos están en condiciones de definir las acciones que forman parte del procedimiento *dormirse*, por lo que se sugiere dejar que lo hagan ellos mismos. Simplemente, deberán arrastrar los bloques *acostarse y cerrar ojos* de la lista *Más bloques* y ubicarlos debajo del bloque *definir dormirse*.

Actividad

Ahora que los alumnos saben cómo definir nuevos bloques de acciones, se les puede pedir que creen y definan un nuevo procedimiento, al que llamarán *despertarse*, que haga que el gato abra los ojos y se levante. La definición del procedimiento debería visualizarse así:



Cierre

Como cierre se les puede solicitar a los alumnos que armen y ejecuten la secuencia de acciones necesarias para hacer que el gato avance un paso, se duerma, se despierte, salude y vuelva a su lugar. Esto debería ser resuelto así:

Finalmente, resulta útil hacer un repaso de todo lo aprendido hasta el momento.

- Los comandos representan acciones y, si los ejecutamos, la computadora realiza dichas acciones.
- Las secuencias de comandos son series de acciones que se realizan una a continuación de la otra, en un orden determinado.
- Un programa es una descripción que le damos a la computadora para que realice lo que le indicamos.
- Los comandos incluidos originalmente en el autómata se llaman **primitivos**.
- Se pueden definir nuevos comandos, que llamamos **procedimientos**, para explicarle a la computadora cómo realizar nuevas acciones.

Secuencia didáctica 3

Presentación de Lightbot

Hasta el momento, se han realizado actividades de índole recreativa, que permitían elegir libremente entre diversas acciones y combinarlas de varias maneras. Así ocurría con los bloques en Scratch. En esta secuencia didáctica, se comenzarán a utilizar los bloques para resolver problemas. De este modo, frente a un problema determinado, habrá una secuencia de bloques que será una solución y otras secuencias que no lo serán. Para introducirse en el tema, se propone el trabajo con Lightbot, un videojuego de ingenio en el que los alumnos deberán programar la solución para que el autómata –en este caso, un robot– pase de un nivel al siguiente. El objetivo es lograr en cada nivel que el robot logre prender la o las luces ubicadas en los cuadrados azules del piso. Para ello, el usuario cuenta con una serie de acciones en forma de íconos (similares a los bloques de Scratch) y una grilla en la que ubicarlas. El juego ofrece, además, la posibilidad de definir dos procedimientos (denominados **funciones**), aunque en los primeros niveles no es necesario utilizarlos.

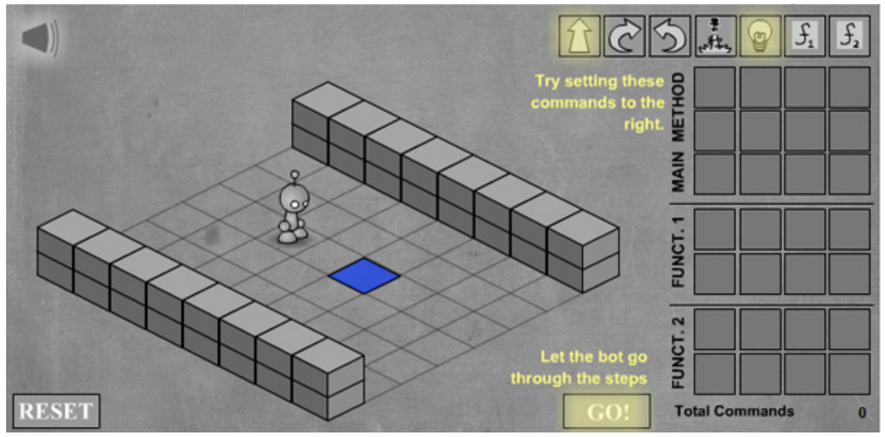
Objetivos

- Inferir que los programas pueden resolver problemas específicos.
- Identificar problemas y proponer soluciones a través de la programación.
- Reconocer el patrón que permite descomponer un problema en acciones más acotadas.
- Resolver problemas de manera más simple a partir del patrón que permite descomponerlos.
- Proponer diversas soluciones para un mismo problema.

Desarrollo

http://armorgames.com/play/2205/light-bot

Luego de una breve introducción, se indica a los alumnos que ingresen a Lightbot, disponible **on line**. Una vez que se ha terminado de cargar el juego, se accede a una pantalla de presentación. En el texto en inglés que allí se encuentra se explica que la inteligencia artificial es una tarea de programación compleja y se aclara que no siempre los robots diseñados pueden manejarse y funcionar por sí mismos: antes bien, algunos robots funcionan dentro de una trayectoria predeterminada por el programador, que varía de una situación a otra. Por último, se señala en qué consiste el juego: utilizando una serie de comandos, se debe lograr que el robot encienda todas las baldosas azules de la fábrica. Para empezar a jugar, hay que hacer clic en la opción *New Game.*



Los comandos primitivos que presenta el juego son los siguientes:

Avanzar (hace que el robot avance un casillero, excepto si está frente a una pared o un desnivel, en cuyo caso permanece en el mismo lugar)

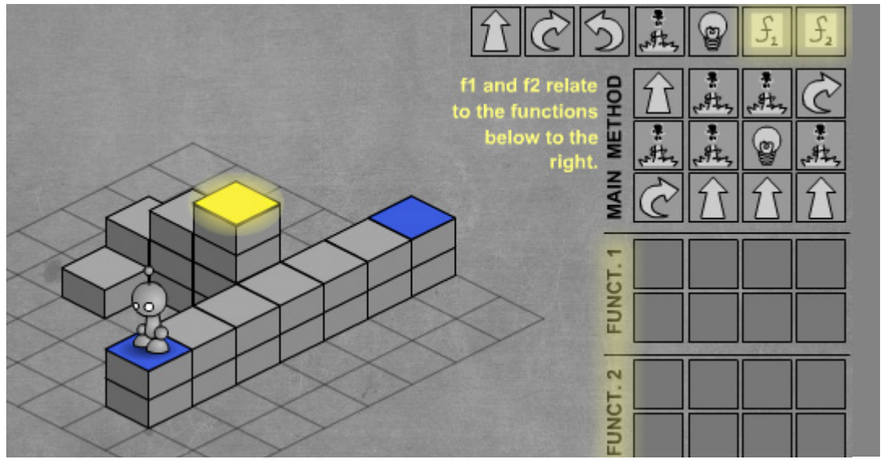
Girar a la derecha **Girar a la izquierda**

Saltar (permite al robot saltar por encima de un solo bloque, o bien saltar de uno)

Encender luz (si está apagada) o **Apagarla** (si está encendida)

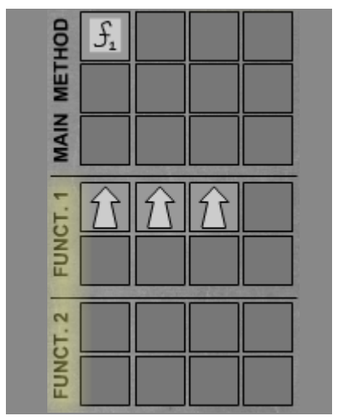
Actividad

Se sugiere dejar a los alumnos que avancen hasta el nivel 6 del juego aclarándoles que utilicen solo la primera de las tres grillas –la titulada *Main method* (“Procedimiento principal”)– y ayudándolos cuando sea necesario. Los alumnos que terminen primero pueden orientar a los compañeros que se demoran más, pero sin resolverles la tarea. Es relativamente sencillo avanzar hasta el nivel 5 inclusive; al llegar al nivel 6, es necesario comenzar a utilizar las subtareas que deja definir el juego (las grillas correspondientes a los procedimientos o funciones), porque no alcanzan las casillas disponibles en la primera grilla. Es recomendable dejar que los alumnos intenten resolver el nivel 6 sin usar subtareas, de manera que ellos mismos comprueben que esto no es posible.

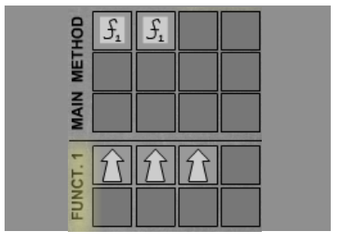


En este punto, se retomará el concepto de **procedimiento** (o **función**, como se lo llama en este juego), ya trabajado en la secuencia didáctica anterior. Así como en Scratch se podían crear nuevos bloques de actividades (como *dormirse* o *levantarse*), cada uno de los cuales involucraba varios pasos o acciones (como *acostarse* y *cerrar ojos*, para el caso de dormirse), en este juego se permite definir dos tareas nuevas delimitadas por los grupos de casilleros *Funct[ion] 1* y *Funct[ion] 2*. Cada grupo admite hasta ocho acciones por tarea ya que es parte del juego que el jugador piense la forma de utilizar las casillas de la manera más eficaz.

Mediante los comandos f_1 y f_2 se indica al robot que ejecute las nuevas tareas definidas. Así, por ejemplo, si el procedimiento o función 1 se define como *avanzar tres veces*, al colocar el comando f_1 en la primera grilla, se está indicando al robot que se mueva tres pasos hacia adelante:



Si se especifica dos veces f_1, el robot se moverá seis veces hacia adelante. De este modo, en vez utilizar seis casilleros de la grilla, se utilizan solo dos ya que cada que cada f_1 vale por tres pasos:

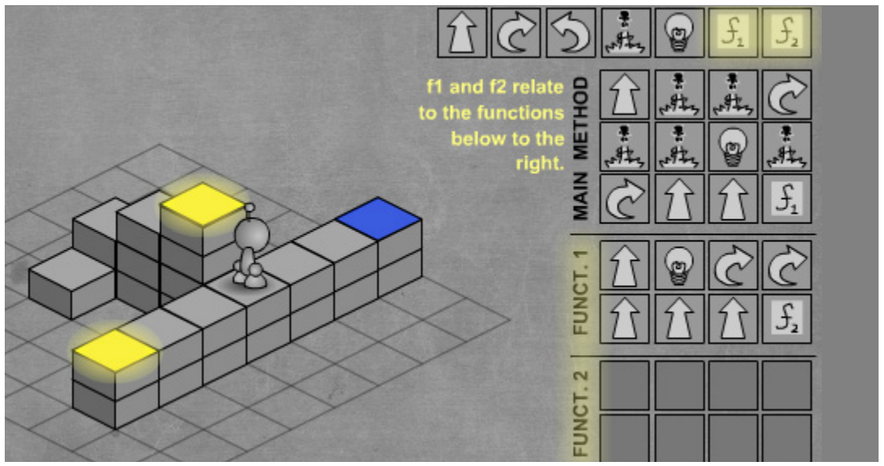


Actividad

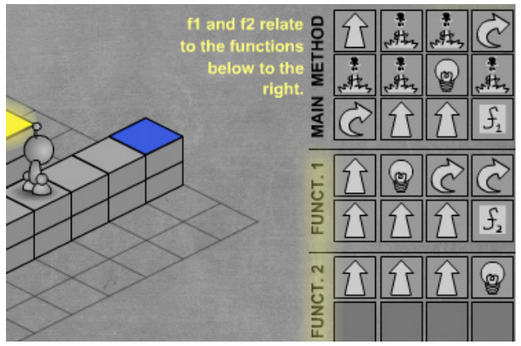
A partir de ahora, interesa sobre todo que los alumnos puedan identificar patrones que les permitan ahorrar casilleros mediante la definición de nuevas tareas. Con el fin de reforzar la continuidad de estas operaciones con las ya practicadas en Scratch, se les puede indicar a los alumnos que propongan un nombre para la tarea ejemplificada en f_1 lo suficientemente descriptivo de la acción que permite realizar.

Actividad

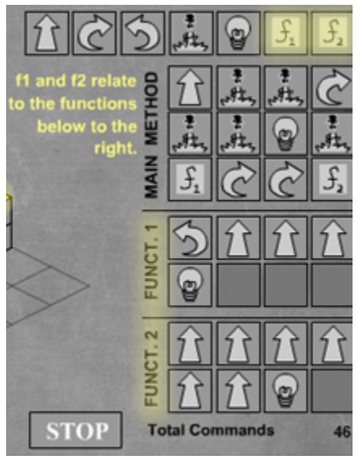
Una vez que se ha explicado cómo crear nuevos procedimientos en Lightbot, los alumnos están en condiciones de solucionar el problema que se plantea en el nivel 6. Al respecto, una opción, seguida por muchos alumnos, para continuar indicando acciones al robot, consiste en definir la función 1 con el detalle de las acciones que no entran en la primera grilla:

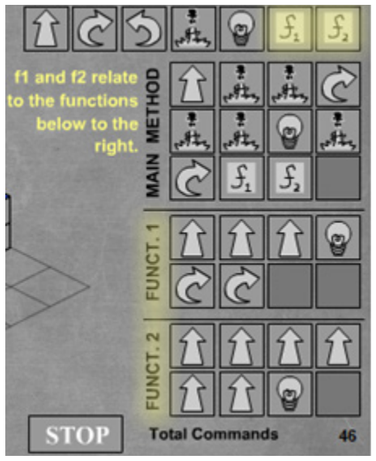


Debido a que las casillas de la función 1 no alcanzan para indicar todas las acciones, se incluyen entonces en la función 2 las acciones restantes:



En esta solución no se pensó en dividir inteligentemente el problema. Si bien el robot llega a encender todas las luces, esta forma de utilizar las subtareas no permitirá ir muy lejos en los niveles más complejos puesto que no serán suficientes todas las casillas disponibles. Otras soluciones más adecuadas (aunque es menos probable que los alumnos las propongan) pueden ser las siguientes:

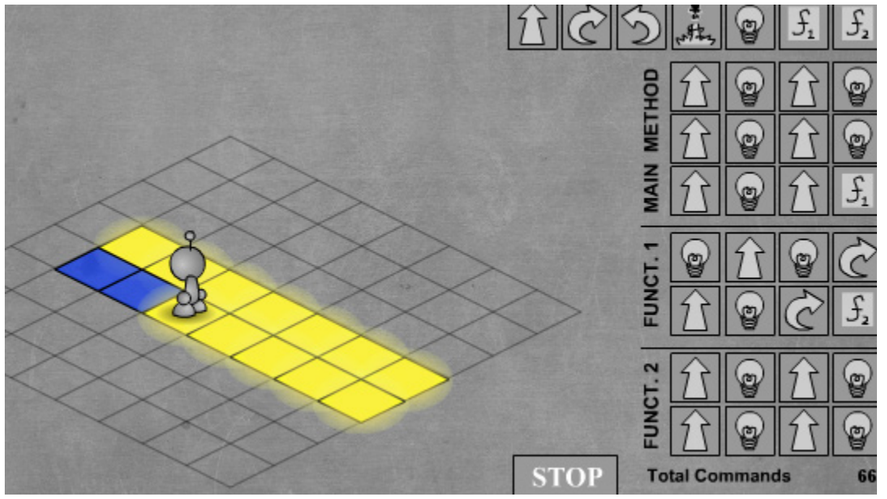




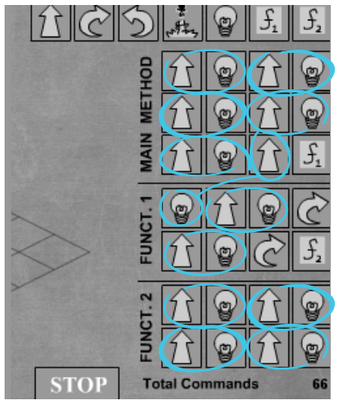
Según las características de la clase, puede ser conveniente dejar que los alumnos exploren diversas soluciones posibles (reiniciando el juego, si es necesario). De todos modos, por el momento, bastará con señalar las limitaciones que presenta la primera solución y pasar al nivel siguiente.

Actividad

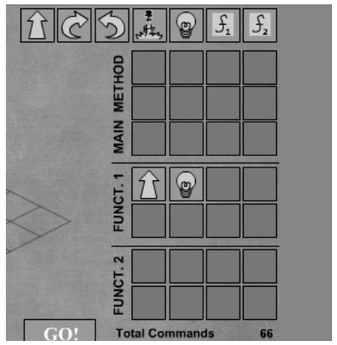
Al llegar al nivel 7, se sugiere darles a los alumnos un tiempo para que intenten resolverlo por su cuenta. Lo más probable es que la mayoría encare una vía de solución inadecuada, intentando repetir lo hecho en el primer ejemplo del nivel anterior, por lo cual quizás resulte conveniente pedirles que interrumpan el juego y se reenfoquen en cómo usar los procedimientos para hallar una solución satisfactoria. Se puede comenzar señalando lo que ocurre cuando se utilizan los procedimientos solo como meras extensiones del espacio disponible para llenarlos con acciones primitivas. En este caso, se llega a una situación como la siguiente:



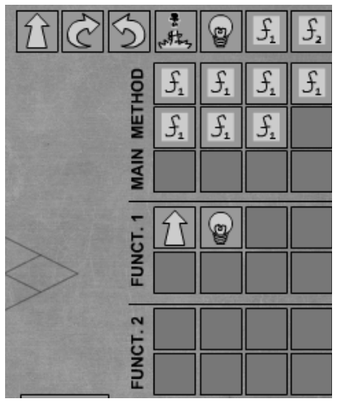
Se han ocupado todos los casilleros y al robot todavía le quedan dos baldosas sin encender. Para que tenga sentido utilizar un procedimiento, tiene que cumplir con un propósito bien definido y estar formado por secuencias de acciones que se repiten muchas veces dentro del programa. En relación con esto, se les puede pedir a los alumnos que busquen, en el intento de solución planteado, alguna secuencia que se repita. Esta secuencia es la conformada por las acciones *avanzar* y *encender luz*.



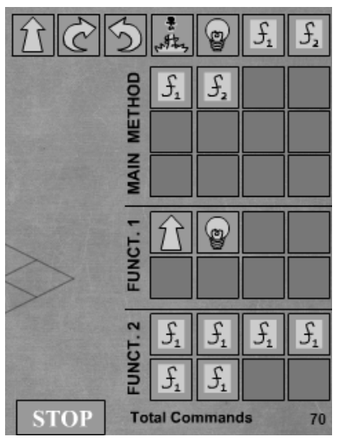
Esta observación constituye un buen punto de partida para que los alumnos profundicen en la utilidad de los procedimientos en programación. Así, siguiendo con el análisis del ejemplo, las secuencias que se reiteran pueden borrarse y ser redefinidas como subtareas de un mismo procedimiento:



De este modo, cada vez que se quiera que el robot avance y encienda la luz, en las casillas de la grilla *Main method* se indicará *f1*:

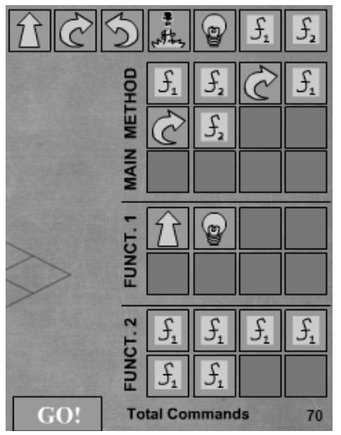


Lo que se ha hecho para la primera fila de luces, puede también aplicarse a la segunda. Para eso, solo hay que definir otro procedimiento que incluya el anterior. De acuerdo con esto, *f2* se puede definir como seis veces *f1*.



¿Y por qué *f2* se define como seis veces *f1* y no como *siete veces* (tal como se hizo al completar las acciones que el robot debía ejecutar en la primera fila)? Porque *f2*, al estar basada en *f1*, hace que el robot encienda ciertas luces partiendo de *la celda en la que se encuentra en la primera fila*. No es posible que el robot se ubique delante de

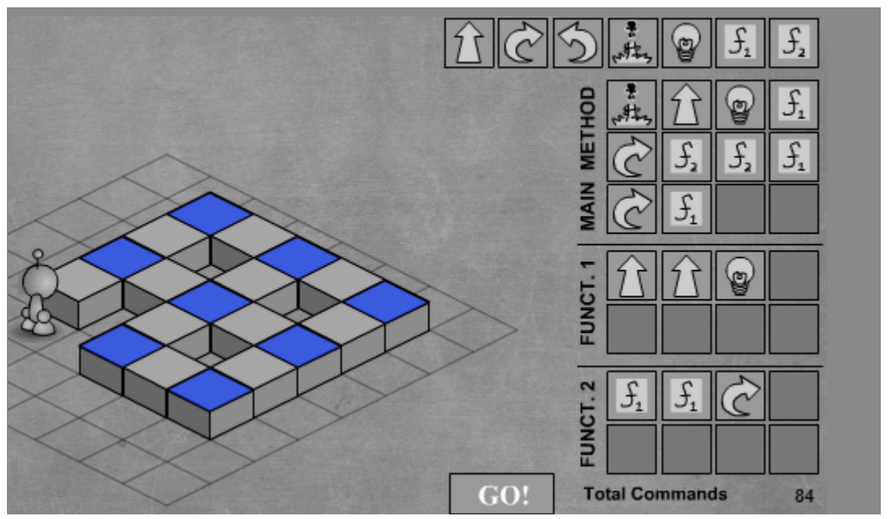
la primera celda en la segunda fila, ya que esta comienza sobre el borde del tablero. En cualquier caso, si a los alumnos les resulta difícil entender esto, bastará con que hagan la prueba: verán que al hacer *f2 = f1* el robot apaga la luz de una baldosa que ya había encendido. Las indicaciones para que el robot encienda todas las baldosas quedan organizadas de la siguiente manera:



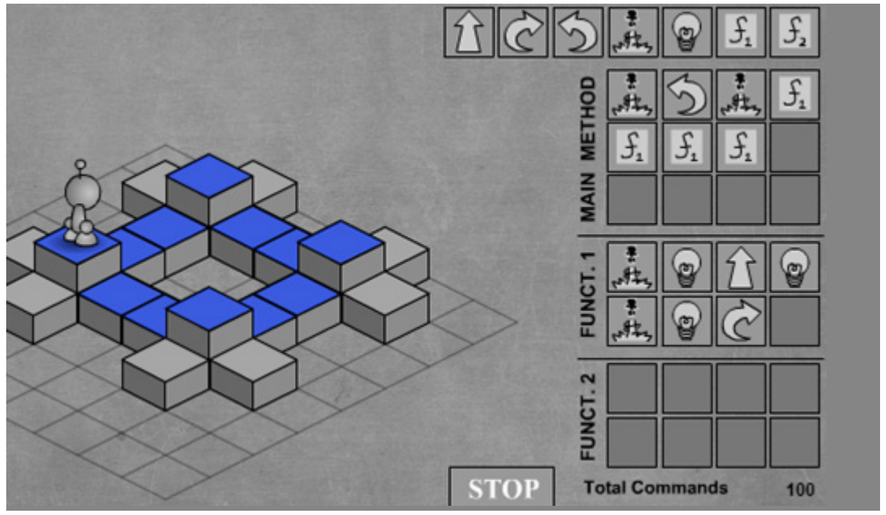
Los próximos niveles requieren usar esta habilidad para ser resueltos. Antes de pasar a ellos, es fundamental que los alumnos se apropien de este nuevo modo de usar los procedimientos. De ser necesario, se dejará que organicen sus propias definiciones de los procedimientos, las prueben y las corrijan. Incluso puede que propongan otras soluciones que, aunque les permitan pasar al siguiente nivel, no sean tan *eficientes* como la descripta arriba.

Actividad

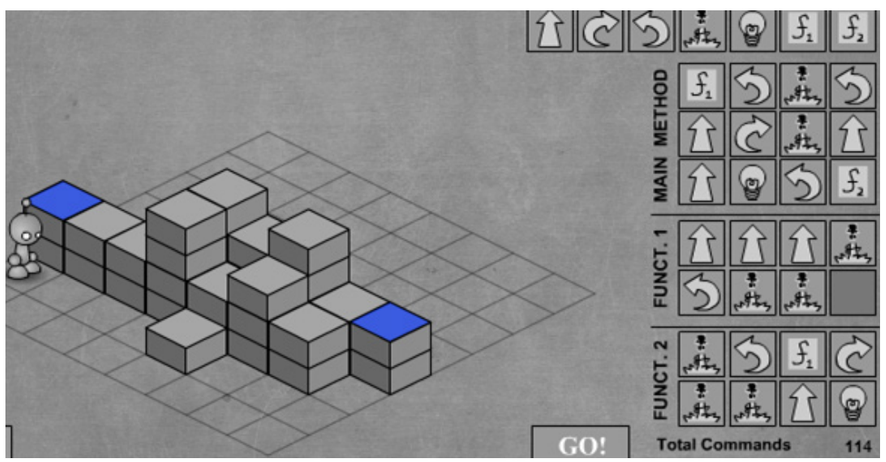
Luego de haberse asegurado de que los alumnos han comprendido cómo aprovechar los procedimientos para la solución de problemas, se les puede proponer que avancen a los siguientes niveles. Las soluciones y los patrones correspondientes a esos niveles son mostrados a continuación.
En el nivel 8 la secuencia de acciones más frecuente consiste en avanzar dos veces y encender la luz de la baldosa:



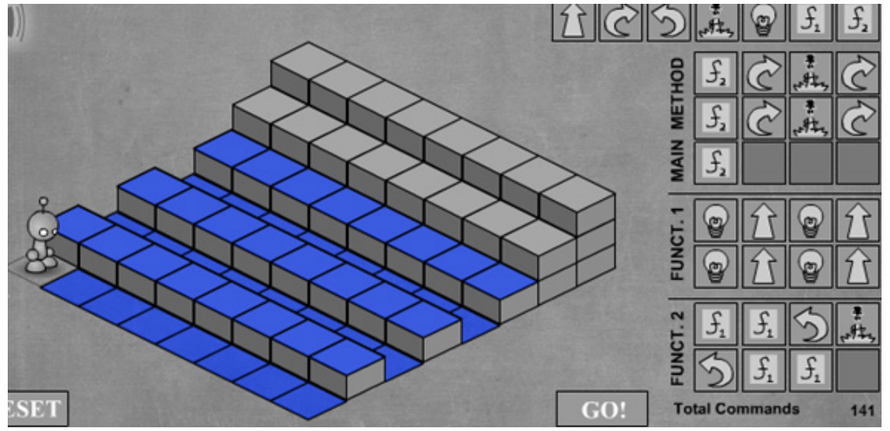
En el nivel 9, las baldosas azules forman un cuadrado, por lo que, una vez que el robot llega a la primera baldosa azul, solo hay que definir el procedimiento adecuado para uno de los lados y luego copiarlo para los lados restantes.



