

Cuaderno de docentes de primaria 2014

Versión 1.5



Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

Índice

[Consideraciones generales](#)

[Propósitos del curso](#)

[Contenidos](#)

[Estrategias didácticas](#)

[Estrategias de evaluación](#)

[Herramientas](#)

[Sobre Scratch](#)

[Sobre Lightbot](#)

[Autómatas, comandos y procedimientos](#)

[Las computadoras hacen todo al pie de la letra](#)

[El gato en la calle](#)

[Lightbot](#)

[No me canso de saltar](#)

[Programar en papel cuadriculado](#)

[El marciano en el desierto](#)

[Lightbot en Scratch](#)

[El recolector de estrellas](#)

[María, la come sandias](#)

[Alimentando a los peces](#)

[Instalando juegos](#)

[La gran aventura del mar encantado](#)

[Reparando la nave](#)

[Alternativas condicionales](#)

[Salida del aula cambiante](#)

[La elección del mono](#)

[Laberinto corto](#)

[Tres naranjas](#)

[Lightbot recargado](#)

[Laberinto largo](#)

[Repetición condicional](#)

[Super Lightbot 1](#)

[Super Lightbot 2](#)

[Laberinto con queso](#)

[El Detective Chaparro](#)

[Fútbol para robots](#)

[Prendiendo las compus](#)

[El mono que sabe contar](#)

[El mono cuenta de nuevo](#)

[El superviaje](#)

[Parametrización de soluciones](#)

[Canciones y estribillos](#)

[El planeta de Nano](#)

[Dibujando figuras](#)

[Más figuras](#)

[Figuras regulares \(opcional\)](#)

[La fiesta de Drácula](#)

[Salvando la navidad](#)

[Prendiendo las compus parametrizado](#)

[Lightbot cuadrado](#)

[Mario bros](#)

[Interactividad](#)

[La música del loro](#)

[Homero Simpson](#)

[El juego más difícil del mundo](#)

[En búsqueda de la llave](#)

[Pacman](#)

[La defensa de la Tierra](#)

Consideraciones generales

Propósitos del curso

- Presentar la idea de programa y demostrar que con estos podemos representar ideas y resolver problemas.
- Entender que las computadoras sirven para ejecutar programas y realizan lo que el programa indique.
- Incentivar a los alumnos a que se animen a ser creadores de programas y no sólo usuarios de aplicaciones hechas por terceros.
- Ejecutar programas diseñados por los propios alumnos.
- Detectar y corregir errores de los programas propios.
- 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 a 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 sobre estas, incluyendo la selección de operaciones primitivas del sistema de cómputo elemental.
- Metodología para la corrección de errores del programa analizando la diferencia entre lo que se espera del programa y lo que éste efectivamente hace.

Estrategias didácticas

- Actividades individuales y de a pares, con y sin computadoras
- Exposiciones de parte del profesor (muy cortas)
- Socialización en clase de las soluciones a los ejercicios

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

Estrategias de evaluación

- Evaluación 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 que facilita a los usuarios crear sus propias historias interactivas, animaciones, juegos, música y arte; además, le permite compartir con otros sus creaciones en la web. Usaremos este entorno de una manera distinta. Proveeremos un entorno de trabajo y los alumnos resolverán actividades pensadas dentro del editor de Scratch. En todas estas actividades se tendrá que resolver un problema específico del entorno creado por nosotros, y la solución se diseña programando.

Socialización en clase de las soluciones a los ejercicios

Haciendo clic sobre el título de la actividad marcado en azul, se puede acceder a una versión online de la misma.

Sobre Lightbot

Lightbot es un juego de ingenio en el que el usuario debe programar para pasar de nivel. La dificultad va aumentando a medida que pasamos de nivel. Utilizaremos esta herramienta para que los alumnos den sus primeros pasos en el desarrollo de programas que resuelven problemas. Deberán secuenciar acciones y abstraer subtareas sobre tareas que se repiten, dado que el espacio para programar es limitado y debemos ahorrar instrucciones.

Autómatas, comandos y procedimientos

Las computadoras hacen todo al pie de la letra

Presentación

En esta actividad no utilizaremos computadoras, sino que haremos una actividad entre todos para comprender mejor qué tipo de tareas realiza la computadora y cómo debemos indicarlas.