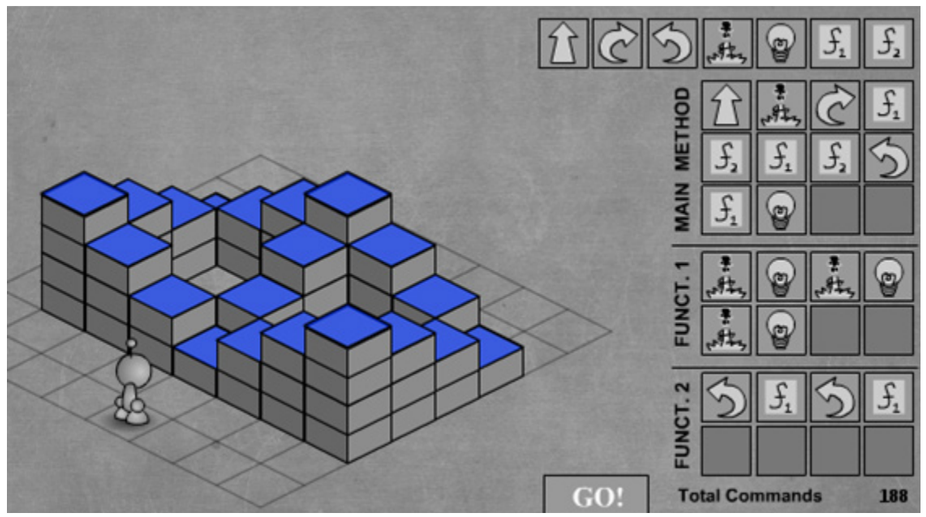
El nivel 10 es uno de los más difíciles. Lo más conveniente es seguir estos pasos.
• Definir un procedimiento que incluya las acciones necesarias para que el robot se dirija hasta el punto intermedio entre las dos baldosas azules.
• Desde allí, hacer que el robot avance primero hacia la baldosa azul ubicada a la izquierda, la encienda y gire.
• Definir un segundo procedimiento, que comience con la acción de saltar desde la baldosa encendida, de modo que el robot quede ubicado en la posición inicial, se continúe con otro giro y el primer procedimiento (que hace que el robot avance nuevamente al punto intermedio) y finalice con la indicación de las acciones requeridas para que el robot encienda la luz de la baldosa del extremo derecho.



En el nivel 11 conviene definir un procedimiento que encienda cuatro luces contiguas ya que todas las filas tienen ocho luces, excepto la primera. Lo más práctico es considerar esta primera fila como si fuera igual que las otras e indicar que el robot realice la acción de encender la luz sobre la baldosa, aunque esta no tenga ninguna. Es una acción redundante, pero válida en este juego.



El último nivel es fácil de resolver. Un inconveniente que puede surgir aquí (si no surgió anteriormente) es que el robot apague una de las luces que encendió con anterioridad. Para solucionarlo, simplemente se indica que ejecute la acción *encender luz* al final de la primera grilla, de manera que el robot vuelva a encenderla.



Cierre

A modo de cierre del trabajo realizado, se sugiere retomar la reflexión, iniciada en la secuencia anterior, acerca del concepto de programa en el ámbito de la informática. Al respecto, los alumnos podrán comparar las actividades realizadas en Scratch con las llevadas a cabo en Lightbot y, a partir de ello, indicar los aspectos nuevos en relación con el tipo de operaciones que hace un programa. Así, mientras en casos anteriores no había un problema bien delimitado para resolver (en la actividad de Scratch en la que había que definir la actividad de dormirse, el foco estaba puesto más bien en cómo *plantear* un problema), ahora se sabe exactamente si el programa resuelve o no el problema planteado. Entre los aspectos novedosos, se encuentra el método de dividir un problema grande en partes más pequeñas, lo cual se conoce en programación como **división en subtareas**. La división en subtareas consiste en:

- Identificar pequeñas tareas fácilmente explicables (avanzar un paso, subir un peldaño, encender una luz) y asignarles un buen nombre para que se entienda qué representan.
- Combinar estas pequeñas tareas para definir el programa que resuelve el problema (los procedimientos, que en Lightbot se llaman *funciones*).

Secuencia didáctica 4

Repetición simple

Buena parte de las secuencias didácticas anteriores estaban orientadas a ofrecer a los alumnos algunas herramientas básicas para manejar Scratch y Lightbot. Con esta secuencia, se empieza a trabajar en procesos más específicos, que los alumnos irán explorando sobre la base de lo ya aprendido. En este caso, la propuesta consiste en indagar, a través de una actividad de Scratch, en los procesos informáticos implicados en la repetición de tareas.

Objetivos

- Aplicar herramientas de programación para la ejecución de tareas repetitivas.
- Reconocer que las computadoras pueden repetir una tarea cuantas veces sea necesario.

Desarrollo

Los alumnos ingresarán a la actividad de Scratch *No me canso de saltar*, disponible **on line**, donde se presenta el siguiente escenario.

http://scratch.mit.edu/projects/42294970/#editor

El objetivo es hacer que el gato salte 30 veces para llegar a la última piedra. Para eso, en la categoría *Más bloques,* solo se da la siguiente primitiva: Saltar

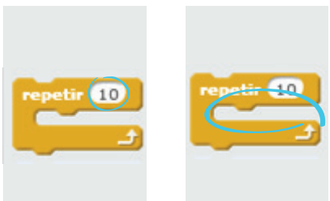
Cuando se ejecuta una vez *saltar,* el gato pasa de una piedra a otra y avisa cuántos saltos le faltan para cumplir el objetivo:

Actividad

A partir de estas breves indicaciones, se sugiere dejar que los alumnos intenten resolver la actividad con lo aprendido hasta el momento. Probablemente, la solución que propongan consista en repetir 30 veces el comando *saltar*:

Además de poco práctica, esta solución presenta el inconveniente de que puede hacer que sea difícil visualizar la manera de modificar la secuencia. Se vuelve a presentar un problema similar al planteado en la secuencia didáctica anterior, cuando, con el fin de ahorrar las casillas disponibles, se recurrió a la definición de diversos procedimientos o funciones, cada uno formado por dos o más subtareas. Puede suceder que algunos alumnos, teniendo esto presente, pregunten si existe algún tipo de comando que permita *economizar* la cantidad de bloques, de manera que la realización de los saltos pueda indicarse de forma menos engorrosa.
Una vez delimitado el problema, se presenta el bloque *repetir*, ubicado en la categoría *Control*:

Tal como se hace con cualquier otro bloque que se desea utilizar, se lo arrastra hasta el editor. Luego, en la casilla de arriba se indica la cantidad de veces que se desea repetir una acción. En el hueco de abajo, se ubica la secuencia de comandos (en este caso, la acción que ejecutará repetidamente el gato):



Actividad

Tras esta explicación, se puede nuevamente dejar que los alumnos intenten resolver el resto de la actividad. La solución es la siguiente:

Cierre

Al finalizar la secuencia, conviene hacer hincapié en los siguientes aspectos.

- Las computadoras pueden repetir una tarea cuantas veces sea necesario.
- Mediante el uso de determinados bloques de control, como el bloque repetir, es posible ahorrar espacio y simplificar lo que se quiere expresar. Además, facilita corregir o modificar una secuencia de comandos.
- A partir de ahora, siempre que un comando o una secuencia de comandos se repita una cantidad fija de veces, se utilizará el bloque **repetir**.
- Los comandos o instrucciones que se quieran repetir se colocarán dentro de ese bloque; en el casillero en blanco del bloque se indicará el número de veces que se deben **repetir** los comandos.

Secuencia didáctica 5

Programar en papel cuadriculado

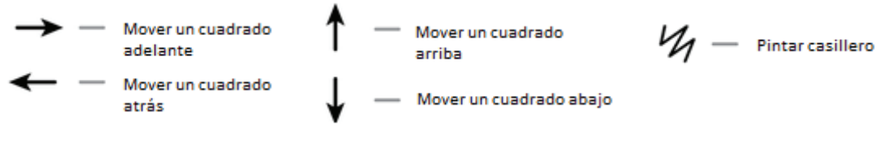
La idea que articula esta secuencia es la indagación en los mecanismos básicos implicados en la programación. Para ello, se proponen algunas actividades en las que los alumnos *programarán* en papel.

Objetivos

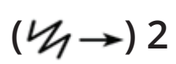
- Ejercitarse en algunos de los principios que se aplican en programación, a través de un modelo.
- Formular instrucciones de modo adecuado e interpretarlas correctamente, de acuerdo con el modelo propuesto.

Desarrollo

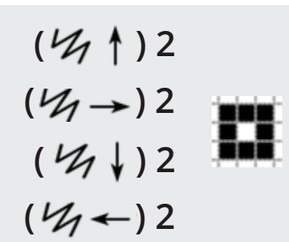
Para las actividades que siguen se emplearán dos hojas cuadriculadas (también pueden ser cuadrículas dibujadas sobre hojas en blanco), un lápiz y una goma de borrar. Una hoja se usará para programar y otra para dibujar, siguiendo las instrucciones de un *programa*. En la parte superior de cada hoja se indicará el uso que le corresponde: *"Programas"* y *"Dibujos"*. Para programar, se emplearán los siguientes comandos (es decir, instrucciones):



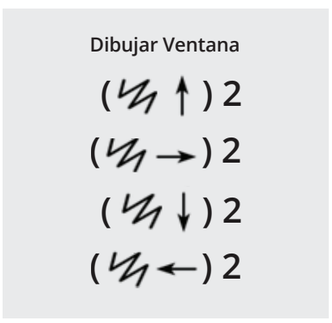
Además, junto a cada comando o secuencia de comandos se puede indicar cuántas veces debe ejecutarse. Para ello, se colocará el comando o la secuencia entre paréntesis, seguido de la cifra correspondiente al número de veces que debe ejecutarse. Por ejemplo, la siguiente secuencia significa "*pintar hacia la derecha 2 cuadrados*":



Esto incluye el cuadrado sobre el que se encuentra el lápiz (el cual equivale al cursor de la computadora). La programación se realizará de forma similar a como se hace en Lightbot, pero en vez ubicar los comandos horizontalmente, de izquierda a derecha, se lo hará de arriba hacia abajo, poniendo un comando por renglón. Por ejemplo, la siguiente secuencia corresponde a las instrucciones necesarias para dibujar un cuadrado:

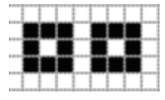


Esta secuencia de cuatro pasos puede conformar un procedimiento (es decir –como se recordará– un comando nuevo integrado por un grupo de acciones o subtareas que forman parte de una actividad más amplia). Cada procedimiento, para cuya definición se podrá utilizar la parte inferior de la hoja Programa, estará encabezado por un nombre breve, lo suficientemente descriptivo de la tarea a la que remite, seguido de su definición. Por ejemplo, a la secuencia anterior se le puede asignar el nombre *Dibujar ventana:*



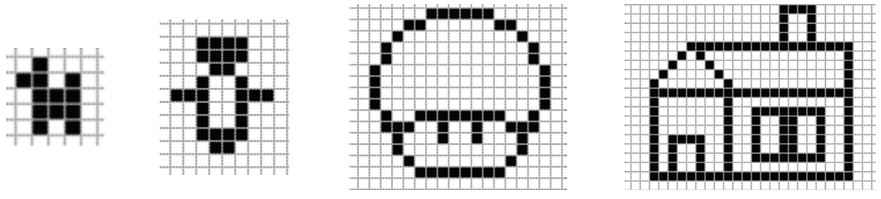
Para utilizar un procedimiento, se escribe su nombre entre corchetes, en el lugar del programa que corresponda: **[Dibujar ventana]**
A su vez, los procedimientos definidos pueden utilizarse para definir otros procedimientos nuevos:
Dibujar ventana doble
[Dibujar ventana]
→ 4
[Dibujar ventana]

Lo que da como resultado el siguiente dibujo:



Actividad

Luego de explicar las reglas anteriores, se les puede proponer a los alumnos que realicen los programas para cada uno de los dibujos que siguen.

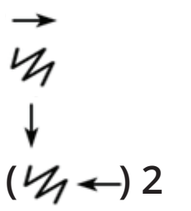


El programa correspondiente al primer dibujo podría ser el siguiente:

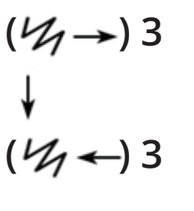
[Dibujar cabeza]
[Ir a cuerpo]
[Dibujar cuerpo]
[Ir a pierna delantera]
[Dibujar pierna]
[Ir a pierna trasera]
[Dibujar pierna]
[Ir a cola]
[Dibujar cola]

Este programa supone las siguientes definiciones de los procedimientos:

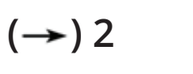
Dibujar cabeza



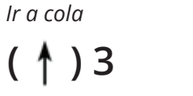
Dibujar cuerpo



Ir a pierna trasera



Ir a cola



Ir a cuerpo

Ir a pierna delantera

Dibujar cola

Dibujar pierna

Conviene hacer hincapié en que, al crear un procedimiento para cada parte del dibujo, no es necesario cambiar el programa principal para que el dibujo del perro sea más grande: basta con hacer los ajustes pertinentes en las definiciones de los procedimientos. Igualmente, mediante esta estrategia (que no es otra cosa que una subdivisión de tareas), es posible encontrar fácilmente la parte del programa que tiene errores en el caso de que el dibujo resultante no sea el esperado. Por otra parte, es importante destacar que si los procedimientos se hubieran definido privilegiando un criterio meramente formal –por ejemplo, indicando los movimientos y los trazos que el lápiz debería hacer columna por columna o fila por fila– en vez de hacerlo por figuras (cabeza, cola, pata), sería necesario cambiar todo el programa para modificar el tamaño del dibujo y, además, resultaría mucho más trabajoso detectar las fallas que pudiera haber. Las mismas abstracciones que usamos habitualmente en la descripción de objetos son útiles para programar.
La estrategia empleada para *programar* el primer dibujo puede aplicarse a los otros dibujos; así, por ejemplo, el programa principal correspondiente al último dibujo quedaría de esta manera:

[Ir a techo]
[Dibujar techo]
[Ir a chimenea]
[Dibujar chimenea]
[Ir a pared frontal]
[Dibujar pared frontal]
[Ir a puerta]
[Dibujar puerta]
[Ir a pared lateral]
[Dibujar pared lateral]
[Ir a ventana izquierda]
[Dibujar ventana]
[Ir a ventana derecha]
[Dibujar ventana]

Actividad

Como variante o complemento de las actividades anteriores, se podrá indicar a los alumnos que cada uno piense un dibujo, lo programe y, luego, le pase a un compañero el programa para que, siguiendo las indicaciones, reproduzca el dibujo.

Cierre

Como cierre, se sugiere socializar las producciones de los alumnos y evaluar las dificultades que hayan surgido al programar en papel. Al respecto, se les podrá pedir a los alumnos que reflexionen, por ejemplo, acerca del uso de los procedimientos: cómo los definieron, cuáles fueron los nombres que les asignaron y si les resultó más conveniente hacer varias subtareas o programar todo el dibujo en una sola tarea.

Secuencia didáctica 6

Repetición Simple (II)

Esta secuencia está centrada en el uso de programas para la resolución de problemas. Se trata de un tema ya visto anteriormente, al abordar el empleo de los procedimientos (o funciones) en Lightbot. Ahora se explorará su uso en Scratch.

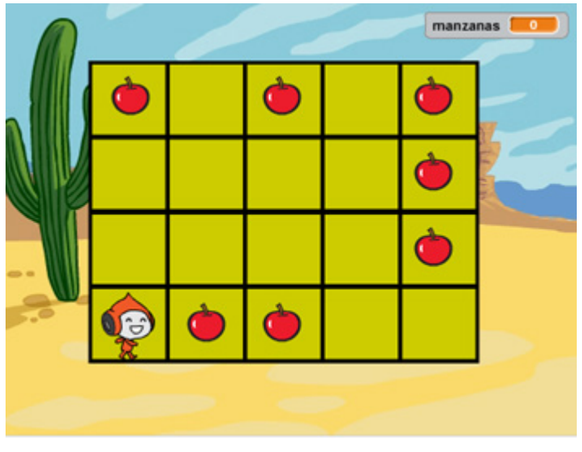
Objetivos

- Ejercitarse en el uso de programas para la resolución de problemas.
- Establecer comparaciones entre las diversas maneras posibles de resolver un problema.
- Reforzar los conceptos de repetición y separación en procedimientos.

Desarrollo

http://scratch.mit.edu/projects/42293070/#editor

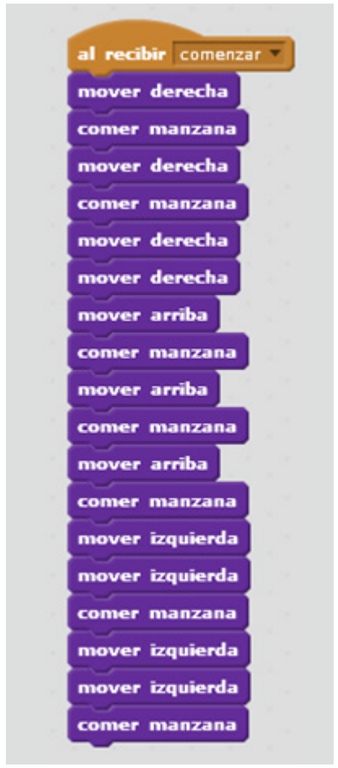
Se trabajará con el proyecto *El marciano en el desierto*, disponible **on line**.



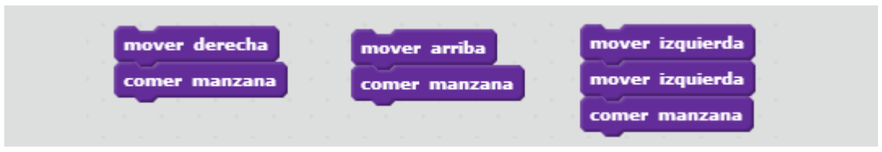
El objetivo es que el marciano coma todas las manzanas del tablero (en el caso de que al ingresar no se vean las manzanas, basta con hacer clic en el ícono ⚑). Para ello, se dispone de las siguientes instrucciones, que se encuentran en la opción *Más bloques*:

Actividad

Al igual que en otras ocasiones, se dejará que los alumnos intenten resolver el problema por sí mismos. Una solución obvia es la siguiente:

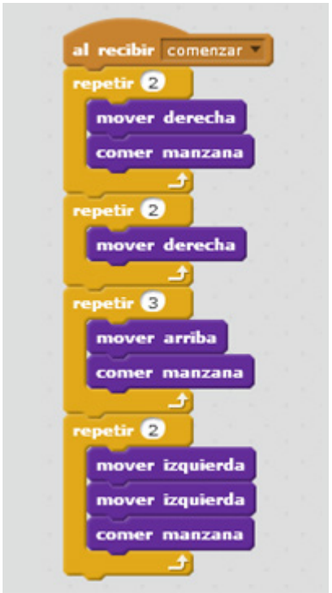


Basta un rápido vistazo para ver que, en esta solución (y en otras parecidas, que solo varían entre sí en el orden en que se ubican las instrucciones primitivas), se repiten algunas secuencias:



Actividad

Una vez que se han identificado estas secuencias, se puede dejar que los alumnos vuelvan sobre el problema y traten de resolverlo mediante el uso del bloque *repetir* (presentado durante la secuencia didáctica 4). Cabe contemplar la posibilidad de que algunos alumnos, en vez de acudir a la solución primitiva mostrada más arriba, prevean el uso de este bloque ya desde una primera instancia y que intenten utilizarlo (con menor o mayor éxito). En ese caso, se sugiere no dar mayores precisiones que las dadas al comienzo y aguardar a que la mayoría de los alumnos haya realizado al menos una primera tentativa para resolver el problema antes de avanzar con la explicación.
El uso del bloque *repetir* permitirá llegar al siguiente resultado:



No obstante ser esta una solución más satisfactoria, puede mejorarse mediante una herramienta ya empleada, el procedimiento, que consiste en crear nuevos bloques y explicarle a la computadora (representada, en este caso, por el marciano) cómo realizar cosas que hasta ahora no sabe hacer. Ya se ha empleado esta herramienta en Scratch al trabajar con la actividad *El gato en la calle*, para definir la actividad de dormirse. Como se recordará, para definir un procedimiento, en primer lugar hay que hacer clic, en la sección *Más bloques*, en *Crear un bloque*. Luego, en la carpeta que se encuentra dentro de la nueva ventana, se escribe el nombre que se desea asignarle al procedimiento; en este caso, el nombre puede ser *Comer 2 derecha:*

Al presionar *OK*, aparecerá en el editor un nuevo bloque, el correspondiente a la definición del procedimiento, que debe completarse con las acciones primitivas que se desea que ejecute el marciano. En este punto, se les puede indicar a los alumnos que definan este procedimiento. La definición se verá así:

Una vez definido el procedimiento, para que el marciano lo ejecute habrá que arrastrar desde la lista de bloques de la izquierda el bloque *Comer 2 derecha* hasta el editor y ubicarlo debajo del bloque de evento *al recibir comenzar* .

Se sugiere que, a partir de aquí, los alumnos definan los restantes procedimientos y, luego, los utilicen para reformular la secuencia de comandos del programa. Los procedimientos deberán quedar definidos de esta manera:

El programa finalmente debería quedar así:

Cierre

Para el cierre, se sugiere volver a conversar sobre las ventajas del uso de procedimientos y de la repetición de acciones. Si bien para la computadora el resultado de ejecutar las secuencias de comandos propuestas al inicio y al final es el mismo (el marciano come las manzanas de la misma forma que antes), para los seres humanos la segunda es mucho más fácil de entender (aunque en principio pueda parecer lo contrario): esta –puede agregarse– sería la forma con la que los alumnos le explicarían a sus amigos cómo comer todas las manzanas del tablero de manera muy simple. Además, resulta muy útil cuando el problema se vuelve complejo.

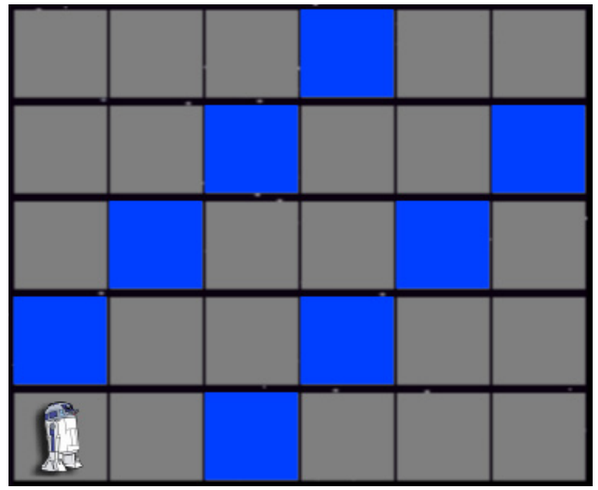
Ejercitaciones

Las actividades de Scratch que siguen presentan diversos escenarios y plantean problemas de complejidad creciente. Se proponen a modo de ejercitación para que los alumnos afiancen lo aprendido hasta el momento. Luego del trabajo con cada actividad, se sugiere hacer una puesta en común a fin de que los alumnos expongan las dificultades que se les hayan podido presentar. Asimismo, también será útil comparar cada una con *El gato en la calle*, ya trabajada.

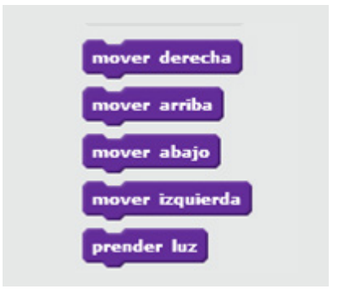
Lightbot en Scratch

http://scratch.mit.edu/projects/42292564/#editor

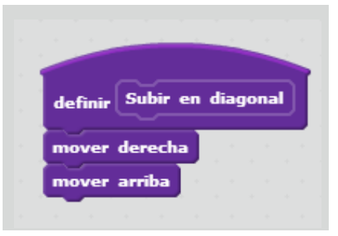
Esta actividad de Scratch es una adaptación de Lightbot. Al ingresar se observa el siguiente escenario:



Como en Lightbot, el objetivo es que el robot encienda las baldosas azules. La diferencia es que, en vez de ubicar los comandos en una grilla, se utilizan los bloques característicos de Scratch. Las acciones primitivas (que, como es habitual, se encuentran en la categoría *Más bloques*) en este caso son:



Se espera que los alumnos se den cuenta de que les conviene crear un procedimiento que mueva al robot en diagonal, ya que ese movimiento es la acción que se repite más a menudo. Este procedimiento podría definirse así:



Luego, pueden crear dos procedimientos más: uno para que el robot encienda las cuatro baldosas azules que forman una diagonal a la izquierda y otro para que encienda las cuatro baldosas azules restantes, que forman otra diagonal a la derecha. Las definiciones de estos procedimientos serían las siguientes:

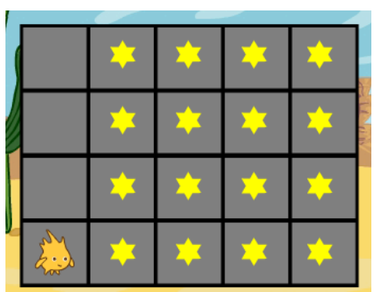
A partir de los procedimientos definidos, la solución quedaría planteada de esta manera:

Como en otras oportunidades, se sugiere dejar a los alumnos que intenten resolver el problema por sí mismos e intervenir solo cuando sea necesario; por ejemplo, si se observa que buscan una solución sin crear procedimientos. En tal caso, se les puede indicar que el desafío es lograr simplificar las tareas, tal como se hizo en Lightbot, de modo que adquieran una destreza que les permitirá resolver problemas más difíciles. Luego, se podrá orientar a cada alumno por separado acerca de qué procedimientos conviene crear.

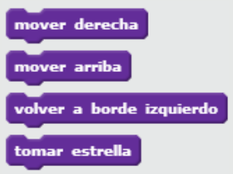
El recolector de estrellas

http://scratch.mit.edu/projects/42294488/#editor

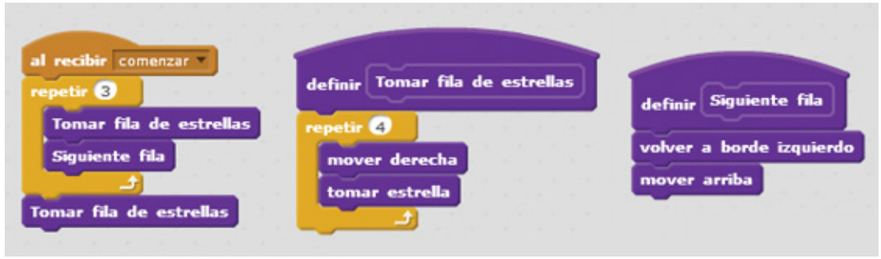
El planteo en esta actividad de Scratch es similar al de *El marciano en el desierto*. Aquí se presenta a otro extraterrestre, que recolecta estrellas. El objetivo es que el extraterrestre tome todas las estrellas del tablero



Las instrucciones primitivas son las siguientes:



Como se puede apreciar, no hay instrucciones para que el extraterrestre se mueva hacia abajo o hacia la izquierda. El bloque *volver al borde izquierdo* simplemente hace que el extraterrestre regrese desde la fila en la que se encuentra a la primera celda de la izquierda. La estrategia –que deberán descubrir los alumnos– será hacer que el extraterrestre tome las estrellas fila por fila.
Una solución posible (entre varias otras) es la siguiente:

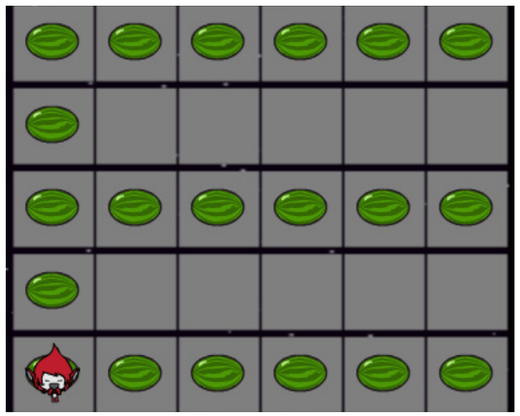


Otras posibilidades, a partir de los mismos procedimientos, son las siguientes:

María, la come sandías

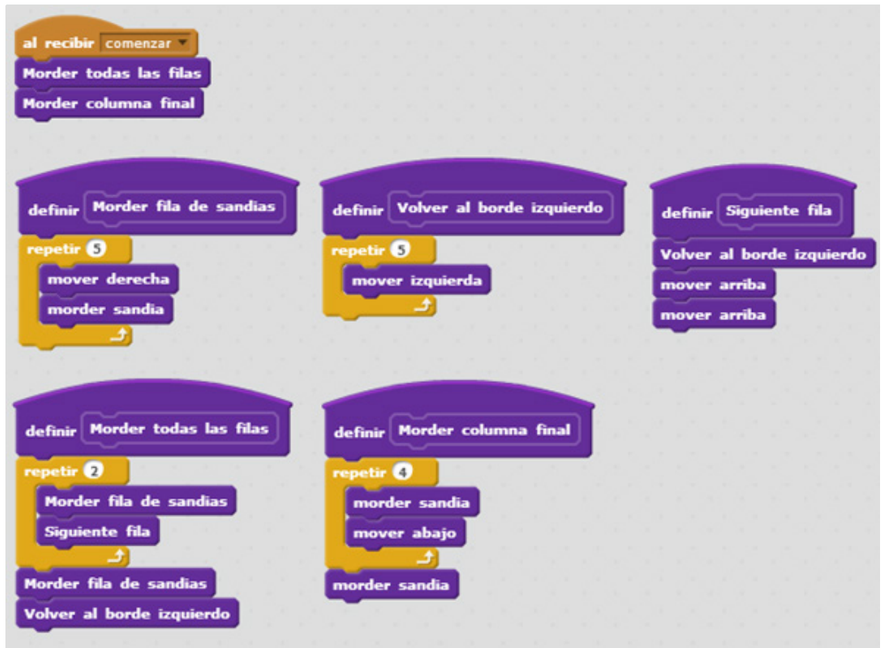
http://scratch.mit.edu/projects/42294654/#editor

En este juego, el personaje, María, debe morder todas las sandías que se encuentran en el tablero.



Las instrucciones primitivas son las siguientes:

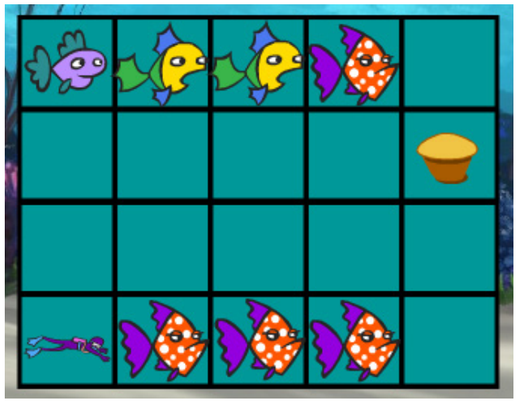
Se sugiere conversar con los alumnos acerca de cómo resolverían el juego antes de que intenten efectivamente hacerlo y orientarlos al respecto. Las soluciones posibles son varias. Una consiste en hacer que el personaje muerda las sandías de las tres filas, a partir de la segunda columna, y que luego muerda las de la primera columna. También puede morder primero las sandías de la primera columna y luego las de las tres filas; es indistinto. En cualquier caso, se requieren al menos dos procedimientos, uno por cada tarea. Además de estos procedimientos, se puede agregar otro que permita que el personaje vuelva al principio de la fila (es decir, el borde izquierdo del tablero). Esto en la actividad anterior era una primitiva, pero ahora habría que definirla.
Una vez planteados los términos generales de la solución, se puede dejar que los alumnos intenten formularla en sus detalles. Si les resulta complicado, se los guiará en los pasos a seguir. Al respecto, se les podrá indicar que lo más conveniente será definir los procedimientos en el siguiente orden: en primer lugar, el procedimiento necesario para que María muerda las sandías de una fila (excepto las que forman la primera columna); luego, el que corresponde a la acción de volver al inicio de la fila; a continuación, un procedimiento que haga que María muerda las sandías de las tres filas (donde se integren los procedimientos anteriores), y, finalmente, uno para que muerda las sandías de la columna de la izquierda. A partir de estos procedimientos, una solución posible es la siguiente:



Alimentando a los peces

http://scratch.mit.edu/projects/42292362/#editor

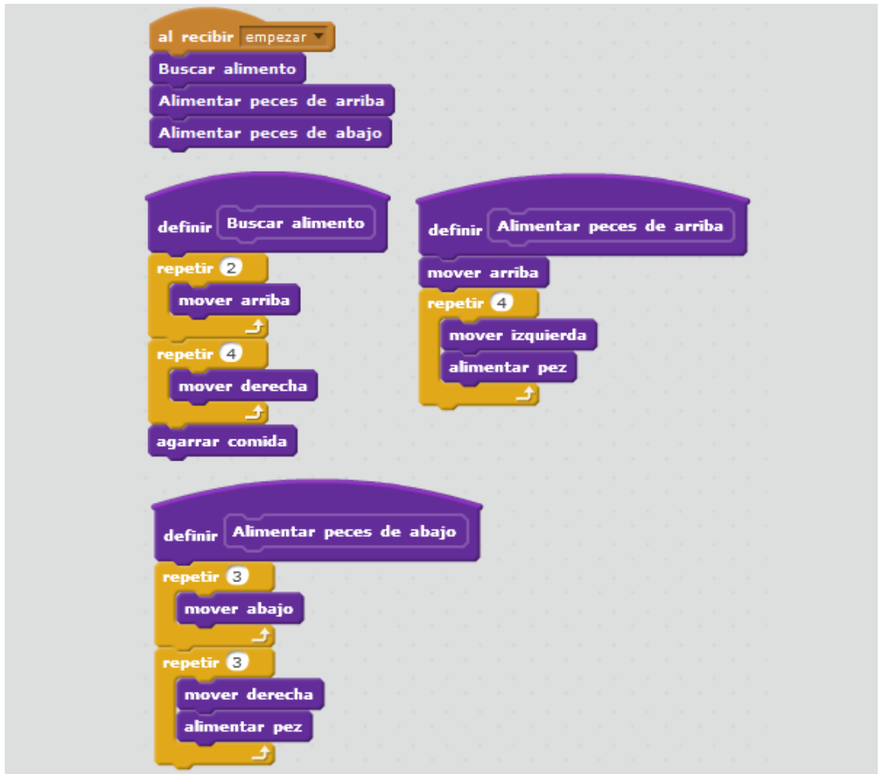
En este caso, el buzo debe buscar la comida y luego acercarse a cada pez para alimentarlo.



Las instrucciones primitivas son las siguientes:

- alimentar pez
- mover arriba
- mover abajo
- mover derecha
- agarrar comida
- mover izquierda

Como en el caso anterior, los alumnos podrán discutir acerca de los procedimientos que sean más adecuados para resolver el problema; por ejemplo, podría crearse un procedimiento para ir a buscar el alimento y otro para alimentar a los peces.
Una solución posible es la siguiente:

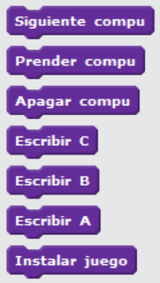


Instalando juegos

http://scratch.mit.edu/projects/42296824/#editor

En esta actividad de Scratch, el personaje tiene que instalar un videojuego en las tres computadoras de la biblioteca. Para ello, debe encender cada computadora, ingresar la contraseña (que, en este caso, es *ABC*), cargar el juego y finalmente apagar la máquina. Al ser un proceso repetitivo, lo más conveniente es automatizarlo.

Las instrucciones primitivas son las siguientes:



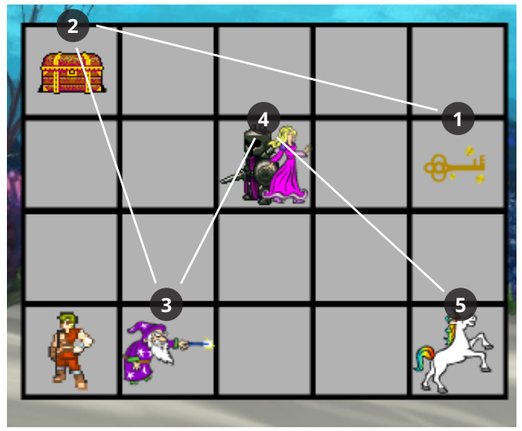
El desafío para las alumnos es definir un procedimiento que permita realizar el proceso descripto y, luego, definir un programa que lo ejecute por cada computadora de la biblioteca. La solución es la siguiente:

La gran aventura del mar encantado

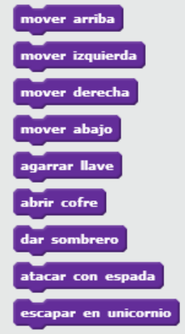
Mediante esta actividad de Scratch, se pretende que los alumnos exploten al máximo la técnica de división en subtareas. El objetivo principal es que el héroe (ubicado en el casillero del extremo inferior izquierdo) logre escapar en el unicornio con la princesa. Para ello, debe superar una serie de pruebas en el siguiente orden.

1. Buscar la llave.
2. Con la llave, abrir el cofre y tomar un sombrero mágico que se encuentra adentro.
3. Entregarle el sombrero al mago para que este le dé una espada.
4. Con la espada, ir a luchar contra el caballero oscuro y rescatar a la princesa.
5. Ir con la princesa hasta el unicornio y escapar.

http://scratch.mit.edu/projects/42294776/#editor



Las instrucciones primitivas son las siguientes:

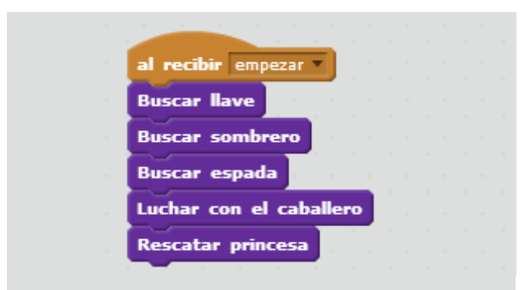


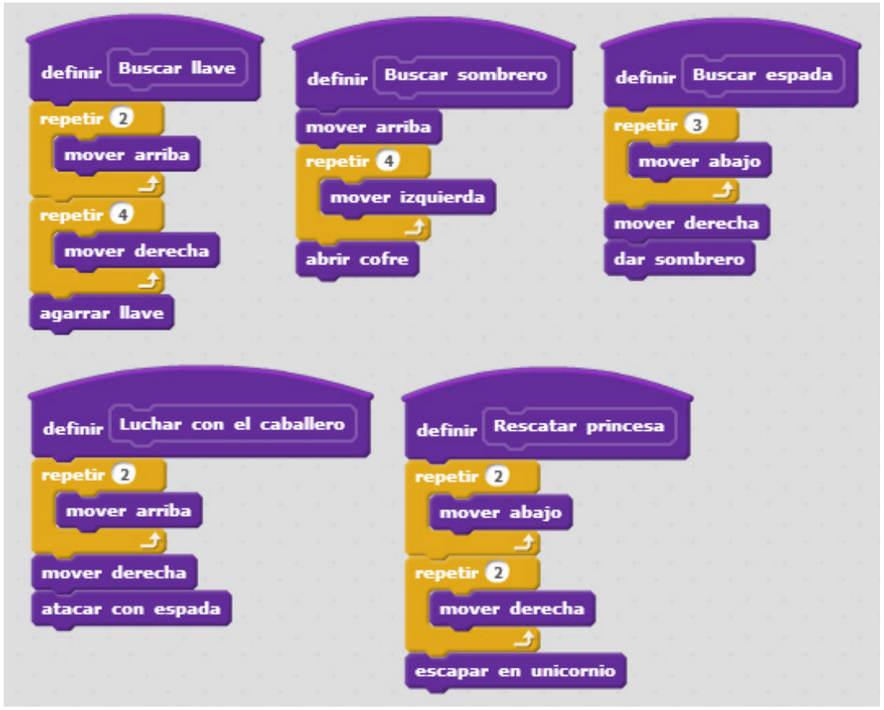
Para que el héroe pueda superar las pruebas, hay que tener en cuenta lo siguiente:

- **No puede agarrar la llave si no está en la casilla donde está la llave;**
- **No puede abrir el cofre sin la llave y si no está junto al cofre;**
- **No puede darle el sombrero al mago si no tiene el sombrero y no está junto al mago;**
- **No puede atacar al caballero oscuro si no tiene la espada y no está en la casilla donde se encuentra el caballero;**
- **No puede escapar con la princesa montado en el unicornio si no está junto con ella en la casilla donde se encuentra el unicornio.**

Como contrapartida a estas restricciones, el héroe puede pasar por encima de cualquier figura sin problemas. Esto significa que no tiene que esquivar ningún obstáculo.

Una vez más, será conveniente dejar que los alumnos resuelvan la actividad como se les ocurra, pero recordándoles que, si no identifican y definen distintos procedimientos, la tarea puede volverse bastante ardua y compleja. Tal vez convenga escribir en el pizarrón los pasos a seguir para que todos puedan visualizarlos cómodamente y no se pierdan. Una solución posible es la siguiente:





Reparando la nave

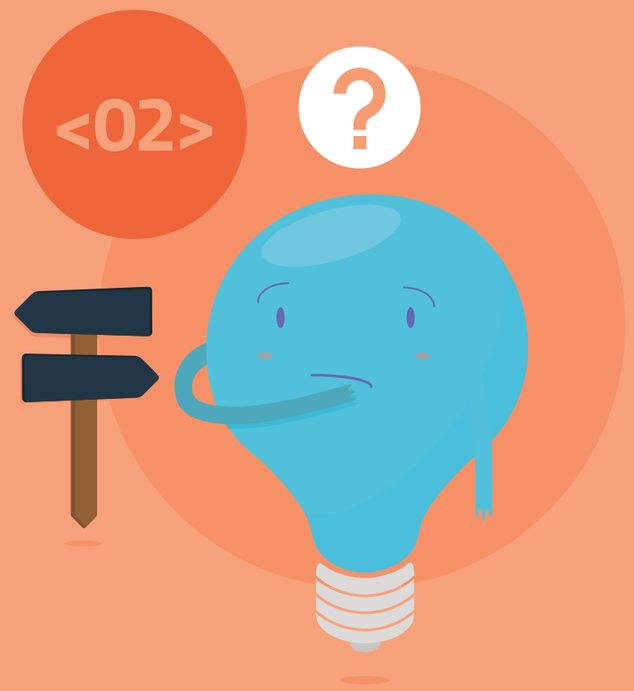
http://scratch.mit.edu/projects/42296150/#editor

El último juego de esta parte, presenta a un marciano que debe aprovisionarse de carbón y hierro para reparar su nave y así poder ponerla en funcionamiento. El marciano tendrá que realizar tres viajes por cada material que necesita. Una vez que lo haga, la nave se desplazará.

Las instrucciones primitivas son las siguientes:

- mover arriba
- mover abajo
- mover izquierda
- mover derecha
- Tomar Hierro
- Tomar Carbon
- Depositar en la nave
- Escapar

Existen varias soluciones igualmente válidas para el problema planteado. Una de ellas es la siguiente:



ALTERNATIVAS CONDICIONALES

Hasta el momento se crearon programas que permitían resolver situaciones en escenarios conocidos. Por ejemplo, cuando en Lightbot se le indicaba al robot que se moviera, se sabía dónde estaban las baldosas o casillas azules, cuáles eran las dimensiones del tablero y la posición original del robot, etcétera. Pero ¿qué ocurriría si hubiera que elaborar un programa que pudiese funcionar para distintas posiciones originales del robot o distintas ubicaciones de las casillas azules? En otras palabras, ¿cómo proveer una solución adecuada para distintos contextos? La respuesta a esta pregunta es el eje de esta parte, donde se aprenderá a trabajar con alternativas condicionales.

Secuencia didáctica 7

Escenarios cambiantes

En esta secuencia se reflexionará sobre el uso de alternativas condicionales en programación, a partir de actividades en las que no se utilizarán computadoras.

Objetivos

- Introducirse en la noción de alternativa condicional.
- Reconocer semejanzas y diferencias entre escenarios fijos y escenarios cambiantes.
- Identificar alternativas en un escenario.

Desarrollo

Para introducir la noción de **alternativa condicional** y el uso de alternativas condicionales en programación conviene –tal como se hizo en el apartado anterior– comenzar por actividades que funcionen como una *puesta en escena* del problema. Al respecto, puede retomarse el escenario planteado en la secuencia didáctica 1, en el que el docente, asumiendo el rol de autómata, recibía las instrucciones que le permitían, por ejemplo, salir del aula.

Actividad

Ahora, en un escenario más complejo, se podrá considerar que el aula tiene dos puertas (o una puerta y una ventana), y que solamente una de ellas está abierta cada día, pero no se sabe cuál. Por lo tanto, el problema consiste en cómo hacer para que el programa que permite al autómata salir del aula funcione cualquier día. Para ello, se les explicará a los alumnos que contarán con una nueva posibilidad para programar al autómata: el control o la instrucción (que corresponde a un bloque en Scratch) **si *<condición>* entonces *<secuencia de acciones>***. Esta indicación permite ejecutar una secuencia de acciones si se cumple determinada condición. También se dispondrá de una variante de esa instrucción, denominada **si *<condición>* entonces *<secuencia de acciones 1>*, si no *<secuencia de acciones 2>***. Al utilizarla, se ejecutará la secuencia de comandos especificada a continuación de **si no** en el caso de que la condición sea falsa (si es verdadera, se ejecutan las instrucciones ubicadas a continuación de si). Mediante estos controles, se puede indicar que el autómata salga del aula, por ejemplo, de la siguiente manera:

si *<puerta izquierda abierta>* entonces *<avanzar diez pasos hacia izquierda>*
si no *<avanzar diez pasos hacia derecha>*

Como se deduce del ejemplo –conviene aclarar–, el autómata no necesita acercarse a la puerta para comprobar si está abierta o no.

Una vez planteado el escenario y presentadas las instrucciones de alternativas condicionales, los alumnos podrán construir grupalmente una solución que guíe al docente a la salida del aula sin importar qué puerta se encuentre abierta. La solución esperada es que el docente mire si la primera puerta está abierta y que, en función de ello, efectúe el recorrido que lo lleve a esa puerta o a la otra. En ambos casos, el programa deberá lograr que el autómata abra finalmente la puerta y salga del aula.

Cierre

Como cierre de esta secuencia, será oportuno conceptualizar lo trabajado en clase explicando que, en ciertas ocasiones, deseamos que algunas instrucciones no se ejecuten siempre, sino sólo cuando se cumple cierta condición. A esto lo llamamos **alternativa condicional**. A fin de afianzar la comprensión de este concepto, se sugiere indicarles a los alumnos que propongan diversos ejemplos de situaciones cotidianas que den lugar al uso de alternativas condicionales (si llueve, se suspende el partido; si se enciende la luz verde, el auto avanza y si se enciende la luz roja, se detiene; si la barrera se baja, el auto frena, etcétera). En cada caso, se deberán precisar las alternativas en juego.

Secuencia didáctica 8

Alternativas condicionales en Scratch

En esta secuencia se trabajará con una serie de actividades de Scratch que presentan un escenario cambiante y cuya resolución implica el uso de alternativas condicionales.

Objetivos

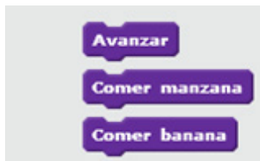
- Identificar los cambios que se producen en un escenario dado.
- Utilizar correctamente los comandos necesarios para ejecutar alternativas condicionales.
- Definir los programas adecuados a escenarios cambiantes.

Desarrollo

http://scratch.mit.edu/projects/42294260/#editor

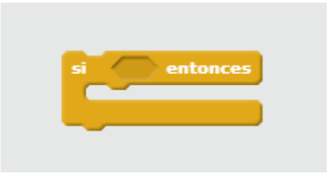
Se comenzará con el proyecto de Scratch *La elección del mono*. Al ingresar y hacer clic varias veces en , se podrán observar aleatoriamente los siguientes cambios de escenario:

El objetivo es simple: lograr que el mono coma una manzana cuando hay una manzana y que coma una banana cuando hay una banana. Para ello, se cuenta con las siguientes acciones primitivas:



Es importante destacar que en la casilla que se encuentra a la derecha del mono hay una u otra fruta, pero no ambas a la vez. Por este motivo, aunque pueda parecer algo evidente, tal vez sea oportuno señalar que no es posible solucionar el problema utilizando solo las primitivas. Por ejemplo, una secuencia como la siguiente, daría un resultado incorrecto, que hace que el programa se detenga ya que intentará comer una banana y una manzana (y solo una fruta puede estar presente):

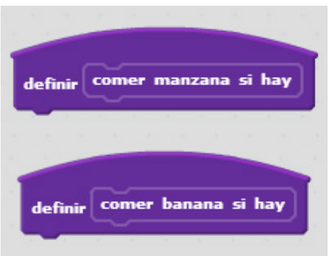
Para poder resolver la actividad, es necesario utilizar un bloque nuevo (ya presentado en la actividad anterior): **si... entonces**. Este bloque, que se encuentra en la categoría *Control*, permite, como se recordará, ejecutar un comando solo en el caso de cumplirse cierta condición.



Sin embargo, antes de intentar ver cómo utilizar este bloque, conviene dividir el problema, es decir, establecer la secuencia de acciones adecuada para el objetivo. Ello implica definir al menos un procedimiento, al que se puede llamar *comer fruta*, que resuelva el problema de comer una manzana o una banana.

Teniendo en cuenta este procedimiento, el programa que ejecutará el mono será tan simple como el siguiente:

Ahora bien, ¿cómo definir *comer fruta*? Puesto que de lo que se trata es de que el mono coma o la manzana o la banana, es necesario darle alguna indicación para que *sepa* de qué fruta se trata en cada caso. Así, el problema de comer fruta incluye, a su vez, otros dos:

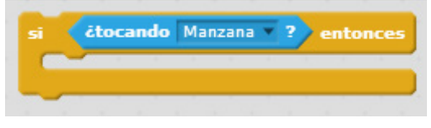


La definición de cada uno de estos dos procedimientos deberá contener las instrucciones necesarias para que el mono reconozca la fruta y la coma si corresponde. En relación con lo primero (y luego de asegurarse de que los alumnos hayan seguido el razonamiento hasta aquí esbozado), se presentará el bloque *¿tocando...?* ubicado en la categoría *Sensores*.

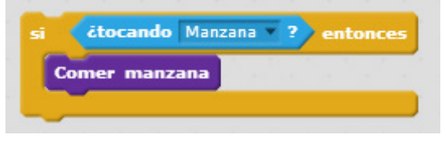
Mediante este bloque, es posible hacer que el mono verifique si está tocando cierta fruta. Para ello, se hace clic exactamente sobre el casillero con la punta de flecha que se encuentra junto a *tocando* y se selecciona de una lista la fruta que se desea; por ejemplo, la manzana:

Si se observa la forma de este bloque, se notará que es idéntica al casillero del bloque *si... entonces*:

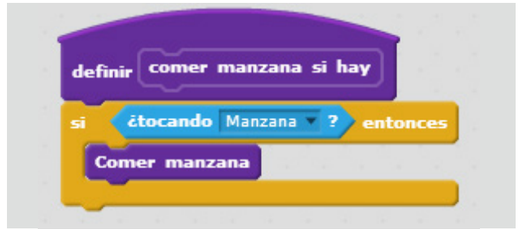
Esto no es casual, ya que corresponde a la condición con la cual se vincula una acción (o una secuencia de acciones):



Esta primera parte de la instrucción se lee: “si tocando manzana, entonces...”. Para completarla, es necesario elegir el comando que se ejecutará en el caso de que se cumpla esa condición.



De esta manera, se ha resuelto el problema de que el mono reconozca la manzana y la coma. El procedimiento *comer manzana si hay* queda definido de la siguiente manera:



Actividad

Luego de haber explicado el uso de los bloques *si entonces... y ¿tocando...?* se puede dejar que los alumnos terminen de resolver la actividad: primero, deberán definir el procedimiento *comer banana si hay*; después, terminar de definir el procedimiento *comer fruta* y, finalmente, verificar que el problema se resuelve correctamente, presionando varias veces la banderita verde.

Cierre

Como cierre de esta secuencia, se sugiere hacer hincapié en la idea de que toda alternativa condicional consta de dos partes: una condición y una acción. Si se cumple la condición, entonces será realizada la acción. Asimismo, se sugiere repasar con los alumnos las alternativas condicionales utilizadas en *La elección del mono* (los dos *si... entonces* de los dos procedimientos) y de qué condición y qué acción consta cada una. A continuación, se les puede indicar que propongan otros ejemplos de alternativas condicionales. Por ejemplo, el programa “irse del salón” podría constar de las alternativas “cerrar la ventana” (si está abierta) y “cerrar la puerta” (si está abierta).

Ejercitaciones

A través de las actividades de Scratch que siguen se espera que los alumnos se entrenen en el uso de alternativas condicionales y se familiaricen con el uso de algunos comandos hasta el momento no utilizados (como el **bloque si… entonces… si no…**, que contempla la posibilidad de ejecutar otra acción en caso de que no se cumpla determinada condición). Como en otras ocasiones, se sugiere realizar luego de cada actividad una puesta en común.

Laberinto corto

http://scratch.mit.edu/projects/42294366/#editor

En esta actividad de Scratch vuelve a presentarse un escenario cambiante. El objetivo es que el robot (un pequeño ratón) avance un paso en la dirección correcta. Las instrucciones primitivas son dos:

Es decir, hay dos movimientos posibles del ratón: hacia abajo o hacia la derecha.

Para orientar a los alumnos, se puede señalar que si bien no es posible saber de antemano hacia dónde deberá desplazarse el autómata, basta que el ratón *revise* la flecha que aparece debajo de cada baldosa, para moverse en la dirección que ella indica; en otras palabras, puede decirse que la observación de la dirección de la flecha le permite al ratón formularse, mediante el uso del bloque **¿tocando…?,** preguntas como las siguientes:

Es evidente que si el ratón está *tocando* la flecha que apunta hacia abajo, debe moverse en esa dirección; caso contrario, deberá hacerlo hacia la derecha.
A partir de esta información, se dejará que los alumnos intenten resolver el problema por sí mismos. Como pista, se les puede sugerir que definan un procedimiento, llamado *avanzar*, en el que se pregunte qué flecha está tocando el ratón, y se mueva a la dirección indicada por ella.

Una solución posible (y que probablemente propongan los alumnos) es la siguiente:

Existe también otra variante, que se podrá presentar luego de que los alumnos hayan arribado a la solución anterior. Esta variante consiste en reemplazar el bloque *si… entonces* por el bloque *si… entonces… si no…*, que se encuentra entre las opciones de la categoría *Control*:

Como se observa, este bloque solo puede usarse cuando se trata de decidir entre dos alternativas. Nótese que en estos casos siempre es posible tomar la condición opuesta e intercambiar las porciones de programa asociadas a las respuestas positiva y negativa de la condición. Por ejemplo, en vez de:
si ¿tocando abajo? entonces mover abajo; si no, mover derecha
podría indicarse
si ¿tocando derecha? entonces mover derecha; si no, mover abajo.

Tres naranjas **http://**scratch.mit.edu/projects/42293196/#editor

En esta ocasión, se presentan tres casillas azules, en cada una de las cuales puede aparecer la mitad de una naranja. El marciano, que se encuentra en una casilla gris, debe recorrer las casillas azules y comer las naranjas.

Las casillas azules siempre son tres. De manera similar a como ocurría en los proyectos de Scratch anteriores, no hay modo de prever en cuáles de ellas habrá una naranja: pueden presentarse en las tres casillas, en dos, en una e, incluso, puede darse el caso de que las tres casillas estén vacías. Para comprobar esto, basta con hacer clic repetidas veces sobre el ícono 🚩 y observar cómo cambia el escenario. Cualquiera sea el escenario, el personaje solo deberá comer las naranjas que haya; si intenta hacerlo donde no las hay, el juego se reiniciará.

Las instrucciones son simples:

Al igual que en otras oportunidades, se sugiere conversar en clase acerca del modo en que podría resolverse el problema. Al respecto, conviene hacer hincapié en las diferencias que presenta este caso en relación con los casos anteriores: por un lado, el hecho de que no haya dos acciones posibles a realizar, lo cual hace que no resulte útil emplear el bloque *si... entonces... si no...*; por otro lado, la circunstancia de que el autómata debe avanzar tres veces, y no una vez, lo que implica que hay una acción que se repite (el bloque *repetir*, empleado en otras oportunidades, puede ser útil ahora). Cada vez que avance, el marciano deberá primero saber si en la casilla azul donde está parado hay una naranja, para luego comerla solo en el caso de que efectivamente la haya; de lo contrario, no deberá hacer nada.
Une vez establecidos los alcances del problema y algunas pautas generales para su resolución, los alumnos podrán trabajar por su cuenta. La solución más adecuada es la siguiente:

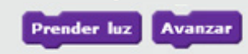
Otra solución, que pueden proponer algunos alumnos, es esta:

A pesar de que parece más sencilla y directa, esta solución, al no delimitar claramente las acciones que el autómata debe ejecutar (avanzar a una casilla azul, por un lado, y comer una naranja si la hay, por otro), no resulta la más conveniente para que los alumnos puedan razonar por partes el problema, lo cual es necesario cuando se abordan situaciones de mayor complejidad. En este sentido, se sugiere recordar a los alumnos que resulta más ventajoso definir procedimientos acotados que resuelvan un problema pequeño para luego poder usarlos en la resolución del problema más grande.

Lightbot recargado **http://**scratch.mit.edu/projects/42293536/#editor

En esta actividad de Scratch, el robot –tal como sucedía en Lightbot– debe encender las celdas azules. Si bien la longitud total de la fila es siempre la misma (once casillas o celdas), el número y la ubicación de las celdas azules pueden cambiar, lo que se verifica al presionar repetidamente la banderita verde. Solamente la última celda permanece invariable: nunca es azul (por lo tanto, no es posible encenderla).

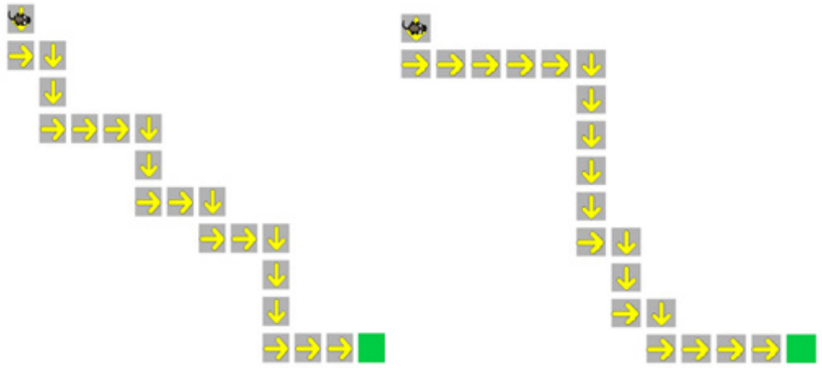
Las instrucciones primitivas son las siguientes:



La solución del problema es similar a la de *Tres naranjas*, con la diferencia de que la acción de *avanzar* es posterior a la otra acción del par (en este caso, *prender si es luz)* y que las dos acciones deben repetirse 10 veces. Se indica que el robot ejecute *prender si es luz* antes de comenzar a avanzar porque la primera celda puede tener una luz; la secuencia se repite 10 veces (y no 11) porque la última celda nunca tendrá una luz.

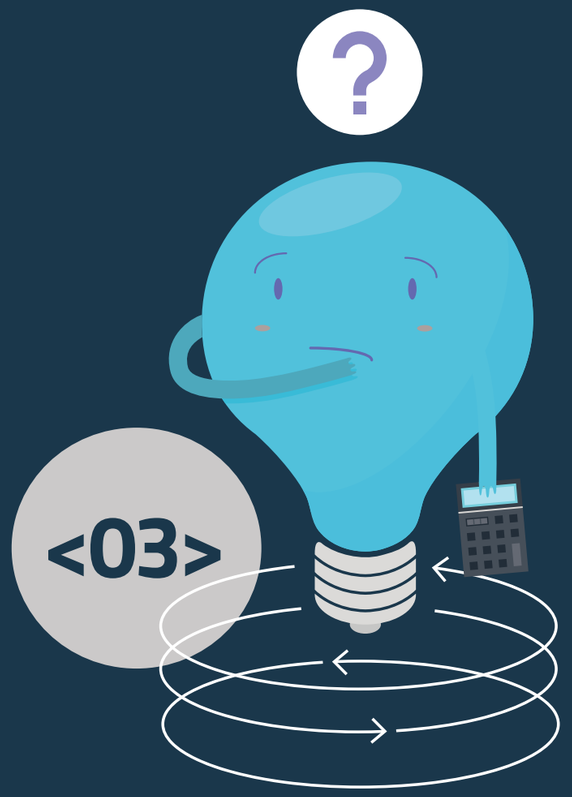
Laberinto largo **http://**scratch.mit.edu/projects/42293904/#editor

En esta actividad de Scratch, se retoma el problema del laberinto, pero ampliado. Anteriormente los movimientos del ratón se limitaban a avanzar a una casilla, ubicada a la derecha o abajo. Ahora, el ratón debe avanzar veinte veces, en esas dos direcciones; las acciones primitivas son, por lo tanto, las mismas.



Aunque a algunos alumnos el problema les pueda parecer a primera vista bastante complejo, la forma de resolverlo es básicamente igual a la empleada en el caso del laberinto corto. La única diferencia radica en la repetición de la acción de *avanzar*. De este modo, pueden formularse las siguientes soluciones:

</02>



REPETICIÓN CONDICIONAL

Luego de haber trabajado en la creación de programas para escenarios fijos y cambiantes, se avanza ahora un paso más, incorporando el análisis de escenarios en los que no solo hay una variación entre dos alternativas, sino también de la secuencia (es decir, del número de unidades o de elementos del escenario, como, por ejemplo, la cantidad total de casillas o de filas).

Secuencia didáctica 9

Escenarios con secuencias de tamaño variable

En esta secuencia se trabajará con una actividad de Scratch en la que, como en otras ocasiones, se presenta un escenario similar al de Lightbot. La propuesta es que los alumnos se introduzcan en el concepto de repetición condicional, a través de la resolución del desafío que plantea el proyecto.

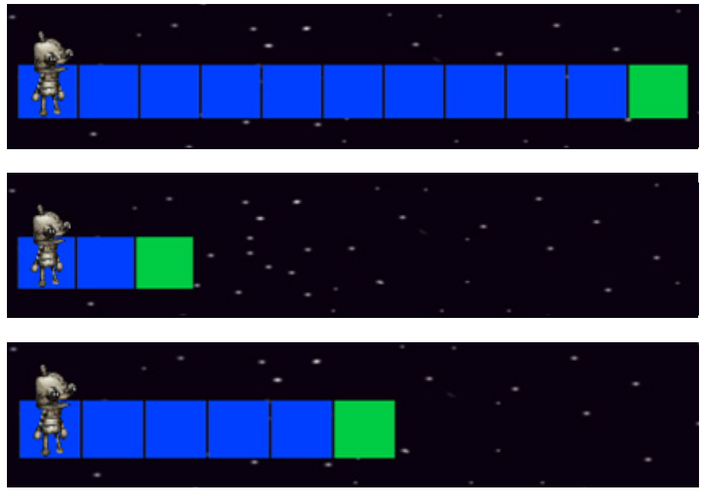
Objetivos

- Distinguir entre escenarios con secuencias de tamaño fijo y escenarios con secuencias de tamaño variable.
- Comprender la noción de repetición condicional y reconocer las dos partes de las que consta toda repetición condicional.
- Identificar la estructura de un problema y utilizar esa información para resolverlo.

Desarrollo

http://scratch.mit.edu/projects/42293454/#editor

Los alumnos ingresarán a *Super Lightbot 1*. El escenario que se presenta es, a primera vista, similar al de la actividad *Lightbot recargado*, con la cual se trabajó anteriormente. Sin embargo, al presionar reiteradamente el ícono , podrán notar que, a diferencia de lo que ocurría en *Lightbot recargado*, donde la cantidad total de celdas era siempre la misma (y solo cambiaba la distribución de las celdas grises y azules), aquí es la cantidad de celdas –todas azules, excepto la última– lo que varía. Mientras que en el caso anterior había una secuencia que tenía un *tamaño fijo*, en este se trata de una **secuencia de tamaño variable**.

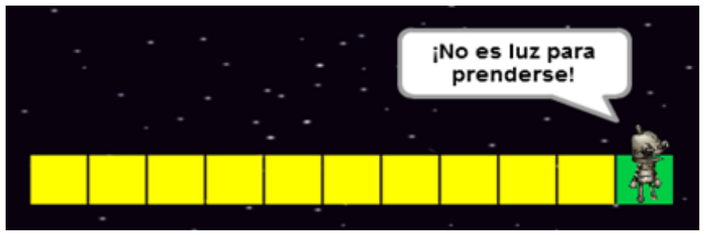


Actividad

Al igual que en Lightbot (y en sus variantes en Scratch), el objetivo es encender las casillas o celdas azules, sin intentar encender las que no son de ese color. El hecho de que ahora se presente una secuencia de tamaño variable añade una mayor dificultad al juego.

Una vez planteado el problema, se puede indicar a los alumnos que intenten resolverlo con los bloques empleados en *Lightbot recargado* que consideren útiles. A través de esta práctica, se pretende que comprueben por sí mismos que el problema no puede ser resuelto con una repetición simple (esto es, mediante el bloque *repetir*), sencillamente porque se ignora cuántas veces se debe repetir la tarea. Si se considera más conveniente, puede dejarse de lado la práctica y enfocarse directamente en la discusión acerca de cómo encarar la resolución del problema.
No obstante lo dicho, puede ocurrir que algunos alumnos, tras haber comprobado que la variabilidad de la secuencia es limitada (es decir, que, por ejemplo, nunca hay más de 11 casillas en total), propongan como solución la siguiente secuencia de comandos:

La ejecución de un programa como este dará lugar a situaciones en las que el robot intente encender la última casilla de la fila (la única que no es azul), lo cual es un error que el mismo autómata señalará:



Por lo tanto, en este juego –conviene insistir a los alumnos–, el robot debe prender las luces hasta que se tope con una casilla verde. Esta circunstancia es una condición que puede ser utilizada para limitar las veces que se repite una tarea. Mediante el bloque **repetir hasta que...** (que se encuentra en la categoría *Control*) es posible dar esa indicación*.

*Una variante de **repetir hasta que** es **mientras** (*while*, en inglés), que es la versión utilizada en la mayoría de los lenguajes de programación industriales. Ambas son formas de repeticiones condicionales, pero la primera sigue repitiendo algo *hasta que* la condición se cumpla; la otra variante repite algo *mientras* la condición se cumpla o, en otras palabras, hasta que deje de cumplirse.

Actividad

Hecha esta salvedad, se les podrá proponer a los alumnos que intenten resolver el problema nuevamente, pero utilizando ahora los siguientes bloques:

La solución es esta:

Una vez que los alumnos hayan realizado la actividad, se les podrá explicar que el comando empleado en la resolución del problema se conoce como **repetición condicional**. Una repetición condicional consta de dos partes:

- Una condición ante la cual el autómata (es decir, el programa) deja de repetir algo;
- La tarea que el autómata/programa debe repetir hasta que la condición se cumpla.

En este caso, la condición (expresada en el bloque *¿tocando fin?*) es la última casilla de la secuencia. Puede ser la segunda casilla, si se trata de una secuencia formada por dos casillas; la cuarta, en una secuencia de cuatro casillas; la décima, si la secuencia consta de diez casillas, etcétera. La tarea que se repite es la de encender la luz de la casilla azul y avanzar.
Al ejecutar la repetición condicional, el autómata/programa ha realizado un *recorrido* a lo largo de todas las celdas, procesando una tarea a continuación de otra. Los recorridos son formas de *procesar* (es decir, de realizar tareas con) objetos dispuestos en forma estructurada (por ejemplo, una fila de celdas contiguas). Un recorrido consta de los siguientes elementos:

- Un comando de repetición (**repetir... o repetir hasta que...**), junto con una forma de acotar el recorrido. Esto puede hacerse indicando el número de objetos –si se conoce ese dato (en cuyo caso se usa el comando **repetir**)– o bien mediante la identificación del último objeto de la serie (en cuyo caso se emplea **repetir hasta que...**);
- Un comando (primitivo o construido por el programador) para procesar cada uno de los objetos (en este caso, **prender luz**);
- Un comando para pasar al siguiente elemento de la estructura (en este caso, el comando **avanzar**).

Cierre

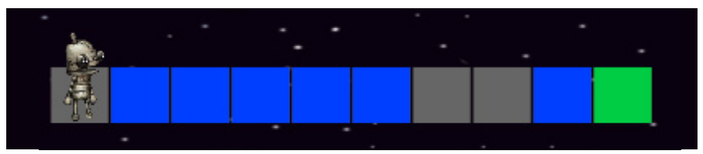
Como cierre de esta secuencia, se sugiere volver sobre los conceptos explicados (repetición condicional, estructura de un problema, recorrido). También se les pueden proponer a los alumnos situaciones en las que no se cumplan los requisitos de un recorrido (por ejemplo, escenarios en los que no sea posible identificar una estructura o escenarios con secuencias de tamaño variable en los que el último objeto no se distinga del resto) para que determinen si, en esos casos, puede aplicarse la solución usada en *Super Lightbot 1*.

Ejercitaciones

Ente las actividades de Scratch de esta sección, que plantean problemas cuya resolución implica el uso de repeticiones condicionales, la tercera *(El detective Chaparro)* y la última *(El superviaje)* pueden resultarles a los alumnos particularmente complejas, por lo que se sugiere dedicarles especial atención. Para que se involucren más en la superación del desafío que presentan, pueden reemplazarse los personajes por otros más cercanos al universo de los alumnos.*

Super Lightbot 2 **http://**scratch.mit.edu/projects/42293670/#editor

Esta adaptación de Lightbot plantea un desafío semejante al de la actividad de Scratch anterior, con una secuencia de tamaño variable, pero con una dificultad adicional: no todas las casillas ubicadas antes del final de la fila son azules, es decir, no todas puede encenderse. Haciendo clic varias veces sobre la banderita verde, se puede observar la estructura del problema.



Como sucedía en *Super Lightbot 1*, si se intenta encender una celda que no es azul, se produce un error:

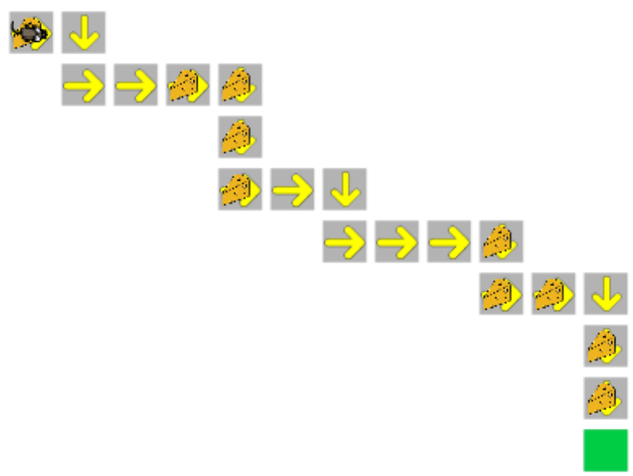
*Para saber cómo reemplazar o modificar los personajes de las actividades de Scratch, consúltese la pág. 115 de este cuaderno.

Sin embargo, antes había una sola casilla que no podía encenderse: la última. Ahora puede haber más de una casilla de este tipo, y no hay forma de saber de antemano cuáles serán (a excepción de la última casilla, que siempre es verde). Este problema ya había surgido al trabajar con alternativas condicionales (por ejemplo, en el proyecto *Lightbot recargado*). Por lo tanto, en la solución deberán combinarse dos estrategias (ambas ya utilizadas anteriormente): una que le permita al robot saber si cada casilla que pisa es azul, de modo que encienda la luz solo en el caso de que así sea, y otra que en la que se especifique que, ante cierta condición, el robot cesa de repetir esa tarea. Tras estas indicaciones, se podrá dejar que los alumnos intenten resolver el problema por su cuenta. De ser necesario, se los ayudará con algunas pistas, sugiriéndoles, por ejemplo, que creen un nuevo bloque para que el autómata sepa si debe encender la casilla o que utilicen el **bloque si... entonces**. Se espera que puedan formular la solución en estos términos:

Laberinto con queso **http://**scratch.mit.edu/projects/42297852/#editor

En este laberinto, el ratón, además de avanzar en la dirección que corresponda (hacia abajo o hacia la derecha), debe comer los trozos de queso que puedan encontrarse en algunas celdas. Tanto la ubicación y la cantidad de los trozos de queso como el largo del laberinto son desconocidos y pueden variar de una a otra ejecución.

Las instrucciones primitivas son las siguientes:



Con el fin de orientar a los alumnos en la resolución del problema, se podrá realizar un repaso de las soluciones aplicadas en los laberintos anteriores de manera que, tal como hicieron en otras ocasiones, puedan establecer cuáles de las estrategias empleadas pueden aplicarse a este caso y cuáles no. Una solución es la siguiente:

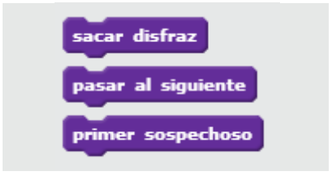
El detective Chaparro

http://scratch.mit.edu/projects/42298168/#editor

En este juego, un detective tiene que descubrir al culpable de un crimen que se oculta disfrazándose de otra persona. Para desenmascararlo, debe interrogar a cada sospechoso, comenzando por el primero de la izquierda. Siempre está el criminal, que suplanta la identidad de alguno de los personajes que se ven en la pantalla, pero no es posible saber a quién suplanta en cada ocasión.

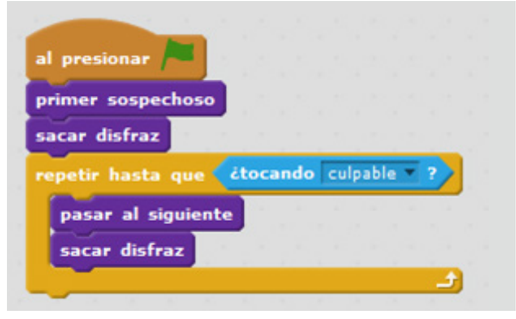
Al presionar varias veces la banderita verde se puede observar que, mientras la posición inicial en la que se encuentra el detective varía, los sospechosos son siempre los mismos y aparecen siempre en el mismo orden. Un primer aspecto que conviene tener en cuenta para poder resolver correctamente el juego es que el detective invariablemente debe empezar su pesquisa dirigiéndose al primer sospechoso, más allá

de que al comienzo se encuentre frente al segundo, el cuarto o el último de la fila; de lo contrario, puede ocurrir que deje sin interrogar a algunos sospechosos, entre los cuales podría hallarse el villano. Esta indicación forma parte de las instrucciones primitivas, que en total son tres:



La segunda instrucción de la lista, al igual que la última, es clara; la primera, cuando se ejecuta, hace que el detective se fije si el sospechoso que tiene frente a sí es el villano. Para resolver el problema también será necesario el siguiente bloque de la opción *Sensores*:

Con estas indicaciones y los conocimientos adquiridos a partir de la realización de las actividades anteriores, los alumnos podrán tratar de resolver el problema, haciendo previamente hincapié en las cuestiones en las que pueda haber mayores dificultades. Por ejemplo, es probable que algunos alumnos piensen que, como el número de sospechosos es fijo, basta con usar una repetición simple que contenga las indicaciones *pasar al siguiente* y *sacar disfraz*, luego de acercarse al primer sospechoso. Este planteo puede aprovecharse para el intercambio de ideas en la clase. Se espera que arriben a la conclusión de que no es posible utilizar una repetición simple en este caso, sencillamente porque, aunque los sospechosos sean siempre siete, el villano puede encontrarse *antes* de que el detective termine de pasar por los siete. En algunas ocasiones, puede ser necesario repetir la tarea siete veces y en otras no.
Una posible solución es la siguiente:



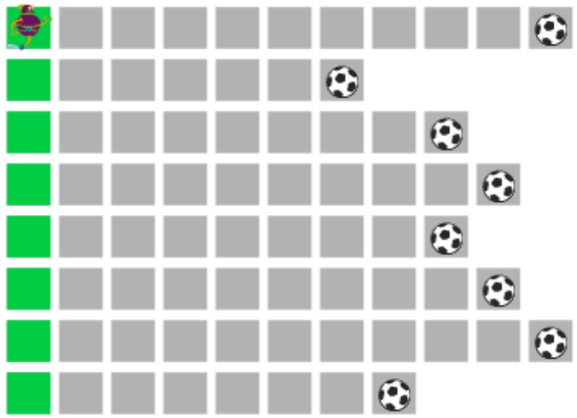
Vale aclarar que es importante las instrucciones dentro del bloque *repetir hasta que…*, sea primero *pasar al siguiente*, y luego *sacar el disfraz*.

Fútbol para robots

http://scratch.mit.edu/projects/42294000/#editor

En esta actividad de Scratch, se presenta un escenario de ocho filas de tamaño variable. Un robot debe recorrer cada fila hasta alcanzar la pelota que se encuentra en el extremo y patearla. Para pasar de una fila a la siguiente, el robot tiene que estar sobre la casilla verde, ubicada al inicio, por lo que luego de patear cada pelota debe volver sobre sus pasos.

Las instrucciones primitivas son las siguientes:

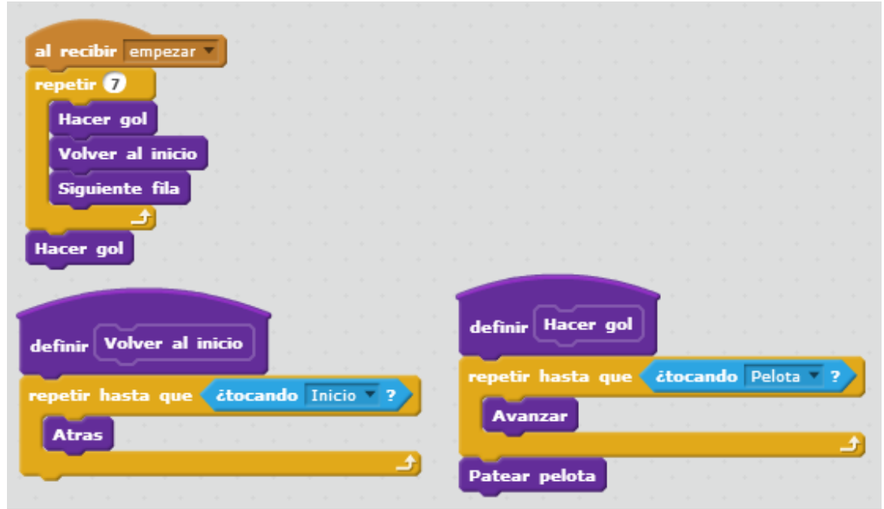


En este caso, para resolver el problema, lo más conveniente es dividirlo en subtareas. Entre estas subtareas se destacan las siguientes:

• Una subtarea (a la que se la puede llamar *hacer gol*) que hace que el robot se dirija hasta la pelota y la patee;
• Otra (denominada, por ejemplo, *volver al inicio*) que haga que el robot retroceda lo necesario para volver a la casilla verde.

Luego de discutir en clase las posibles vías para resolver el problema y, una vez planteada la ventaja de definir las dos subtareas mencionadas, puede dejarse que los alumnos las definan y, a continuación, establezcan el programa más adecuado para que el robot patee todas las pelotas. A manera de ayuda, se sugiere hacerles notar que la tarea de hacer un gol es similar a otras vistas en actividades de Scratch anteriores (como *prender luz*, en *Lightbot*, o *sacar disfraz*, en *El detective Chaparro*): se trata de una acción que solo debe ejecutarse si se cumple una condición determinada (en Lightbot, esta condición era que el robot se encontrara en un casilla azul; en *El detective Chaparro*, que el detective estuviera frente a cada uno de los sospechosos). También es oportuno señalar que, en la secuencia de comandos del programa (a diferencia de lo que sucede en la definición de las subtareas), es posible utilizar una repetición simple, ya que siempre hay ocho filas. Esta secuencia incluirá las dos subtareas mencionadas y la acción de pasar a la fila siguiente. Vale aclarar que aunque las filas son ocho, el robot cambia de fila siete veces, por lo que la secuencia se repetirá solo ese número de veces.

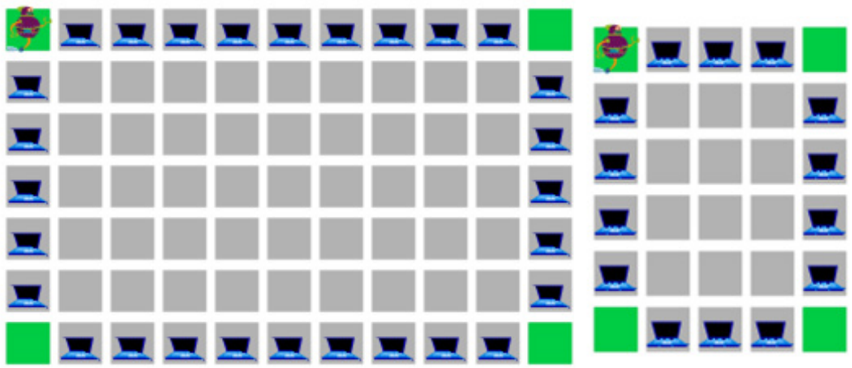
Una posible solución es la siguiente:



Prendiendo las compus

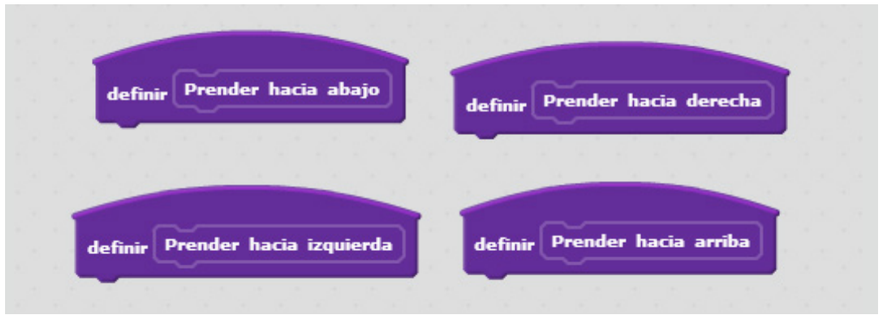
http://scratch.mit.edu/projects/42293830/#editor

Aquí, el escenario es una serie de computadoras que forman un rectángulo de tamaño variable, es decir, cuyo ancho y alto cambian cada vez que se presiona la banderita verde. Un robot debe dirigirse a cada computadora y encenderla.



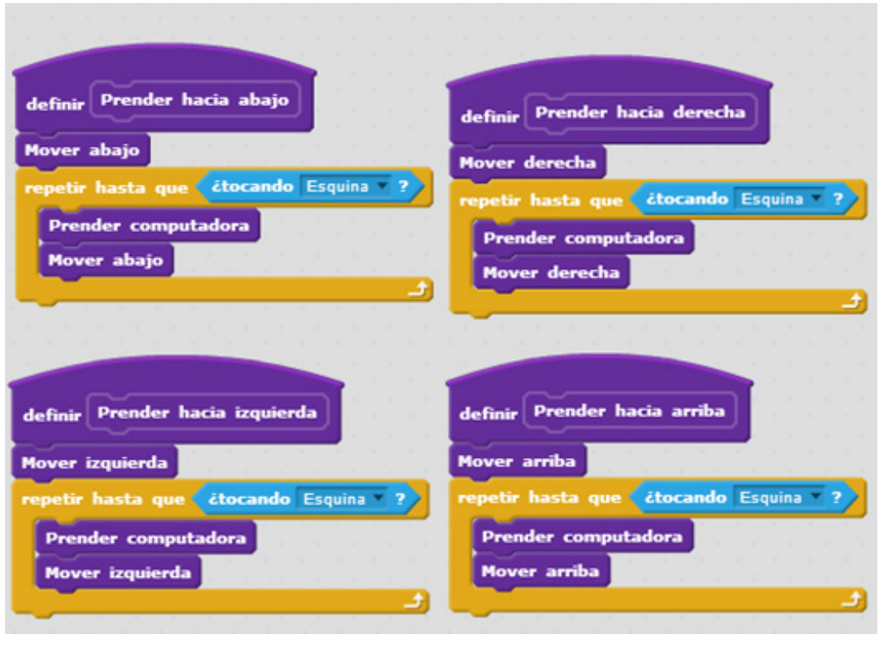
Las instrucciones primitivas son las siguientes:

Como en otras oportunidades, la resolución del problema requiere definir varios procedimientos. Una buena división en subtareas debería conducir a una propuesta como la que sigue:



Cada una de estas subtareas resuelve un lado específico del rectángulo. Todas ellas permiten definir alguno de los siguientes programas:

La definición de cada una de las cuatro subtareas anteriores sería esta:

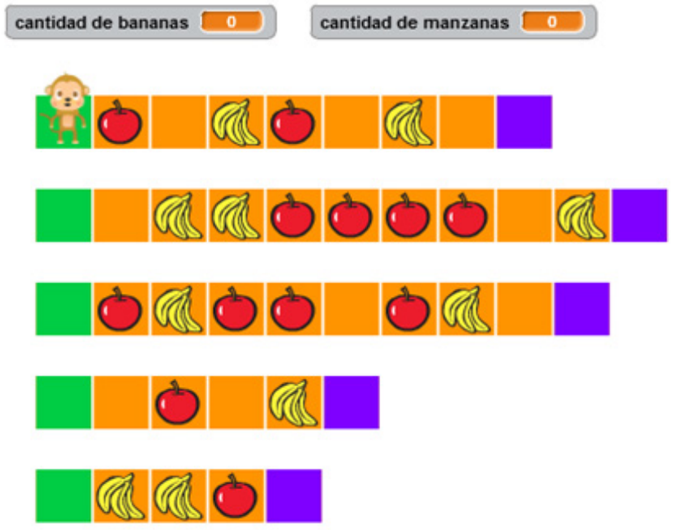


Nótese que, como el robot comienza el recorrido de cada lado del rectángulo por una esquina, primero debe moverse un paso en la dirección del lado que recorrerá. De ahí que cada subtarea comience con el subcomando *mover <dirección>*.

El mono que sabe contar

http://scratch.mit.edu/projects/42292960/#editor

Mediante esta actividad de Scratch se pretende aprovechar todo lo aprendido hasta ahora. Como en ocasiones anteriores, hay que hacer clic sobre la banderita verde varias veces para poder ver las variantes y, a la vez, la estructura del escenario. Este consiste en cinco filas, a lo largo de las cuales se distribuyen de manera aleatoria manzanas y bananas. Tanto el largo de las filas como las bananas y las manzanas que se encuentran en cada una de ellas varían. El autómata está representado por un mono.



El objetivo es que el mono recorra cada fila pasando por cada casilla y contando cada manzana y cada banana que encuentra en su recorrido, pero de acuerdo con una serie de reglas que se describen a continuación.

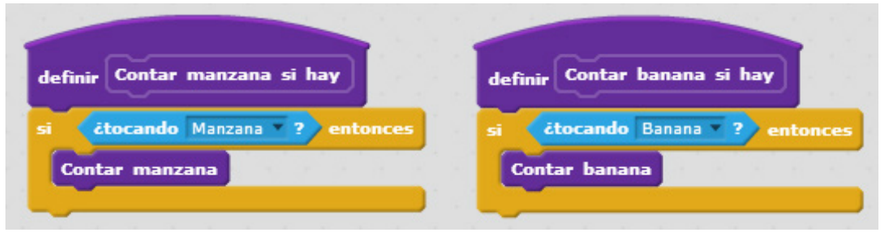
- **Solo puede contar una manzana o una banana si está parado sobre una casilla que contenga alguna de esas frutas;**
- **Solo puede avanzar o volver hacia atrás si hay una celda a la cual moverse;**
- **Solo puede pasar a la siguiente fila si está parado en una casilla verde.**

Las instrucciones primitivas son las siguientes:

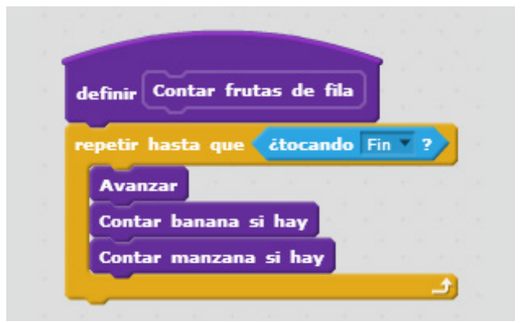
Puesto que el desafío es bastante complejo, se sugiere guiar a los alumnos en su resolución paso a paso: primero se buscará una solución para una de las filas, y luego se generalizará esa solución para todas. Al igual que otras veces, se les puede dar la consigna de que establezcan las subtareas que consideren necesarias para lograr el objetivo. Al respecto, las primeras subtareas que habría que resolver son las siguientes:

- **Contar una manzana si la hay;**
- **Contar una banana si la hay;**
- **Recorrer toda una fila;**
- **Volver hacia la casilla verde para poder pasar a otra fila.**

Una vez resueltas estas cuatro subtareas, el resto del juego será muy sencillo. Los alumnos deberían estar en condiciones de definir correctamente cada una de ellas, comenzando por las dos primeras, que quedarían así:



Estas dos subtareas servirán, a su vez, para definir la tercera subtarea, en la que el mono tiene que recorrer toda la fila hasta llegar al último casillero (marcado con color violeta) y, a medida que avanza, contar las frutas con las que se tope.



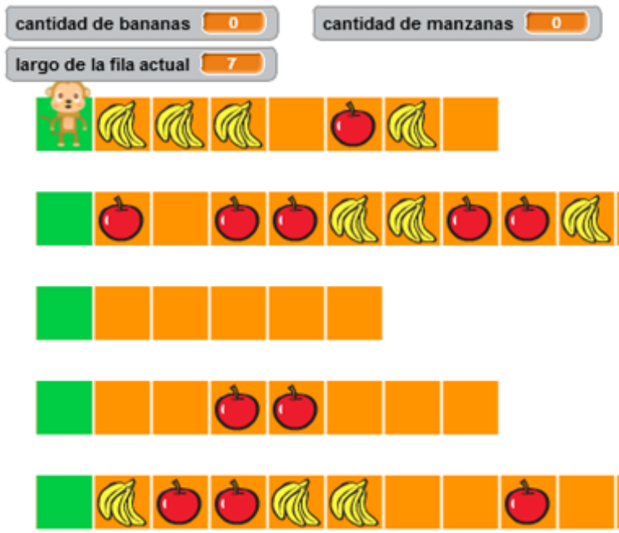
La última subtarea es fácil de definir:

Resta definir el programa. De manera similar a lo que ocurría en Fútbol para robots, la cantidad de filas es siempre la misma, por lo que puede usarse una repetición simple. La secuencia deberá repetirse solo cuatro veces (y no cinco, que es el número total de filas), ya que son cuatro las ocasiones en que el mono debe pasar de una fila a otra. La última fila tendrá que ser tratada de una manera distinta a las anteriores. Así, la definición de este procedimiento queda como sigue:

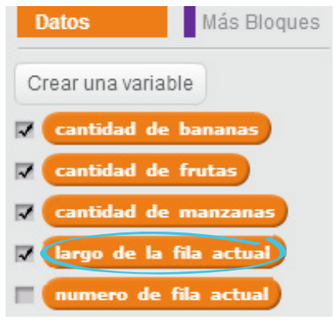
El mono cuenta de nuevo

http://scratch.mit.edu/projects/42293342/#editor

Se trata de una variante del ejercicio anterior. Tanto el objetivo como las reglas siguen siendo los mismos, pero hay una importante diferencia: la última celda no está diferenciada de ninguna manera (en el caso anterior, era de color violeta), de manera que se dificulta la tarea de identificarla. No es posible utilizar ahora el bloque *repetir hasta que...* simplemente porque no hay una condición que le permita al mono frenar su avance a lo largo de cada hilera.



Interesa que los alumnos comprendan la dificultad mencionada para que puedan entender luego el modo en que es posible resolverla. Eventualmente, podrán intentar plantear alguna solución al respecto. La respuesta hay que buscarla en la opción *Datos*. Allí se encuentra un bloque llamado *largo de la fila actual*:

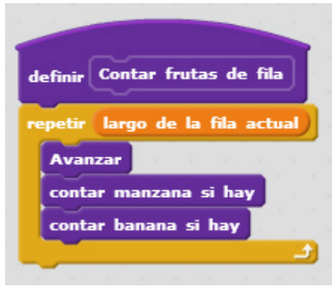


Ese bloque permite, precisamente, que el mono *sepa* la cantidad de casillas o celdas que posee la fila en la que se encuentra. Al utilizarlo, esa cantidad aparecerá en una de las pequeñas ventanas alargadas ubicadas justo arriba de la primera fila.

Hay que tener presente que, como la primera casilla no se cuenta, lo que indica el número es la cantidad de celdas hacia la derecha del mono. Así, *largo de la fila actual* tiene en este caso un valor igual a 5 porque el mono está parado sobre una fila que tiene seis casillas en total, contando la casilla verde.

El bloque largo de la fila actual puede usarse junto con el bloque *repetir*. De esta manera, el mono logrará recorrer la fila hasta el final sin contar con la marca que indicaba la última casilla:

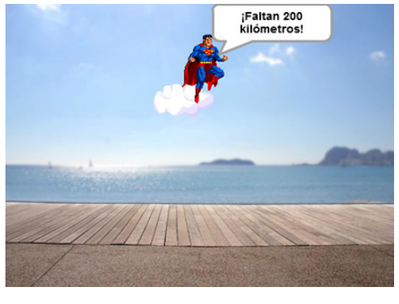
La solución de esta actividad es igual que la anterior, solo que en la definición de contar frutas de fila el bloque de repetición condicional repetir hasta tocando fin se reemplaza por el bloque repetir largo de la fila actual:



El superviaje

http://scratch.mit.edu/projects/42295788/#editor

En esta actividad de Scratch, un superhéroe debe llegar a una ciudad. Esta se encuentra a una distancia que varía cada vez que se reinicia la presentación (es decir, cada vez que se presiona la banderita verde), entre 50 km y 200 km.

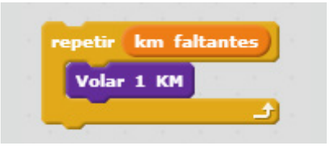


Se da una sola primitiva: Volar 1 KM

Cuando se ejecuta este comando, la distancia que debe recorrer nuestro superhéroe se acorta en una unidad:

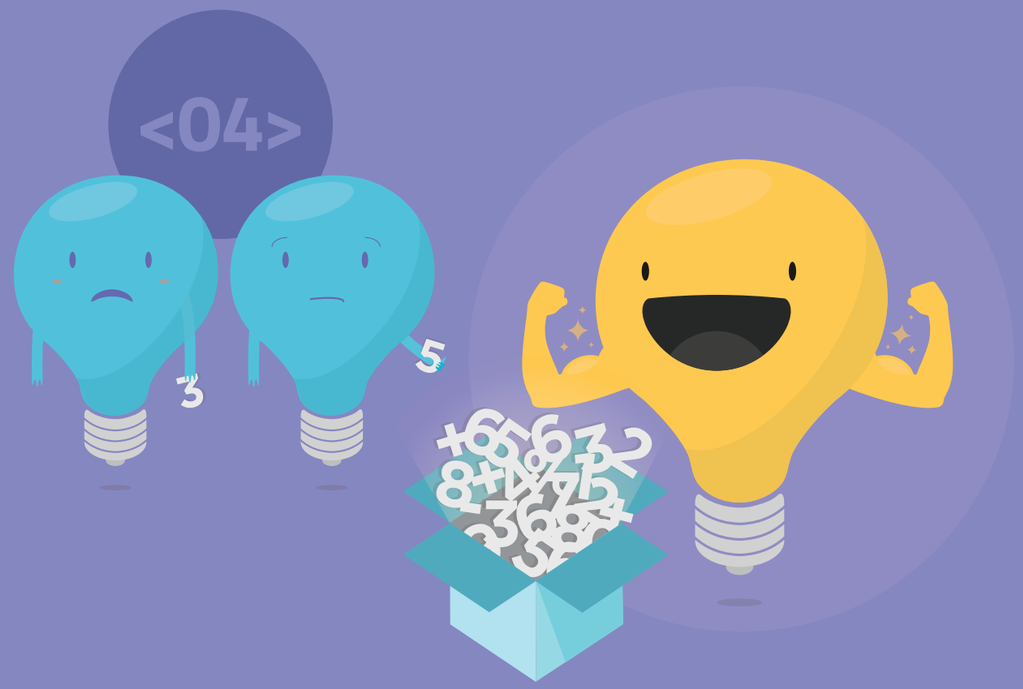
También se provee una variable relativa a la distancia restante hasta la ciudad. Esta variable se encuentra en la categoría *Datos:*

A partir de esta información, se sugiere permitir a los alumnos resolver la actividad como les sea posible. La solución más sencilla –y que probablemente propongan los alumnos– consiste en utilizar una repetición simple:



A manera de cierre de este apartado, resultará útil repasar las características de la repetición simple y la condicional, señalando en qué casos conviene usar cada una de ellas.

</03>



PARAMETRIZACIÓN DE SOLUCIONES

En esta parte, se aborda la creación de parámetros, es decir, de datos que, al ser recibidos por un procedimiento, hacen que la ejecución de este varíe. Se trata de uno de los temas más interesantes de la programación informática. El empleo de parámetros permite reducir notoriamente la extensión de los programas e incrementar su versatilidad. En las secuencias que siguen, se propone comenzar con la identificación de pequeñas variaciones entre grupos de componentes similares para luego, en un mayor nivel de formalización, reconocer valores que describen datos y, desde allí, arribar a la noción de parámetro.

Secuencia didáctica 10

Canciones y estribillos

En esta secuencia se trabajará con letras de canciones o poemas que contengan estribillos, con el fin de que los alumnos comprendan que los estribillos de las canciones representan procedimientos. Asimismo, el análisis de la estructura general de este tipo de composiciones permitirá introducir el tema de la parametrización en informática.

Objetivos

- Identificar la estructura de una secuencia.
- Comparar secuencias similares y reconocer en ellas datos que varían.
- Introducirse en los conceptos de parámetro y parametrización.
- Interpretar la parametrización como una estrategia complementaria al procedimiento.

Desarrollo

Para introducir el tema, se invitará a los alumnos a reflexionar acerca de las semejanzas entre los estribillos de las composiciones poéticas o musicales y muchos de los procedimientos empleados en programación. En ambos casos, se trata de una repetición que se realiza con cierta regularidad: en un poema o una canción, de la repetición de un grupo de versos entre una estrofa y otra; en un programa –y también en un procedimiento–, de la de una secuencia de acciones. Así, las funciones de Lightbot no eran otra cosa que procedimientos que se repetían, y en los ejercicios *Lightbot en Scratch* y *El recolector de estrellas*, la repetición de una secuencia de acciones (el movimiento en diagonal del robot y en horizontal del extraterrestre) formaba parte de la definición misma de un procedimiento.

Como se recordará, los procedimientos cumplen dos funciones:

- Ayudan a separar una acción o una tarea en varias subtareas más acotadas, para poder plantear una solución más compleja a un problema determinado (lo que se conoce como **división en subtareas**);
- Reúnen una secuencia de pasos bajo un nombre, de manera que permiten ejecutar esa secuencia con solo utilizar el nombre que se le haya asignado.

De estos aspectos, interesa ahora el último. La propuesta consiste en volver sobre la utilidad de asignar nombres a secuencias de acciones (ya que permiten simplificar la solución de un problema), a través de la realización de un proceso análogo con la letra de una canción. Se recomienda trabajar con una canción que los alumnos conozcan.

Deberán seguir los pasos que se mencionan a continuación.

1) Marcar el grupo de versos que se repite, es decir, el estribillo. Aunque puede ocurrir que las estrofas contengan alguna frase o verso que también se repita, solo se tendrán en cuenta los grupos enteros de versos.

2) Extraer el grupo de versos que se repite y ubicarlo al final, precedido del rótulo *[Estribillo]*. Si el estribillo ya se encuentra al comienzo de la composición, bastará con rotularlo.

3) En cada uno de los lugares en los que aparezca el estribillo, reemplazarlo por el rótulo *[Estribillo]*.

A continuación, se ejemplifica la realización de esta actividad con el poema de Federico García Lorca “La Tarara”, basado en diversas coplas populares españolas.*

Canción	**Canción**
La Tarara, sí; la tarara, no; la Tarara, niña, que la he visto yo.	[Estribillo]
Lleva la Tarara un vestido verde lleno de volantes y de cascabeles.	Lleva la Tarara un vestido verde lleno de volantes y de cascabeles.
La Tarara, sí; la tarara, no; la Tarara, niña, que la he visto yo.	[Estribillo]
Luce mi Tarara su cola de seda sobre las retamas y la hierbabuena.	Luce mi Tarara su cola de seda sobre las retamas y la hierbabuena.
	[Estribillo]
Ay, Tarara loca. Mueve, la cintura para los muchachos de las aceitunas.	Ay, Tarara loca. Mueve, la cintura para los muchachos de las aceitunas.
La Tarara, sí; la tarara, no; la Tarara, niña, que la he visto yo.	[Estribillo]
	La Tarara, sí; *la tarara, no;* *la Tarara, niña,* *que la he visto yo.*

- Dos formas distintas de escribir la misma canción -

La actividad permite extraer algunas conclusiones, que se sugiere discutir con los alumnos. Por un lado, es obvio que, al leer o cantar el *estribillo*, no se pronuncia la palabra *estribillo*, sino que lo que se hace es buscar el contenido de la categoría *estribillo* (ubicada al comienzo) y leer o cantar los versos que lo forman. Por otro lado, como se puede observar en el ejemplo, el reemplazo del estribillo por su sola indicación mediante el rótulo *[Estribillo*] permite ganar espacio y evitar la engorrosa tarea de tener que copiar una y otra vez el mismo texto. En algunas canciones, puede suceder que el estribillo se repita dos veces al final. Esto (que, por lo general, en los impresos o manuscritos se indica mediante la palabra *bis*) se puede reemplazar por la expresión *x2*, más próxima al concepto de repetición ya aprendido: *[Estribillo] x2*
¿A qué se parece todo esto? Tal como se señaló al comienzo, la metodología es básicamente la misma que la empleada al programar: se identifica la secuencia de comandos que se repite, se le asigna un nombre y de ahí en adelante se utiliza solo ese nombre en lugar de toda la secuencia.

*Existen diferentes versiones cantadas del poema. Algunas de ellas pueden escucharse y verse en http://bit.ly/1zp3cFU

Luego de esta actividad, se sugiere continuar el análisis con la letra de alguna canción en la que existe una parte que se repite, pero cambia ligeramente en algo. Un ejemplo de este tipo de canciones es la famosa canción *Un elefante se balanceaba.* Si se les pregunta a los alumnos qué es lo que se repite y qué es lo que varía en este caso, seguramente todos coincidan en que el texto es siempre el mismo, salvo por el número de elefantes. Para que comprendan cómo esta estructura se puede expresar en términos abstractos, convendrá comenzar por transcribir al menos las primeras variaciones de la letra de la canción:

Un elefante se balanceaba
sobre la tela de una araña,
como veía que resistía
fue a llamar a otro elefante.

Dos elefantes se balanceaban
sobre la tela de una araña,
como veían que resistía
fueron a llamar a otro elefante.

Tres elefantes se balanceaban
sobre la tela de una araña,
como veían que resistía
fueron a llamar a otro elefante.

Cuatro elefantes se balanceaban
sobre la tela de una araña,
como veían que resistía
fueron a llamar a otro elefante.

A partir de esto, lo que puede hacerse para simplificar la transcripción de la letra de la canción es, en primer lugar, indicar mediante un *hueco* lo que cambia cada vez:

__elefante(s) se balanceaba(n)
sobre la tela de una araña,
como veía(n) que resistía
fue(ron) a llamar a otro elefante *

El hueco representa la parte que falta para completar una instrucción. Esta parte podría enunciarse como: “Elija usted el número que desea para usar en ese hueco y tendrá la canción para la parte en la que se dice esa cantidad de elefantes”. Para alguien que no conoce la canción, el hueco solo no significará nada; necesitará que se le indique de alguna manera que ese hueco se llena exactamente con un número. Para eso, sobre esa línea, puede ponerse un nombre que refiera a aquello que puede ir en ese lugar. En este caso, el nombre puede ser *número o cantidad*. De este modo, el texto quedaría así:

*Desde el punto de vista estrictamente gramatical, en la letra de la canción hay una variación dada por el paso del singular al plural (indicada aquí mediante la puesta entre paréntesis de los morfemas correspondientes a este último). No obstante, a los efectos de la actividad que aquí se plantea, dicha variación puede considerarse irrelevante.

Cantidad elefantes se balanceaba(n)
sobre la tela de una araña,
como veía(n) que resistía
fue(ron) a llamar a otro elefante

Si se considera la parte de la canción que no varía como un estribillo (aunque estrictamente no lo sea), se obtiene la siguiente fórmula:

Canción
Un elefante se balanceaba
sobre la tela de una araña,
como veía que resistía fue
a llamar a otro elefante.

[Estribillo dos]

[Estribillo tres]

[Estribillo cuatro]
(... así para todos los números que siguen)

Estribillo cantidad

Cantidad elefante(s)
se balanceaba(n)
sobre la tela de una araña,
como veía(n) que resistía
fue(ron) a llamar a otro elefante

El análisis de este ejemplo servirá para indicar que lo que se ha denominado informalmente *hueco* en programación se conoce como **parámetro**, y al proceso de agregar un parámetro a un procedimiento, **parametrización**.

Actividad

Con el fin de que los alumnos apliquen lo aprendido hasta aquí, se sugiere el trabajo con la letra de otra canción (u otra composición poética) en la que claramente puedan distinguirse partes fijas y variaciones. A modo de ejemplo, se propone *El viejo MacDonald tenía una granja**. Se trata de una canción bastante conocida, de la que existen varias versiones de la letra que difieren ligeramente entre sí. Una de ellas es la siguiente:

Canción
El viejo MacDonald tenía una
granja, ¡ia, ia, io!
Y en su granja tenía una vaca,
¡ia, ia, io!,
con un *muu* por aquí,
con un muu por allá,
en todos lados *muu muu*.
El viejo MacDonald tenía una
granja, ¡ia, ia, io!

*Puede escucharse una versión de esta canción en http://bit.ly/1oWD9F5

El viejo MacDonald tenía una
granja, ¡ia, ia, io!
Y en su granja había un cerdito,
¡ia, ia, io!,
con un *oink* por aquí,
con un oink por allá,
por todos lados *oink oink*.
El viejo MacDonald tenía una
granja, ¡ia, ia, io!

El viejo MacDonald tenía una
granja, ¡ia, ia, io!
Y en su granja tenía un caballo,
¡ia, ia, io!,
con un *eee* por aquí,
con un *eee* por allá,
por todos lados *eee eee*.
El viejo MacDonald tenía una
granja, ¡ia, ia, io!

El viejo MacDonald tenía una
granja, ¡ia, ia, io!
Y en su granja había un gallito,
¡ia, ia, io!,
con un *quiquirí* de aquí,
un *quiquirí* de allá,
por todos lados *quiquirí*.
El viejo MacDonald tenía una
granja, ¡ia, ia, io!

Actividad

Luego de leer la letra de la canción (y de escucharla, si no la conocen), se puede dejar que los alumnos determinen qué es lo que se repite y qué es lo que varía en ella. Una vez establecido esto, deberán modificar la letra de acuerdo con estas indicaciones.

- **Marcar los dos parámetros que hay en las estrofas y poner un nombre a cada parámetro;**
- **Armar el estribillo de la canción utilizando los parámetros identificados;**
- **Formular la estructura general de la canción de manera similar a como se hizo anteriormente con otras canciones.**

Se espera que los alumnos reconozcan que hay un parámetro para el animal y otro para el sonido que produce el animal. Aunque lo mejor es que ellos mismos piensen el nombre de cada parámetro, se les puede orientar si se considera necesario. La solución es la siguiente:

El viejo MacDonald tenía una granja, ¡ia, ia, io!
Y en su granja tenía una vaca,
¡ia, ia, io!,
con un muu por aquí,
con un muu por allá,
en todos lados muu muu.
El viejo MacDonald tenía una granja, ¡ia, ia, io!

El viejo MacDonald tenía una granja, ¡ia, ia, io!
Y en su granja había un cerdito,
¡ia, ia, io!,
con un oink por aquí,
con un oink por allá,
por todos lados oink oink.
El viejo MacDonald tenía una granja, ¡ia, ia, io!

El viejo MacDonald tenía una granja, ¡ia, ia, io!
Y en su granja tenía un caballo,
¡ia, ia, io!,
con un eee por aquí,
con un eee por allá,
por todos lados eee eee.
El viejo MacDonald tenía una granja, ¡ia, ia, io!

El viejo MacDonald tenía una granja, ¡ia, ia, io!
Y en su granja había un gallito,
¡ia, ia, io!,
con un quiquirí de aquí,
un quiquirí de allá,
por todos lados quiquirí.
El viejo MacDonald tenía una granja, ¡ia, ia, io!

Canción
[Estribillo (una vaca) (*muu)*]

[Estribillo (un cerdito) (*oink)*]

[Estribillo (un caballo) (*eee)*]

[Estribillo (un gallito) (*quiquirí)*]

[Estribillo animal sonido]
El viejo MacDonald tenía una granja, ¡ia, ia, io!
Y en su granja tenía animal, ¡ia, ia, io!
con un sonido por aquí, con un sonido por allá,
en todos lados sonido sonido.
El viejo MacDonald tenía una granja, ¡ia, ia, io!

Cierre

Como cierre de la secuencia, se sugiere volver sobre la noción de parametrización y señalar que se trata de una herramienta muy poderosa utilizada en programación, que permite resumir en un solo procedimiento lo que de otra forma precisaría varios o, en algunos casos, un número que no es posible determinar en el momento de pensar el programa, como se verá más adelante. Eventualmente, se les podrá proponer a los alumnos que, en pequeños grupos, formulen la estructura general de una canción relativamente conocida y que, luego, los compañeros de otros grupos *descifren* la letra original.

Secuencia didáctica 11

Parámetros en Scratch

Continuando con el tema presentado en la secuencia anterior, en esta secuencia se aborda de manera específica la creación de procedimientos parametrizados, utilizando para ello una actividad de Scratch con un escenario fijo.

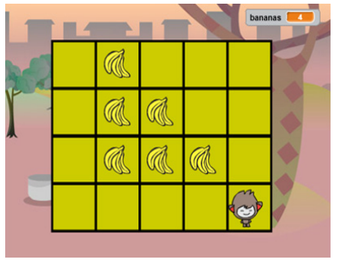
Objetivos

- Identificar los valores que cambian en secuencias formadas por comandos similares.
- Crear procedimientos para dichas secuencias, y nombrar adecuadamente los parámetros que pueden determinarse en ellas.
- Elaborar programas empleando procedimientos con parámetros creados por los alumnos.
- Valorar la utilidad de la parametrización para la resolución de problemas.

Desarrollo

http://scratch.mit.edu/projects/42292740/#editor

Se trabajará con la actividad de Scratch *El planeta de Nano*. El escenario que se presenta es parecido al de *El marciano en el desierto* (ya trabajado en la secuencia didáctica 5) y *El recolector de estrellas* (ejercicio I, 3), con la diferencia de que, en este caso, el extraterrestre tiene que comer bananas.

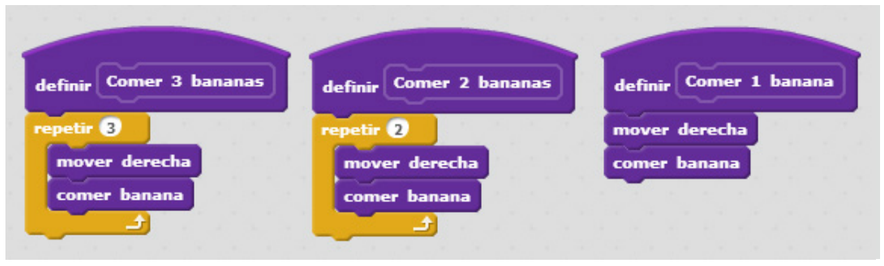


Ahora, es necesario un procedimiento que nos permita pasar a la siguiente fila de bananas:

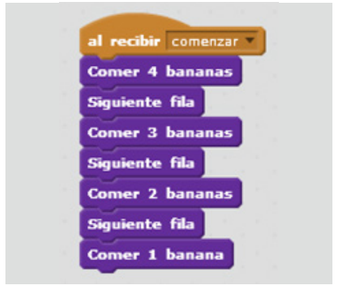
Con esta nueva instrucción, el programa queda así:

Con el siguiente tablero:

Para terminar de resolver el problema, es fácil observar que basta con crear otros procedimientos similares al primero para las demás filas:

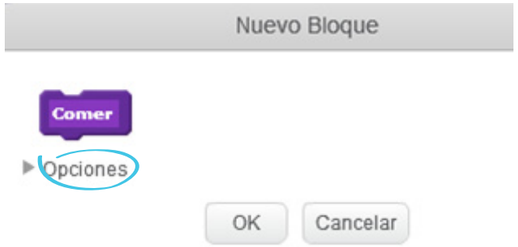


En resumen, queda el siguiente programa final:

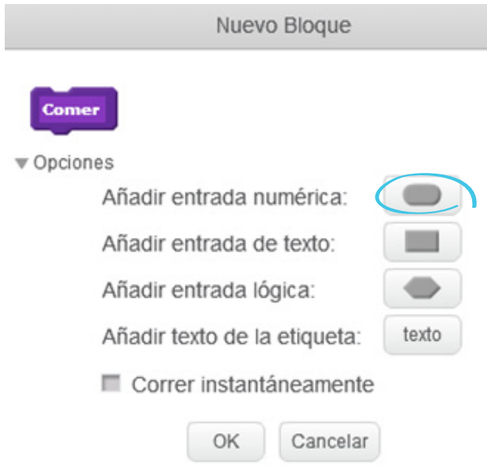


Esta solución no es la única posible, pero es más fácil de explicar y de leer que otras (a la computadora cualquier solución en la que el marciano coma todas las bananas le es indistinta); sin embargo, presenta una característica que la hace susceptible de ser mejorada: repite muchas veces algo *igual* o *muy similar* (comer bananas) que solo cambia en un *valor* (la cantidad de bananas). En lugar de crear cuatro procedimientos que solo varían en un número, lo que puede hacerse es indicar de alguna manera en un comando que este recibirá un dato y que ese dato va a decir cuántas bananas debe

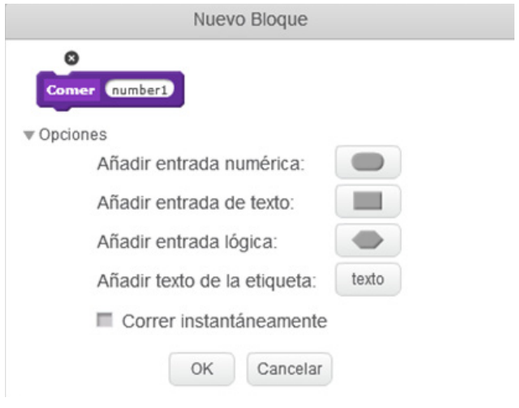
comer el extraterrestre. ¿Cómo se hace esto? Primero se debe establecer un parámetro. Para ello, se crea un nuevo bloque (com*er*, en este caso) y, dentro de la ventana del nuevo bloque, se hace clic en *Opciones*:



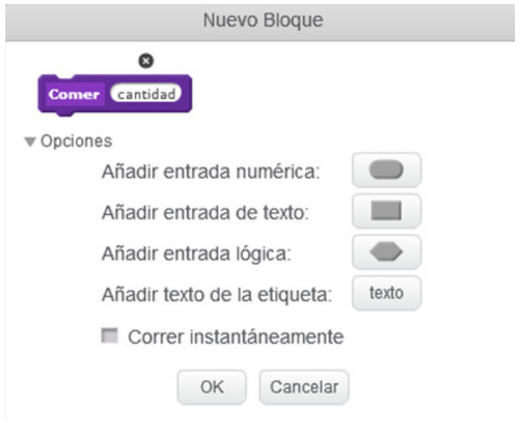
De las opciones que se despliegan, se selecciona la que dice *Añadir entrada numérica:*



Al presionar ese botón, se agregará en el comando un hueco en blanco con un nombre por defecto:



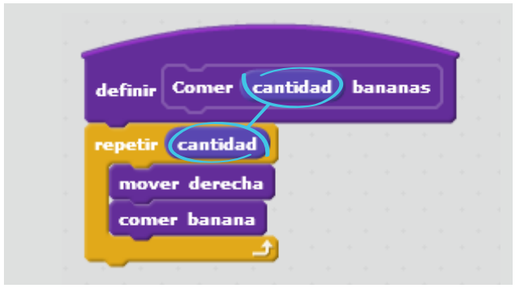
A continuación, se cambia el nombre que figura en el hueco por el que se considere más adecuado, teniendo en cuenta el dato –es decir, el parámetro– que se decide crear. En este caso, puede llamarse *cantidad*, para indicar que representará la cantidad de bananas que deben comerse en cada fila:



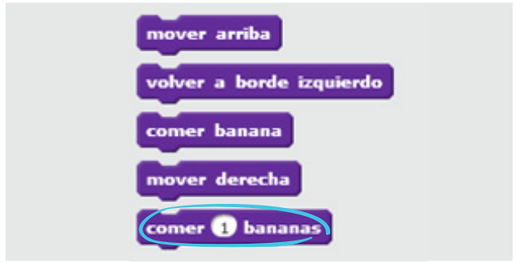
Para precisar más el parámetro, puede añadirse la información de que lo que se debe comer es una cantidad de bananas, mediante el botón *Añadir texto de la etiqueta*:

Al presionar *OK*, el nuevo bloque se verá así:

Este procedimiento genérico, que incluye un parámetro, se define de la siguiente manera:



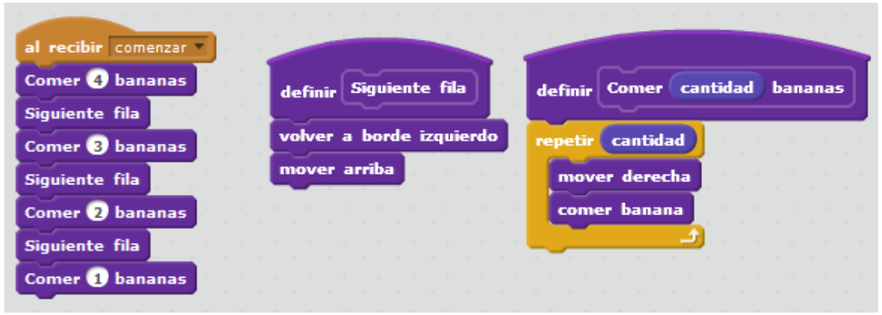
En la lista de bloques que pueden utilizarse, se visualiza así:



En la definición del procedimiento, puede observarse que, en el bloque *repetir*, el espacio que antes se completaba con un número ahora está ocupado por el bloque azul correspondiente al parámetro. Este bloque es una copia del que está arriba y se arrastró con el mouse desde allí.
Al definir el procedimiento con un parámetro, en lugar de indicar que la secuencia de avanzar y comer las bananas se repetirá un número concreto de veces, se especifica que esa cantidad será arbitraria y dependerá de lo que ingrese el usuario. Por ejemplo, si se escribe "4" en el bloque **comer <*cantidad*>bananas**, ese número reemplazará al parámetro en la definición del bloque en todos los lugares en donde aparezca:

Actividad

Tras la explicación acerca de cómo crear bloques que incluyan parámetros y cómo utilizar los bloques así definidos, se podrá dejar a los alumnos que reformulen la primera solución a la que arribaron empleando esta nueva herramienta. La solución es la siguiente:



Cierre

Como cierre, convendrá reforzar lo aprendido hasta aquí acerca de los parámetros. En particular, se sugiere destacar que mediante el uso de parámetros se puede definir un procedimiento de carácter **genérico**. Esto permite eliminar todos los procedimientos que eran instancias particulares de este, lo que resulta en una reducción del código usado en la definición de un programa.
Lo dicho acerca del uso de parámetros puede complementarse con otra idea, más intuitiva, que tal vez aporten los propios alumnos: la de que un parámetro es una *opción* entre varias, que se elige para un procedimiento determinado. Cada vez que se utilice ese procedimiento, se deberá elegir la opción que se considere adecuada; por ejemplo, si el extraterrestre debe comer una, dos tres o cuatro bananas. Algunos bloques predefinidos de Scratch poseen esta característica.

Secuencia didáctica 12

Dibujando figuras

En esta secuencia se trabajará con una actividad en Scratch ideada para crear figuras (básicamente, polígonos regulares); para ello, se deberán definir los programas necesarios y crear los parámetros adecuados.
A diferencia de lo que ocurría en otras oportunidades, en este caso no hay un objetivo predeterminado que se deba alcanzar (como recoger las bananas distribuidas en un tablero, encender las casillas de una hilera o descubrir a un sospechoso); en todo caso, las *metas* (que consisten en definir los programas necesarios para realizar determinados dibujos) son establecidas, por decirlo así, *sobre la marcha*. Esto le permitirá al docente una mayor libertad para omitir o modificar las actividades que considere adecuado de las que se proponen.

Objetivos

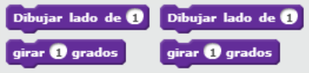
- Aplicar lo aprendido a lo largo del curso a nuevas situaciones.
- Definir y ejecutar programas que dibujen polígonos regulares.
- Parametrizar programas que dibujen polígonos regulares.

Desarrollo

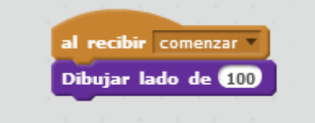
http://scratch.mit.edu/projects/42292872/#editor

Luego de presentar brevemente las características de la actividad, los alumnos ingresarán a la página web de Scratch correspondiente. El escenario que se presenta es bastante distinto del de otras actividades de Scratch: solo se ve un robot sobre un fondo blanco.

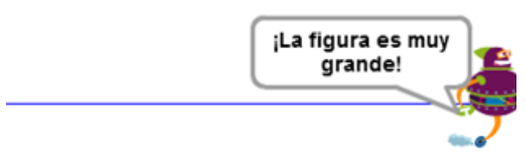
Las instrucciones primitivas son las siguientes:



Se comenzará utilizando los bloques *dibujar lado de...* y *girar... grados*. El primero se emplea para dibujar una línea. La pequeña ventana o el hueco que se encuentra a la derecha de *dibujar lado de...* permite indicar la longitud del lado a dibujar. Por ejemplo:



Para que pueda apreciarse la línea que traza el robot, la medida debe ser mayor que 25. Si la línea que se intenta trazar es mayor que 378, el robot se saldrá de la pantalla y avisará que no puede continuar:

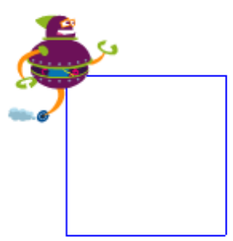


El bloque *girar... grados* permite indicar los grados que debe girar el robot. Si éste dibuja uno de los lados de una figura y luego gira, el próximo lado que dibujará estará inclinado tantos grados como haya girado.

Actividad

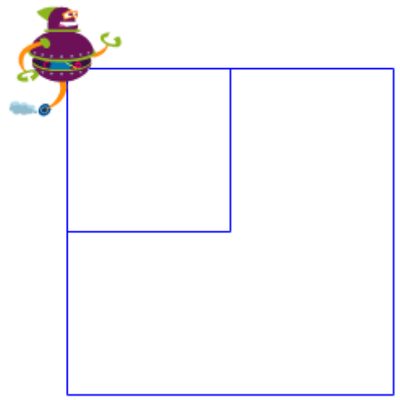
Tras esta exposición, se sugiere conversar con los alumnos acerca de cómo piensan que el robot podría dibujar un cuadrado utilizando estos dos comandos. Para orientar la discusión, tal vez convenga recordarles que los cuadrados poseen ángulos de 90 grados, es decir, ángulos rectos, y que constan de cuatro lados y cuatro ángulos. Ya sea que arriben o no a una solución satisfactoria, se les mostrará cómo dibujar un cuadrado con una longitud de 100 de lado:

Como parte de la explicación, se hará hincapié en que, si bien esta secuencia de comandos *funciona*, lo más apropiado es asignarle un nombre que guarde relación con la tarea a la que se refiere. De este modo, la solución del problema queda así:

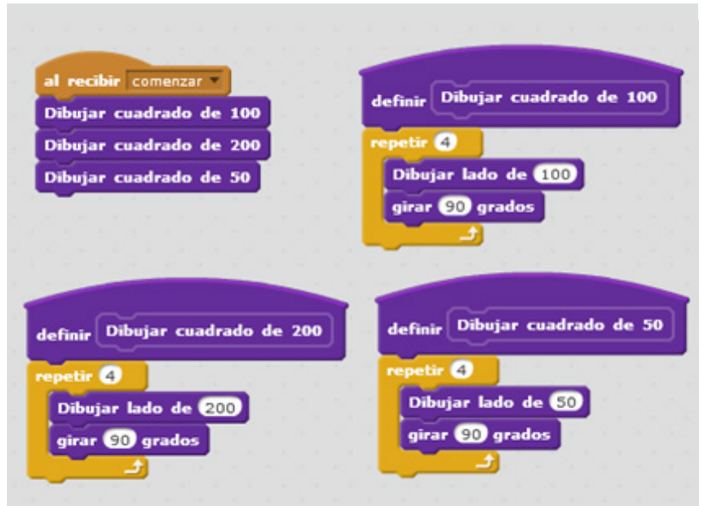


Actividad

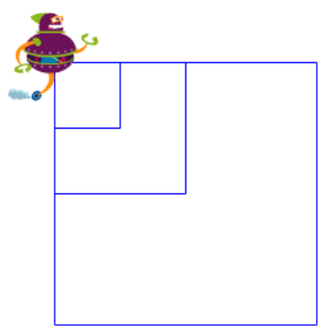
A partir de estas indicaciones, se les podrá pedir a los alumnos que creen un procedimiento para que el robot dibuje un cuadrado de 200 por lado, que se agregará al anterior. La solución, que da como resultado un cuadrado dentro de otro, es esta:



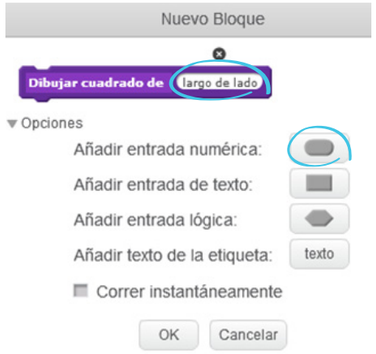
Es fácil darse cuenta de que el cuadrado más chico es el de 100 y el más grande, el de 200. A continuación, y siguiendo el mismo esquema, los alumnos podrán definir un procedimiento para un cuadrado de 50 de lado. Como resultado de la realización de esta consigna, el editor de Scratch debería quedar de esta manera:



El resultado de la ejecución del programa será este:



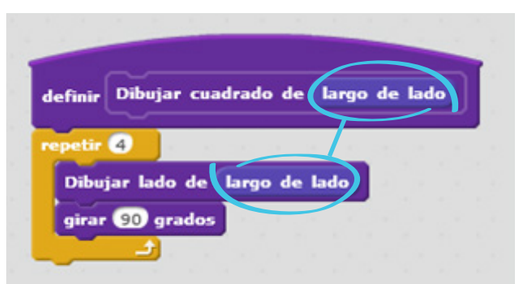
Luego de estas actividades introductorias, se abordará la formulación de parámetros. Al respecto, se les puede señalar a los alumnos que lo que el robot ha estado haciendo es una serie de acciones muy parecidas, que solo se diferencian entre sí por un valor. Así como el comando *girar...* permite indicarle al robot cuántos grados debe girar y el comando *mover...* cuál es la longitud de la línea que debe trazar, es posible crear un procedimiento mediante el cual el robot dibuje directamente toda la figura. Este procedimiento debe contener un parámetro. A dicho parámetro (que consiste en una entrada numérica) se lo llamará aquí *largo de lado*:



De este modo se creará un bloque como el siguiente:

Y uno así para usar, en la lista de bloques: 

El hueco que se encuentra en el lado derecho del bloque -al igual que en los bloques de instrucciones primitivas (y tal como se vio durante la secuencia anterior)- sirve para indicar el dato del parámetro que varía. En la definición del bloque, en vez de especificar la longitud del lado con un número determinado, se arrastrará el bloque que hace de parámetro:



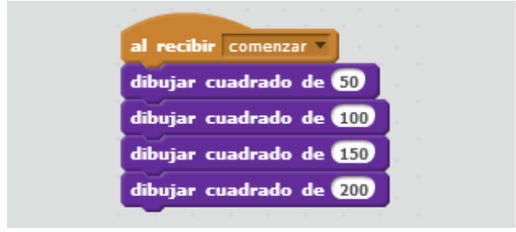
Actividad

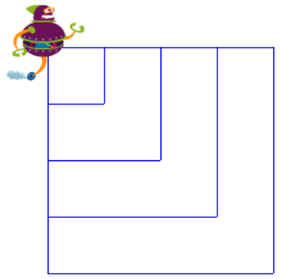
Los procedimientos anteriores ahora están de más, de modo que pueden eliminarse. En esta instancia, se les podrá indicar a los alumnos que usen el nuevo procedimiento parametrizado en la formulación del programa correspondiente para que el robot realice los siguientes dibujos:

a. Cuatro cuadrados de siguientes tamaños: 50, 100, 150 y 200.
b. Cinco cuadrados de 50 cada uno, uno al lado del otro.
c. Los mismos cuadrados anteriores, pero dispuestos en diagonal.

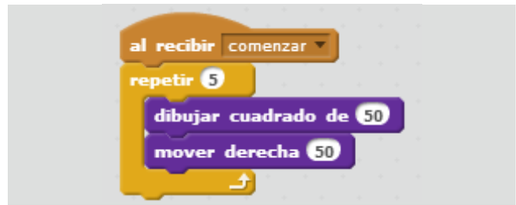
A continuación se muestran algunos resultados posibles. Los alumnos podrán encontrar otros, igualmente válidos, teniendo en cuenta que el objetivo es que logren dibujar las figuras parametrizadas solicitadas.

a.

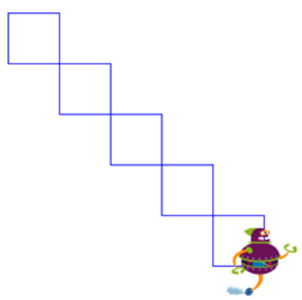




b.



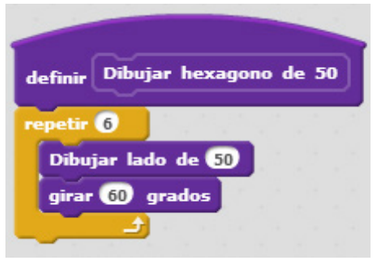
c.

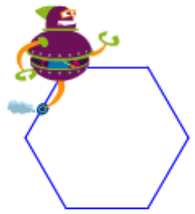


Además de estas figuras, se pueden proponer otras un poco más complejas, como un hexágono o un triángulo. Previamente, se sugiere repasar los siguientes aspectos de lo realizado al comienzo, al dibujar un cuadrado.
• La suma de la extensión de todos los lados es el perímetro del cuadrado. Si el cuadrado tiene 100 de lado, el perímetro es 400.
• Independientemente del tamaño del cuadrado, la suma de todos sus ángulos da 360 grados, que es el resultado de sumar cuatro ángulos de 90 grados.
• El procedimiento propuesto al principio para dibujar un cuadrado fue mejorado mediante el agregado de un parámetro.

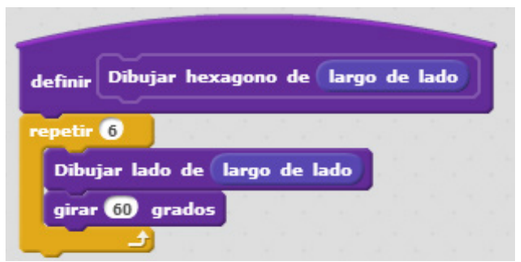
Actividad

Luego de este repaso, los alumnos podrán aplicar lo aprendido dibujando un hexágono (es decir, un polígono de seis lados). Para ello, deberán reproducir el siguiente código en el editor de Scratch y parametrizar el largo de cada lado del hexágono.





La solución es la siguiente:



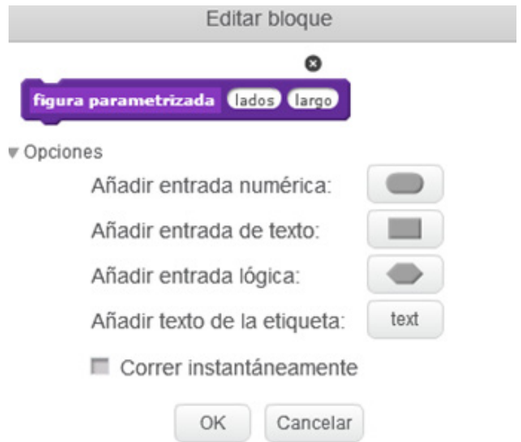
Actividad

Una vez resuelto el problema, convendrá volver sobre él y que los alumnos respondan entre todos las siguientes preguntas: ¿cuántos lados tiene esta figura?, ¿cómo se calcula el perímetro?, ¿cuántos grados gira el robot por cada lado?, ¿cuánto da la suma de todos sus giros? Se espera que los alumnos señalen que la figura consta de seis lados, que para calcular su perímetro simplemente se debe multiplicar 6 por 50, que el robot debe girar seis veces y que, tal como ocurre con todos los polígonos regulares, la suma de los ángulos (es decir, el total de giros que realiza el, robot) es igual a 360. De alguna manera, puede pensarse que el robot debe dar siempre *una vuelta completa* para trazar el polígono.
Teniendo en cuenta la última respuesta, se podrá llegar a la generalización de que, para saber cuántos grados debe girar el robot para dibujar un polígono regular, basta con dividir 360 por la cantidad de lados de la figura. A partir de ello, se les podrá plantear a

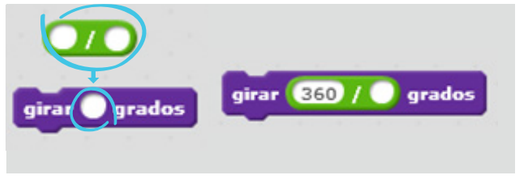
los alumnos que indiquen los grados que debería girar el robot para hacer una figura de ocho lados y otra de tres lados (es decir un triángulo), e indicarles que definan los procedimientos correspondientes a cada una de esas figuras, parametrizando el largo de cada lado, tal como se hizo en la actividad anterior. Las soluciones son las siguientes:

Tras estas actividades (y siempre que se considere adecuado al nivel general del curso), se puede avanzar un poco más en lo formalización de parámetros y plantear cómo podrían parametrizarse, además del largo de cada uno de los lados de una figura, la cantidad de giros que el robot debe hacer para dibujarla, teniendo en cuenta que estos arrojan siempre el mismo total (360 grados). Retomando el cálculo mencionado anteriormente, la cantidad de giros puede obtenerse mediante la siguiente fórmula:
360 / cantidad de lados = cantidad de giro (grados) por lado
Asimismo, sería interesante parametrizar la cantidad de lados que deben dibujarse. De este modo, el robot sería capaz de dibujar cualquier polígono regular a partir de un único procedimiento. Para crear este procedimiento, pueden seguirse estos pasos.

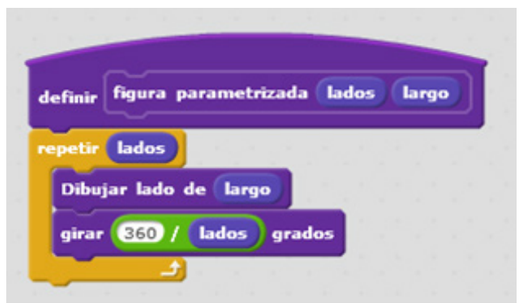
• En primer lugar, crear un bloque con dos entradas numéricas, una para el largo de cada lado y otra para la cantidad de lados:



• Luego, definir el bloque, utilizando para indicar la longitud de cada lado de la figura el comando *dibujar lado de...* y para la cantidad de giros el bloque *girar... grados* (dentro de la categoría *Más bloques*), junto con la expresión dividir de la categoría *Operadores*.



La definición del procedimiento quedará así:



Con este procedimiento, para que el robot dibuje una figura, bastará con arrastrar el bloque al editor y completar el primer hueco con la cantidad de lados de la figura (por ejemplo, 4 si se trata de un cuadrado) y el segundo, con la extensión que se quiere que tenga cada lado (50, 100 o 200):

Cierre

Como cierre de la secuencia se sugiere repasar brevemente lo trabajado en clase, enfocándose en cómo se utilizaron los parámetros en las actividades. Así, se podrá señalar que, en una primera instancia, se usó un parámetro (referido a la longitud del lado de una figura) que permitió simplificar el programa para dibujar una misma figura (un cuadrado) varias veces, en diversos tamaños y diversas posiciones; posteriormente, mediante otro parámetro (que añadía al valor anterior el de la cantidad de lados), fue posible ampliar el rango de figuras (polígonos regulares) que podían dibujarse con una secuencia de comandos simple. Eventualmente, se les podrá proponer a los alumnos que realicen nuevamente los tres ejercicios de la página 93, pero con al menos dos polígonos regulares (por ejemplo, un cuadrado y un pentágono).

Ejercitaciones

Se proponen a continuación cinco actividades de Scratch centradas en el uso de parámetros. Como en otras ocasiones, cada una de ellas plantea problemas de complejidad creciente.

La fiesta de Drácula

http://scratch.mit.edu/projects/42297138/#editor

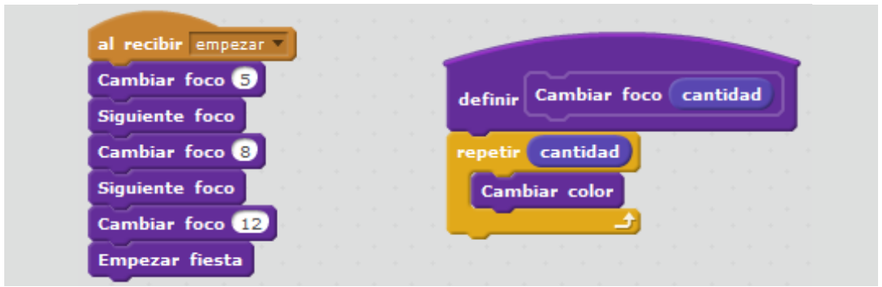
En esta actividad de Scratch se presenta el siguiente escenario (presionar el ícono para visualizarlo):

El objetivo es cambiar los colores de los tres focos de manera que cada foco quede con un color determinado para que pueda empezar la fiesta organizada por Drácula. Las primitivas son las siguientes:

Las instrucciones primitivas son las siguientes:

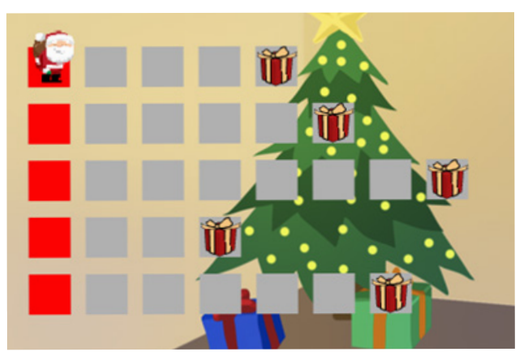
Para cambiar el color de cada foco hasta conseguir el color esperado, se debe ejecutar la primitiva *cambiar color* una cierta cantidad de veces: cinco veces en el caso del primer foco, ocho veces en el segundo y doce veces en el tercero. El resultado será el siguiente:

Luego de haber cambiado correctamente los focos, se puede ejecutar *empezar fiesta* para que Drácula y sus amigos comiencen a bailar. Si los focos no tienen el color correcto, la fiesta no empezará.
Los alumnos tendrán que definir un procedimiento con un parámetro que indique la cantidad de veces que se debe ejecutar la instrucción primitiva *cambiar color* y, luego, utilizar ese procedimiento para cambiar el color de cada foco para así resolver el problema. Una solución posible es la siguiente:

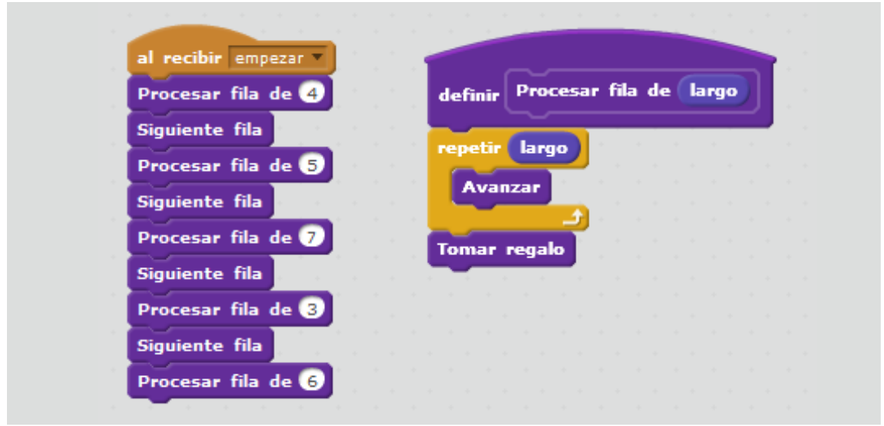


Salvando la Navidad **http://**scratch.mit.edu/projects/42297302/#editor

En esta actividad de Scratch, Papá Noel debe recoger el regalo que se encuentra al final de cada fila, cuya longitud permanece fija (es decir, no se trata de un escenario cambiante).



Los alumnos deberán resolver este problema mediante la definición de un procedimiento que le permita a Papá Noel recorrer cada fila y recoger el regalo que se encuentra en el extremo, sin utilizar una repetición condicional. Esto implica parametrizar ese procedimiento, estableciendo como parámetro la longitud o el largo de fila. El procedimiento así definido podrá luego ser utilizado indicando el largo específico de cada una de las filas. Conviene destacar que este se cuenta a partir de la primera casilla gris (no de la casilla roja). Una solución posible es la siguiente:

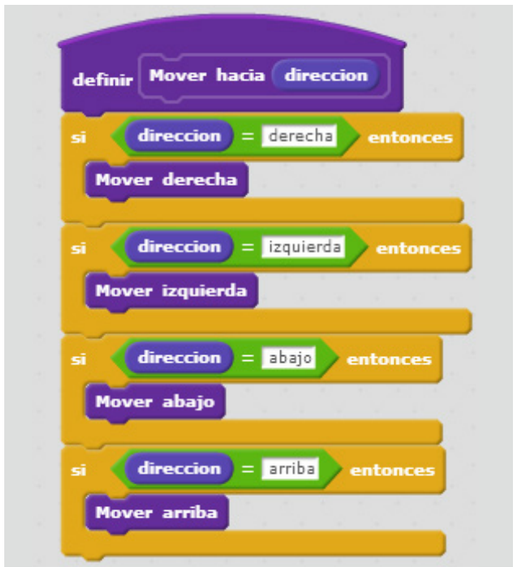


Prendiendo las compus (parametrizado) **http://**scratch.mit.edu/projects/42293830/#editor

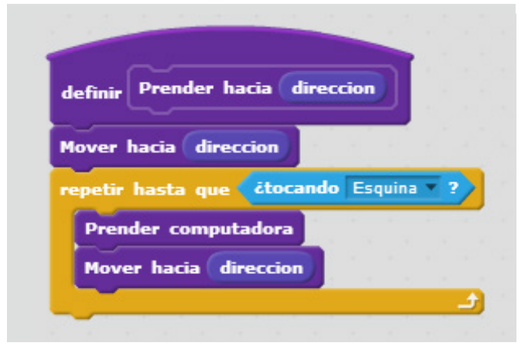
Se retoma aquí una actividad de Scratch (disponible en) ya trabajada anteriormente, al abordar el uso de repeticiones condicionales. El propósito ahora es aplicar en ella lo aprendido hasta el momento respecto de la parametrización. En primer lugar, se sugiere repasar el problema y su solución. Como se recordará, el escenario era un rectángulo de tamaño variable con computadoras que debían ser encendidas por un robot. La solución propuesta incluía definir cuatro procedimientos –uno por cada lado del rectángulo– en los que se indicaba al robot moverse en determinada dirección e ir prendiendo las computadoras hasta llegar al final de cada lado.
Una primera observación que puede hacerse a esta solución es que los procedimientos creados eran prácticamente iguales; solo diferían en la dirección en que debía moverse el robot. Vale la pena preguntarles a los alumnos cómo se podría resolver ahora el problema utilizando un parámetro. La respuesta consiste, primero, en crear un procedimiento que contenga como parámetro la dirección, de manera que, al ejecutarlo, el robot se mueva en una dirección determinada.

La definición de este procedimiento deberá incluir la referencia a las cuatro direcciones posibles utilizando para ello, como parte de una alternativa condicional, el bloque de equivalencia (<*valor*> = <*valor*>), en la categoría *Operadores*:

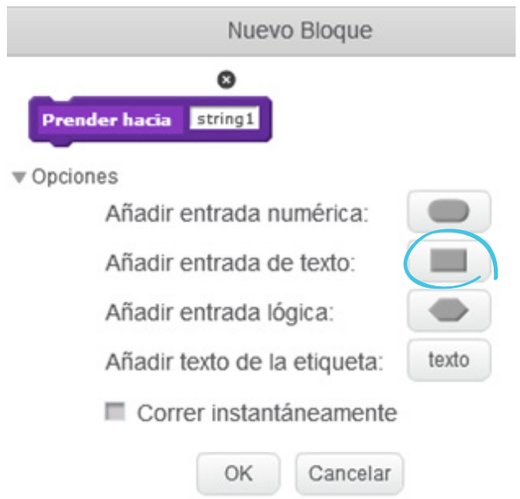
Este bloque indica que un valor es verdadero, si los valores ubicados a cada lado del '=' son idénticos, o falso, si no lo son. De este modo, la condición se cumplirá solo si ambos elementos son iguales. En el casillero de la izquierda, se insertará el bloque del parámetro dirección y en el de la derecha la indicación correspondiente a una de las cuatro posibles direcciones.

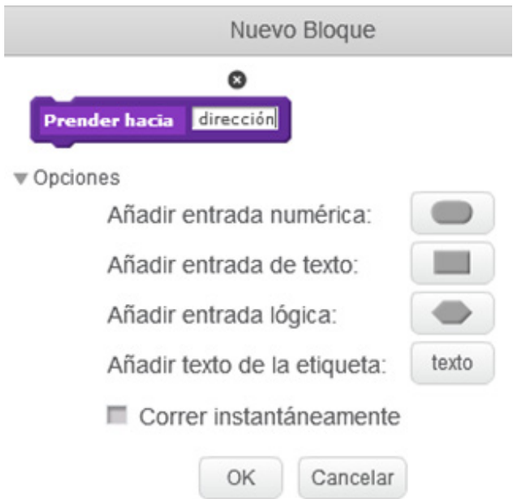


Este procedimiento permitirá, luego, reemplazar los cuatro procedimientos anteriores por uno solo:

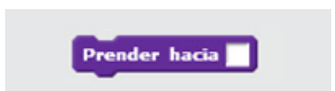


Es importante tener en cuenta que, en este caso, el parámetro no es un valor numérico; por eso, en la definición del procedimiento, se deberá seleccionar el botón *Añadir entrada de texto:*





De esta manera, para hacer el programa, se dispondrá de un bloque en el que se podrá ingresar la dirección que se desea (izquierda, derecha, abajo, arriba).



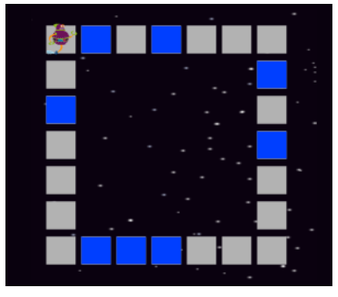
El programa que ejecutará el robot quedará así:

O bien, así:

Puesto que la resolución de este ejercicio presenta cierta complejidad, convendrá guiar a los alumnos, explicando cada uno de los pasos de la solución expuesta, sin que ello signifique dárselas en su totalidad, al menos antes de que hayan intentado ir resolviendo los inconvenientes con los que pudieron haberse encontrado.

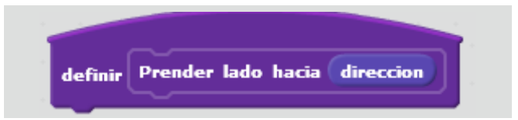
Lightbot cuadrado **http://**scratch.mit.edu/projects/42297692/#editor

En esta actividad de Scratch, como en otros casos similares, el objetivo es que el robot encienda las luces de las casillas azules.



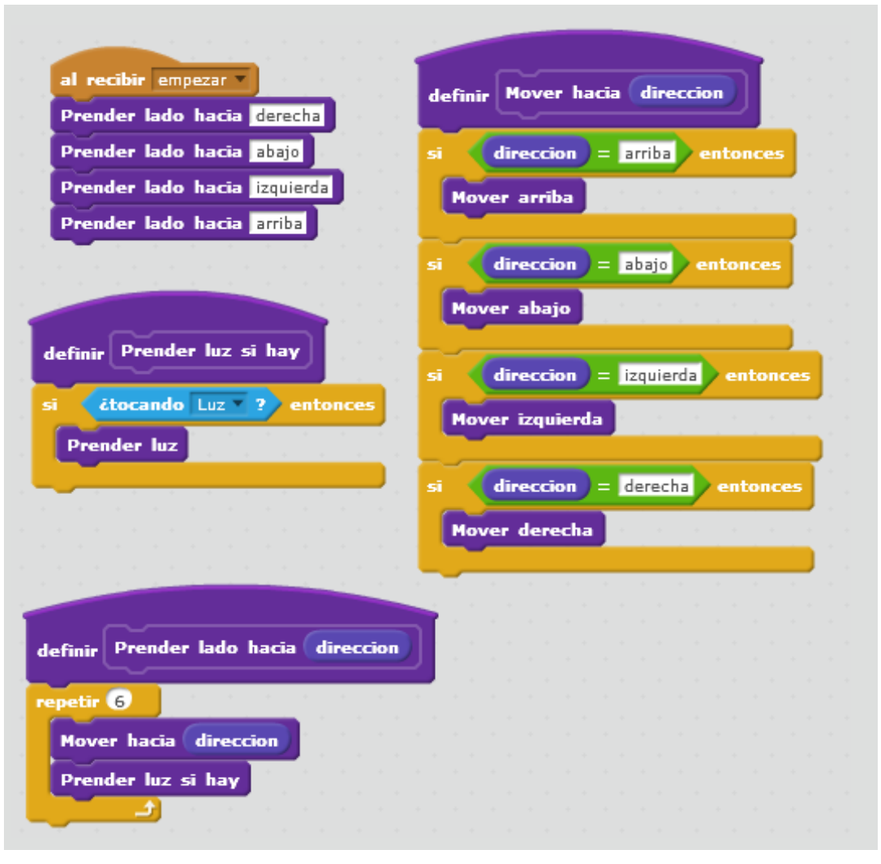
Hay algunas semejanzas con la ejercitación anterior y, también, con *Lightbot recargado*, trabajada en el apartado de alternativas condicionales. Pero, a diferencia del primero, el tamaño del rectángulo es siempre el mismo; lo que varía de una sesión a otra es la cantidad de casillas azules y su distribución. Únicamente las casillas de las esquinas nunca tienen luces. A diferencia de *Lightbot recargado*, aquí las casillas forman un rectángulo y no solo una fila. Teniendo en cuenta esto, las soluciones empleadas en esas dos ejercitaciones pueden servir de guía para resolver el problema que se plantea en esta oportunidad.
En primer lugar, será necesario definir la subtarea *Mover hacia...*, idéntica a la actividad anterior.

Luego, se deberá definir otra subtarea que permita procesar cualquier lado del cuadrado:



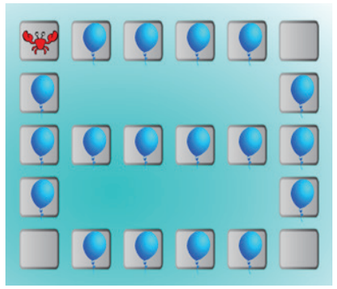
A continuación, se procederá a definir un tercer procedimiento para que el robot encienda las luces de las casillas solo en el caso de que estas sean azules. Esto significa que el procedimiento deberá contener una alternativa condicional:

Con estas indicaciones, se podrá dejar que los alumnos completen la definición de las subtareas mencionadas y que, luego, definan un programa utilizando solo la segunda subtarea. Una solución posible es la siguiente:



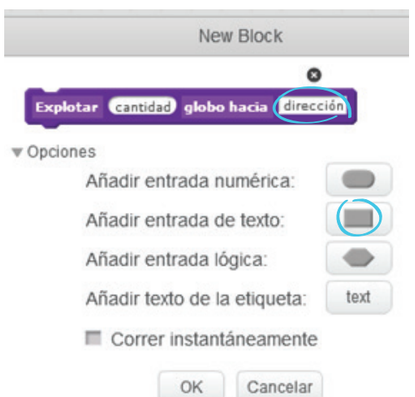
El cangrejo aguafiestas **http://**scratch.mit.edu/projects/42298948/#editor

En este caso se presenta el siguiente escenario fijo:



El cangrejo debe pinchar todos los globos. Para ello, se puede recurrir a una solución similar a la anterior, en la que se definirá una subtarea que le permita al autómata moverse en una dirección determinada, saber cuántos son los globos que debe pinchar en esa dirección y pincharlos efectivamente. Para crear el bloque de esta subtarea (a la que se la llamará aquí *explotar <cantidad> globos hacia <dirección>*), en primer lugar se elige, de las opciones de la ventana del nuevo bloque, la opción *Añadir entrada numérica*, tal como se ha hecho en otras ocasiones, y se completa la entrada con el nombre del primer parámetro:

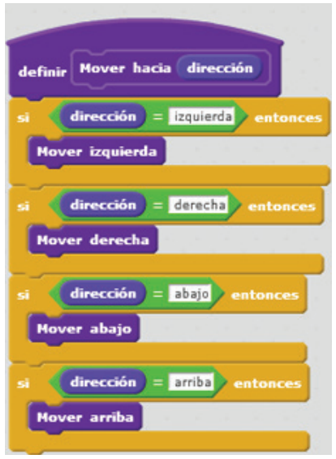
A continuación, se presiona *"Añadir texto de la etiqueta"* y se completa el nombre el bloque. Por último, se hace clic en *"Añadir entrada de texto"* y se completa el nombre del segundo parámetro:

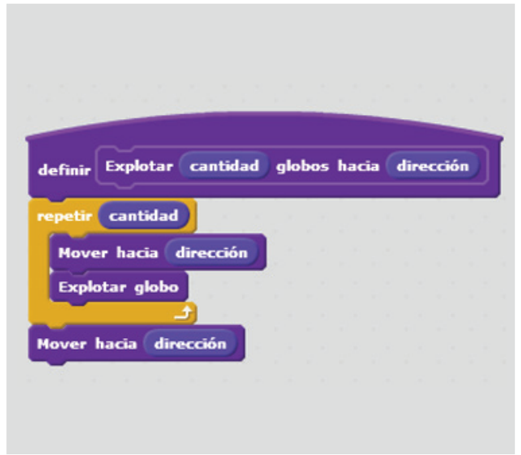


El bloque que se quiere definir quedará así:

En la solución que se propone aquí, se utilizará, además de esta subtarea, otra que le permitirá al personaje avanzar de a un casillero por vez en una dirección determinada:

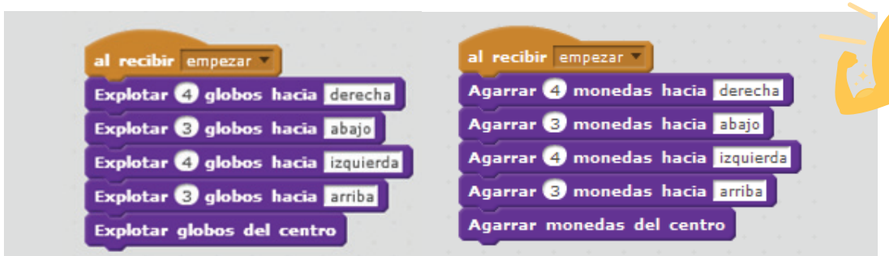
Tras estas indicaciones, los alumnos podrán completar la resolución del problema; esto es: definir los dos procedimientos mencionados, evaluar si es necesario añadir otra u otras subtareas adicionales y, por último, definir y ejecutar el programa que corresponda. Una solución consiste en definir primero la subtarea *Mover hacia* y luego *Explotar <cantidad> globos hacia <dirección>* de esta manera:





Con ello, se asegura que el personaje explote todos los globos monedas, excepto cuatro, que quedan en la fila del centro. Para que pueda pinchar estos globos, es necesario definir un procedimiento adicional:

Y con esto el programa que podemos definir es el siguiente:



INTERACTIVIDAD

Esta última parte está dedicada a la programación de juegos. Cuando se habla de juegos, es inevitable referirse a la noción de interactividad, ya que es lo que permite que los programas realicen acciones a partir de determinados comportamientos del usuario, tales como presionar el teclado, el mouse, el joystick o cualquier otro dispositivo a través del cual puedan ingresarse datos a la computadora.
De manera general, la propuesta consiste en guiar a los alumnos para que comprendan que cada juego posee una lógica a la que es necesario atenerse para elaborar el programa adecuado. Desde ya, la finalidad no es simplemente que se dediquen a jugar y ganar, sino que logren programar los juegos, utilizando las herramientas que ofrece Scratch.

Secuencia didáctica 13

Juegos y escenarios cambiantes

En esta primera secuencia, se trabajará con dos proyectos de Scratch que presentan escenarios cambiantes. Estos proyectos no contemplan la posibilidad de algún tipo de interactividad, pero permitirán introducir elementos de la lógica de los juegos que se programarán más adelante.

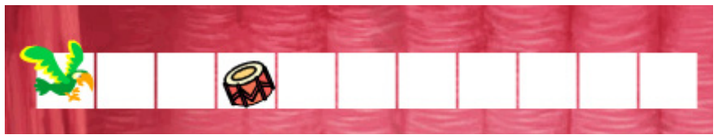
Objetivos

- Determinar todas las alternativas posibles en un escenario cambiante y establecer las relaciones de subordinación que puede haber entre diferentes acciones.
- Reconocer que todos los juegos están basados en una serie de alternativas, de las cuales depende la realización de ciertas acciones.

Desarrollo

http://scratch.mit.edu/projects/42298352/#editor

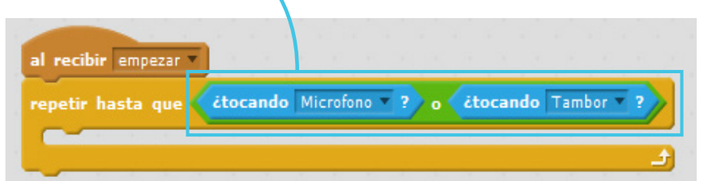
Se comenzará con la actividad de Scratch *La música del loro*. Allí se presentan dos escenarios cambiantes y un mismo problema: un loro debe recorrer una serie de casillas para acercarse a un micrófono o a un tambor y cantar o tocar el instrumento, según el caso. Tanto la aparición de uno u otro escenario como la ubicación del micrófono o del tambor en la fila de casillas no pueden preverse de antemano. Como en otras oportunidades, presionando varias veces el ícono de la banderita verde, podrán verse los dos escenarios y sus variaciones



Las instrucciones primitivas son las siguientes:

Se sugiere conversar con los alumnos acerca de cómo resolverían el problema. A fin de orientar la discusión, será útil plantear algunas posibles estrategias de solución. Así, se podrá comenzar observando que, puesto que no es posible saber a qué distancia se encuentra el objeto en cada instancia, el programa que ejecutará el loro debería contener el bloque *repetir hasta que...*, empleado cada vez que hay una repetición condicional.

En este caso, no hay una sola condición: no es posible indicar que el loro tocará *solo* un tambor o *solo* un micrófono, porque en el escenario puede aparecer cualquiera de estos objetos. Por lo tanto, lo que se precisa es indicar de alguna manera las dos condiciones. Esto se realiza mediante el operador o de la categoría de operadores.



Con esto, ya se puede hacer que el loro avance hasta el objeto, sea cual sea este:

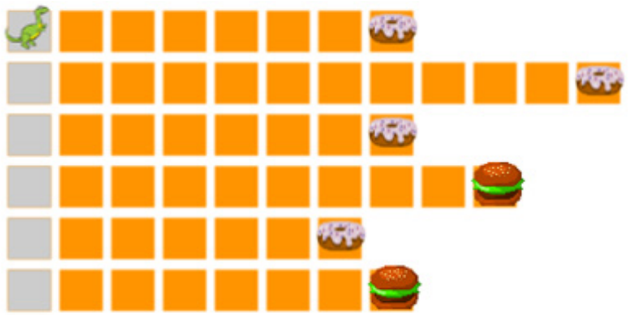
Al llegar a este punto, se les puede preguntar a los alumnos si consideran que así se ha resuelto el problema por completo. A partir de las intervenciones que realicen, deberá quedar claro que, ciertamente, falta formular las instrucciones necesarias para que el loro cante con el micrófono o toque el tambor. Para ello, debe saberse previamente ante cuál de los dos objetos se encuentra. Se trata de una alternativa condicional. Al respecto, resultará oportuno proponer un rápido repaso sobre este tema recordando lo resuelto en ejercicios anteriores, como el del laberinto corto (ver pág. 52). Conviene hacer hincapié en que, al ejecutarse el bloque *repetir hasta que*, el loro se detuvo porque se cumplía una de las dos condiciones (toparse con el micrófono o toparse con el tambor); de lo contrario, hubiese seguido avanzando. Luego, es necesario que preguntar si el loro se detuvo por haber tocado un tambor o un micrófono, porque no es posible saber por cuál de las dos condiciones se ha detenido. Esta técnica será utilizada más adelante para modelar la lógica de ciertos juegos.

Actividad

Tras estas precisiones, se puede dejar que los alumnos intenten resolver lo que resta del problema. La solución es la siguiente:

http://scratch.mit.edu/projects/42298684/#editor

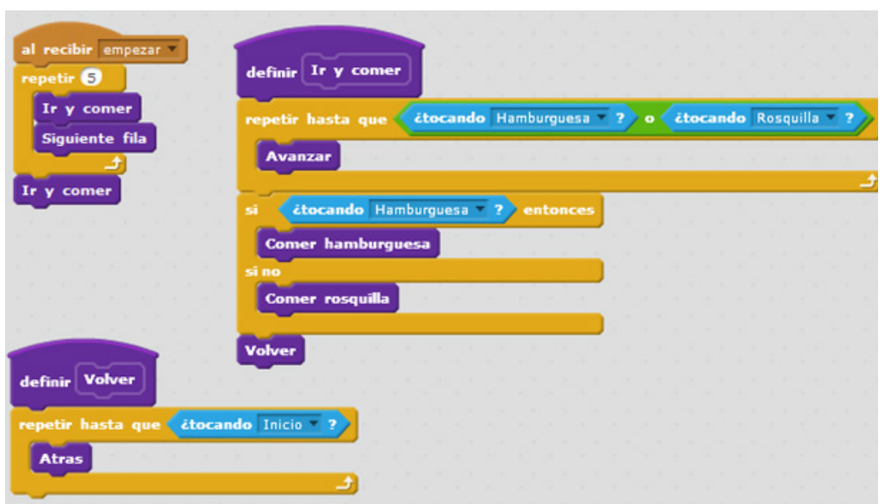
Antes de finalizar, se les podrá proponer a los alumnos que ingresen a la actividad de Scratch *Dino come chatarra* y, utilizando la estrategia vista anteriormente, definan un programa para que el dinosaurio que allí aparece coma los dos tipos de alimentos (hamburguesa y rosquilla) ubicados al final de cada fila.



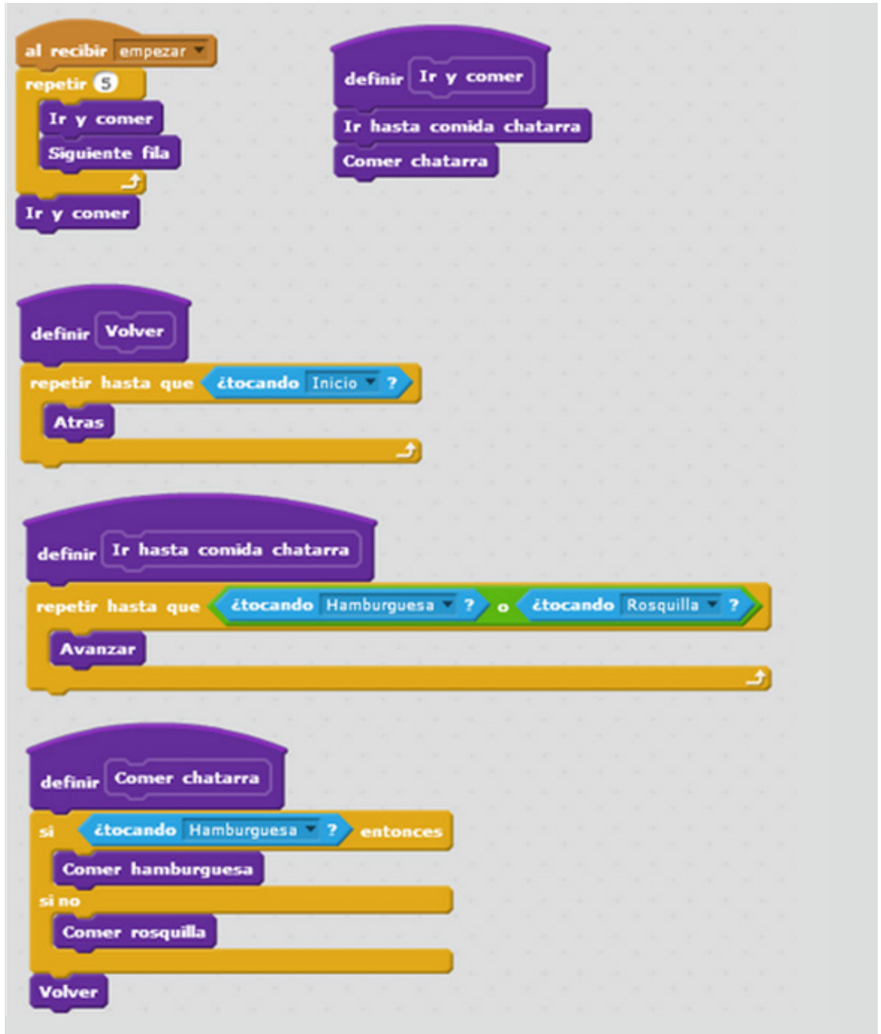
Si bien hay seis filas, el largo de cada una varía de una a otra instancia. Del mismo modo, el alimento que se encuentra al final puede alternar con el otro: en una misma fila puede haber una hamburguesa o una rosquilla.

Las instrucciones primitivas son las siguientes:

Una posible solución es la siguiente:



Otra solución consiste es subdividir el bloque ir y comer en más procedimientos, para precisar mejor las acciones que se deben realizar:



Cierre

Como cierre de la secuencia se puede volver sobre las actividades de Scratch trabajadas y –a manera de adelanto de lo que se verá en la próxima secuencia– solicitarles a los alumnos que propongan qué teclas del teclado de una computadora podrían vincularse con cada una de las acciones que deben realizar el loro y dinosaurio (por ejemplo, la tecla → se podría asociar con el desplazamiento hacia la derecha, la tecla ← con el movimiento en dirección izquierda, etcétera).

Secuencia didáctica 14

Programación de juegos

En esta secuencia se trabajará con una versión en Scratch de un juego, disponible en **www.el-juego-mas-dificil-del-mundo.com**. En el juego original, se debe mover un cuadrado rojo de una zona a otra, esquivando una serie de círculos azules que se desplazan. La versión que aquí se presenta es un poco distinta, ya que no es posible mover el cuadrado rojo. Esto se relaciona con lo que se pretende mediante la actividad: que los alumnos logren definir un programa que le permita a un usuario mover el cuadrado y, así, pueda jugar al juego.

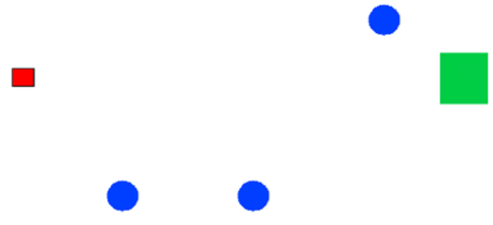
Objetivos

- Establecer la lógica específica de un juego y aplicar esa lógica para programar el juego.
- Definir las subtareas adecuadas a la interactividad que requiere el juego.

Desarrollo

http://scratch.mit.edu/projects/42297000/#editor

Los alumnos ingresarán a la versión del juego realizada en Scratch. Previamente, podrán explorar el juego original para comprender la lógica que lo rige. En la versión en Scratch, el cuadrado rojo (que representa a un personaje indeterminado y que, como se indicó anteriormente, se encuentra fijo) debe desplazarse hasta la salida (representada por el cuadrado verde), esquivando los círculos o bolas azules. Al presionar el ícono, podrá verse cómo estos se mueven.

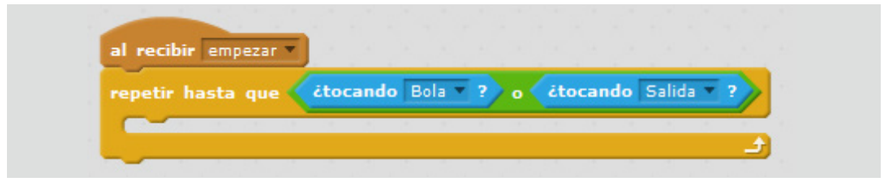


Las instrucciones primitivas son las siguientes:

Luego de explicar en qué consiste el desafío (esto es, establecer el programa apropiado para que el juego *funcione*) y presentar las instrucciones primitivas, se sugiere analizar la lógica del juego, de manera de guiar a los alumnos en la resolución del problema. En este sentido, se señalará que las alternativas que determinan el final del juego son dos:

- **Si el cuadrado rojo llega a la salida (el cuadrado verde), se gana el juego;**
- **Si el cuadrado rojo toca una bola azul, se pierde el juego.**

Mientras no se produzca ninguna de estas alternativas, el juego puede continuar. Esto significa, por un lado, que la técnica aplicada en otras oportunidades puede utilizarse también ahora:



Por otro lado, el hecho de que el juego pueda continuar hasta tanto no se pierda o se gane implica que el cuadrado rojo puede moverse hacia alguna de las cuatro direcciones posibles. Para indicar esta posibilidad, se utilizará una condición que no usada hasta ahora, llamada *¿tecla... presionada?*, de la categoría *Sensores*:

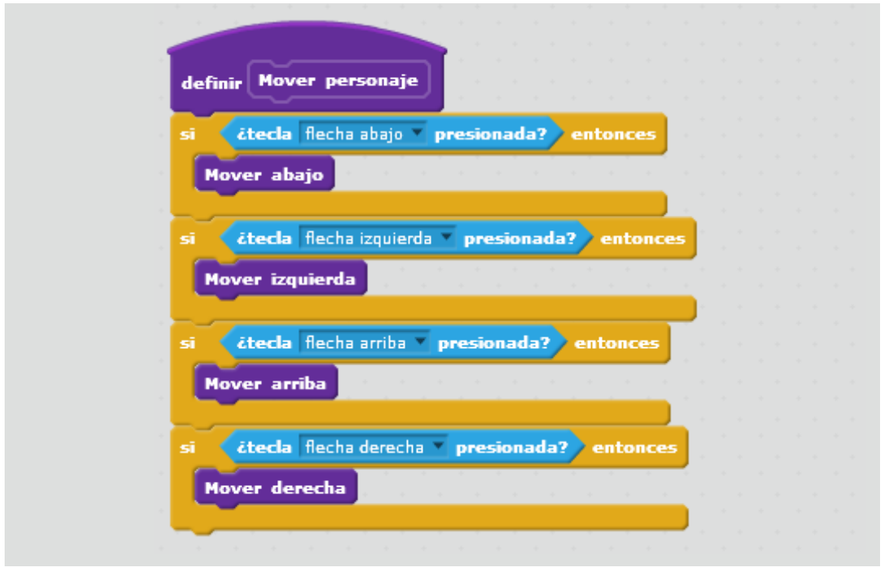
Esta condición permite preguntar si se ha presionado una tecla en particular (suponiendo, desde ya, que se emplea un teclado para jugar). Así, por ejemplo, puede relacionarse la acción de presionar la tecla → del teclado con la acción de moverse hacia la derecha:

Ubicando esta indicación dentro de la repetición se obtiene lo siguiente:

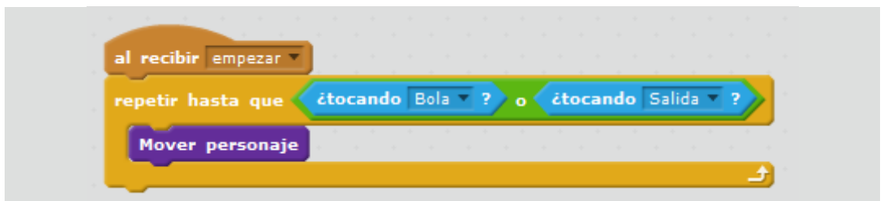
El funcionamiento de este programa (todavía incompleto) puede comprobarse presionando el ícono de la banderita verde y, luego, la tecla →. Al hacerlo, se verá que el cuadrado rojo avanza efectivamente hacia la derecha y se detiene cuando se choca con una bola azul o bien cuando llega al cuadrado verde. De este modo, se puede jugar un poco, pero de manera limitada. Para hacerlo adecuadamente, el programa debe incluir las indicaciones necesarias que hagan que el cuadrado rojo se mueva en alguna de las cuatro direcciones (y no solo hacia la derecha), de acuerdo con la tecla que se presione. Lo más práctico es hacerlo mediante un procedimiento.

Actividad

En esta instancia, se podrá dejar a los alumnos que definan dicho procedimiento (llamado aquí *mover personaje*). La solución es la siguiente:



Utilizando este procedimiento, se puede establecer un programa que permita jugar un poco más que antes, ya que se puede mover el cuadrado rojo en más direcciones:



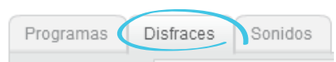
Cuando el cuadrado rojo toca un círculo azul o llega a la salida, termina la repetición y, por lo tanto, también el programa, ya que no hay más comandos luego de ella. El programa puede completarse con la indicación para que se explicite si se ganó o perdió el juego, de manera similar a lo que ocurría en las actividades de Scratch anteriores. Para ello, se agrega a la secuencia una alternativa condicional, como la siguiente:

El programa final queda así:

Cierre

Como cierre de la secuencia, se les puede proponer a los alumnos que cambien el aspecto de los objetos del juego, de modo que comprendan que un juego puede transformarse en otro con solo cambiar su presentación, sin que ello implique una modificación de las reglas. Por ejemplo, se puede hacer que el cuadrado rojo sea una bruja, las bolas azules murciélagos y el cuadrado verde una varita que hay que recuperar.

Para cambiar los objetos del juego, primero hay que clic en la pestaña *Disfraces:*



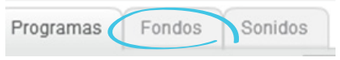
Luego, se selecciona, en el sector *Objetos* (ubicado en el lado inferior izquierdo de la pantalla), el objeto que se desea cambiar:

A continuación, se hace clic en el ícono que se encuentra debajo de la categoría *Disfraz nuevo*:

En la pantalla, aparecerá una serie íconos correspondientes a distintas clases de objetos. Se hace clic en el que se desea y, con ello, cambia el *disfraz*.

También puede elegirse un objeto que se encuentre guardado en un archivo de la computadora o, incluso, uno proveniente de una cámara fotográfica. Para ello, hay que hacer clic en o , respectivamente.
Además de los objetos, se puede cambiar el fondo del escenario. En este caso, se selecciona el recuadro ubicado a la izquierda de los objetos:

En lugar de la pestaña *Disfraces*, aparecerá la pestaña *Fondos*:



Se hace clic en esa pestaña y, luego, en el ícono que figura debajo de *Fondo nuevo*:

Finalmente, se selecciona el fondo que se desea de la lista.

Al igual que los objetos, haciendo clic en los íconos o es posible cargar un fondo desde un archivo o una cámara fotográfica.

Ejercitaciones

Las tres actividades de Scratch que se proponen a continuación consisten en juegos. El primero (el único en que no se trabaja la interactividad) plantea un desafío que se espera que los alumnos puedan resolver por sí mismos; los dos restantes deben ser abordados con mayor detenimiento. Se sugiere que, como cierre de cada una de estas actividades, los alumnos reemplacen los elementos del escenario (los personajes y el fondo) por otros que ellos elijan. El desafío será que el cambio no *altere* la coherencia del juego (por ejemplo, en ciertos casos, cambiar un personaje "animado" –como un ser humano o un animal– por un objeto inanimado –una piedra o un barco– afecta el sentido del juego).

En búsqueda de la llave **http://**scratch.mit.edu/projects/42298070/#editor

En este juego, el objetivo es que el buzo agarre la llave evitando al tiburón.

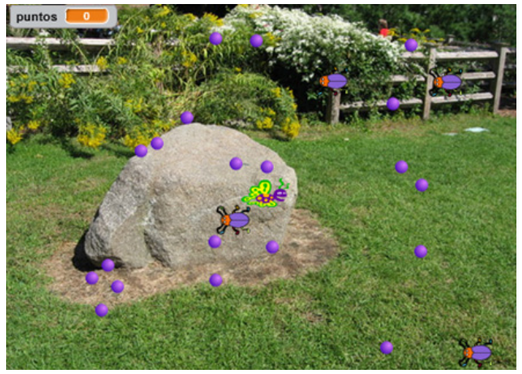
La lógica del juego es simple:

• Se gana si se logra que el buzo agarre la llave;
• Se pierde si el tiburón toca al buzo;

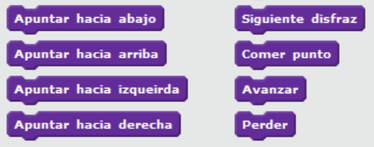
Los alumnos podrán programar este juego utilizando las técnicas vistas en la actividad anterior. Una posible solución es la siguiente:

El jardín abichado **http://**scratch.mit.edu/projects/42298768/#editor

En esta actividad de Scratch, se programará un juego en el que una mariposa debe comer todos los puntos violetas que se encuentran distribuidos en la pantalla, evitando tocar o ser tocado por cuatro escarabajos que se mueven de un lado a otro. Por cada punto comido, se gana un punto y se pierde otro tanto al tocar o ser tocado por un escarabajo.



Las instrucciones primitivas son las siguientes:

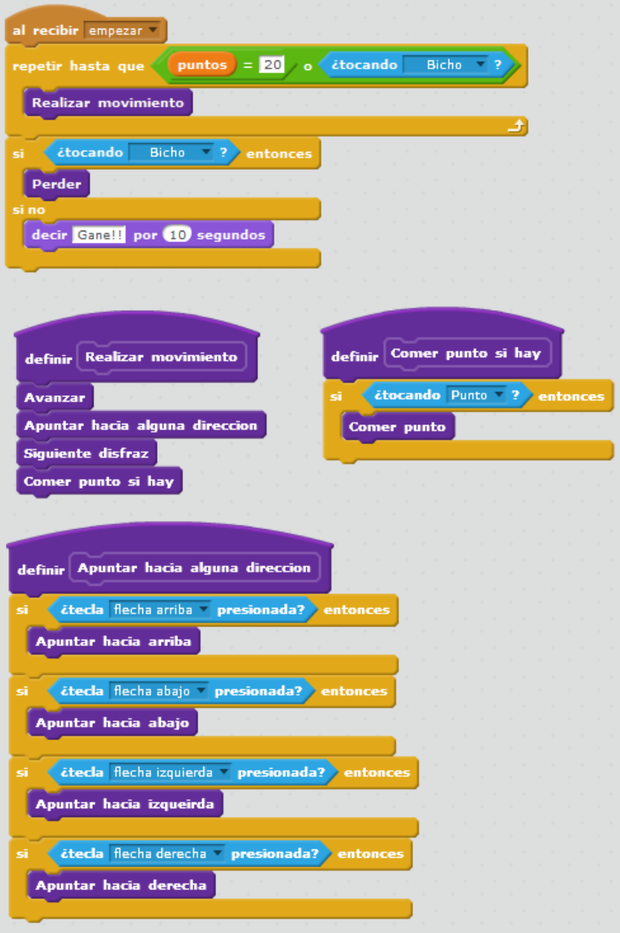


Los alumnos deberán definir un programa acorde a la lógica del juego. Para ello, deberán tener en cuenta las siguientes indicaciones:

• La dirección en que avanza la mariposa se puede cambiar mediante las teclas ← ↑ ↓ → (siempre y cuando, desde ya, previamente se hayan indicado en el programa estas acciones, utilizando para ello los cuatro primeros bloques).
• Mientras la mariposa se desplaza, la posición de las alas debe cambiar, de manera que representen el movimiento del insecto; estos cambios se indican mediante el bloque *siguiente disfraz*.
• De acuerdo con lo anterior, las acciones de la mariposa forman la siguiente secuencia:
1°) avanzar,
2°) apuntar hacia la dirección que indique el jugador con el teclado,
3°) pasar al siguiente disfraz,
4°) comer punto si la mariposa choca con uno.
Esta secuencia será útil para definir un procedimiento que sintetice los movimientos de la mariposa.
• El juego se gana cuando la mariposa come veinte puntos, que es la cantidad total de puntos que hay en la pantalla. Esta circunstancia tiene que indicarse en el programa; para ello, se cuenta con el bloque puntos en la categoría *Datos* y el operador <*valor*> = <*valor*> en la categoría Operadores.

• Al ganar o perder el juego, la repetición principal se detendrá. Entonces, tal como se hizo en el juego del cuadrado rojo y las bolas azules, se deberá indicar qué pasa en cada caso. Si se pierde, basta con indicarlo usando la primitiva correspondiente; pero si se gana, se deberá definir un procedimiento que haga que la mariposa declare explícitamente que ha ganado. Para ello, se puede utilizar alguno de los cuatro primeros bloques que se encuentran en la categoría *Apariencia*:

Una solución posible es la siguiente:

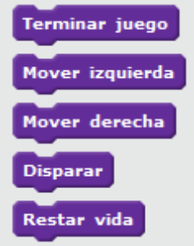


La defensa de la Tierra

http://scratch.mit.edu/projects/42297970/#editor

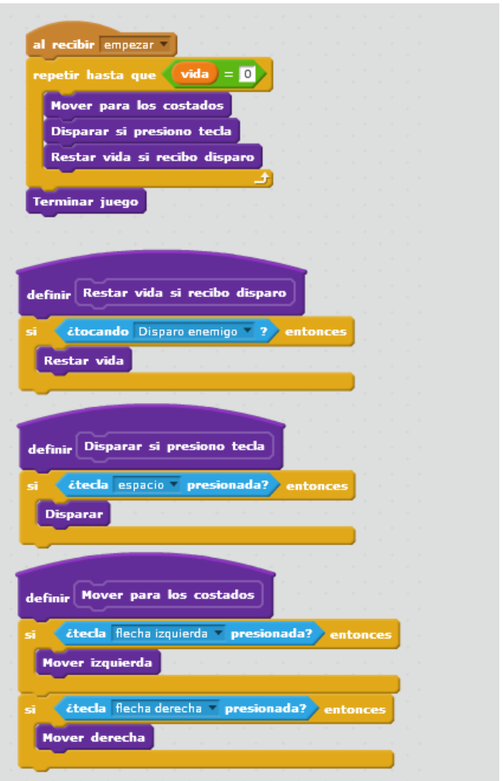
En este juego, el jugador debe maniobrar un tanque que intenta defenderse del ataque de extraterrestres. El juego consiste en destruir todas las naves enemigas posibles.

Las instrucciones primitivas son las siguientes:

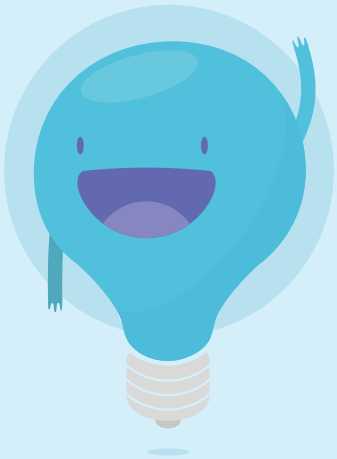


La propuesta es definir el programa correspondiente, de acuerdo con las siguientes reglas:

• Mientras el jugador no haya perdido, puede mover el tanque y disparar.
• Cada vez que se recibe un disparo, se debe restar una vida.
• Solo se pueden recibir como máximo cinco disparos. El juego se pierde cuando la cantidad de vida sea igual a 0. Para establecer esto, se puede utilizar el bloque *vida*, dentro de la categoría *Datos*. Además de estas precisiones, se pueden mencionar otras indicaciones adicionales, a manera de ayuda.
• El tanque solo se mueve hacia los costados. Por las características del vehículo, es imposible que vuele (esto es, que se desplace hacia arriba o hacia abajo).
• Se debe tener en cuenta que, para que el jugador pueda hacer que el tanque dispare tiros, es necesario asociar una tecla a esa acción. Así como el desplazamiento del vehículo está vinculado al uso de las teclas [←] [→], cada disparo del tanque puede asociarse a la barra espaciadora del teclado; de esta manera, cuando el jugador presione la barra, la nave lanzará disparos. El uso de esta tecla es una de las opciones que se incluyen en el sensor *¿tecla... presionada?*, con el nombre de *espacio*.
Una posible solución es la siguiente:



</05>



<Program.AR/>

Actividades para aprender a Program.AR

Segundo ciclo de la educación primaria
y primero de la secundaria

Versión 2.0

9 789872 741617

Material de distribución gratuita

fundación SADOSKY Investigación y Desarrollo en TIC
educar
CONECTAR IGUALDAD
Presidencia de la Nación
Ministerio de Ciencia, Tecnología e Innovación Productiva
Presidencia de la Nación
Ministerio de Educación
ANSES