Desarrollo

El docente hará de autómata y los alumnos darán instrucciones para cumplir un objetivo.

El objetivo puede ser que el docente se ubique en un rincón del aula y los alumnos lo guíen hasta salir del aula, con la puerta cerrada.

Las primitivas en este caso son:

- Dar un paso hacia delante
- Girar hacia la derecha
- Girar hacia la izquierda
- Abrir puerta

Seguir la siguiente secuencia:

- Pensar en comandos que podríamos dar a compañeros (levantar brazo derecho, pestañear, mirar a un determinado compañero, levantarse de la silla, etc.).
- Proponer que un determinado alumno ejecute una lista de comandos como la siguiente:
 - Levantarse de la silla
 - Levantar brazo derecho
 - Cerrar ojos
 - Abrir boca
 - Sentarse en la silla
- Ver como estas instrucciones son bien específicas.
- Ver cómo combinando de distinta manera los efectos de estos comandos nos conducen a distintos resultados (y puede ser que distintas combinaciones den los mismos resultados también). Mostrar también que en ejercicios anteriores el orden a veces importaba y a veces no.
- Pedir a los alumnos que ordenen la secuencia para provocar otros resultados.

- Pensar en casos sin sentido:
 - ¿Y si no hay una silla?
 - ¿Qué pasa si me dicen que levante el brazo derecho y ya lo tengo levantado?
 - ¿Qué pasa si me dicen que levante el brazo derecho y tengo levantado el brazo izquierdo? ¿Debo bajarlo antes de levantar el derecho? (Suponer que levantar el brazo derecho no especifica nada sobre el izquierdo)

Actividad: Ahora el docente hará nuevamente de computadora. Pedir a un alumno que lo guíe para que vaya hasta el pizarrón y escriba su nombre (el del alumno). La idea es escribir el nombre letra por letra, pero que esto no se sepa de entrada. Si el alumno dicta el nombre todo junto, decir “no entiendo”. Si el docente no está en el pizarrón decir “no puedo escribir en el aire”. Al poco tiempo indicar que sólo se sabe escribir letra por letra.

Cierre

Es importante que los alumnos aprendan a poner nombre a distintas tareas para luego combinarlas y diseñar soluciones. En este sentido queremos que los alumnos describan paso a paso cómo hacen para empezar a trabajar en Scratch (prenden la PC, abren el programa, abren el proyecto, cambian el idioma si no está en el español, etc.). Así mismo, estaría bueno que distingan qué cosas realizan a diario que involucran una serie de pasos, y qué nombre poseen esas actividades (cepillarse los dientes se compone de varias subtareas, cocinar una torta, ir a la escuela, etc.). En relación a todo esto proponer actividades que involucren asignar nombres a tareas grandes, partirlas en tareas más pequeñas que también tienen nombre, y agruparlas para formar otras tareas más grandes todavía (como “vivir un día entero”).

El gato en la calle

Objetivos

A través de estas actividades queremos que los alumnos comprendan:

- Que un comando es una acción que genera un efecto (reproducir un sonido, pintar a una imagen, mover un objeto, etc.).
- Que las secuencias de comandos permiten encadenar (o componer) comandos de forma ordenada.
- Que los programas están formados por comandos y secuencias de comandos.
- Que ejecutar un programa es seguir una secuencia ordenada de pasos y llevar a cabo el efecto de cada comando.
- Que los procedimientos nuevos son comandos en los que le explicamos a la computadora cómo realizar una tarea nueva.

Presentación

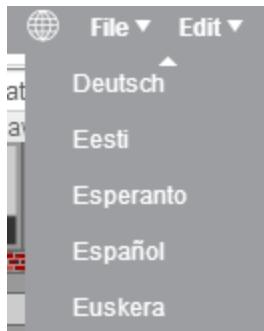
Diremos que vamos a usar un programa que nos permitirá programar autómatas (un objeto que sigue nuestras órdenes al pie de la letra).

Desarrollo

Haremos la actividad [El gato en la calle.](#)



Con esta actividad se presenta la interfaz de Scratch. Si la interfaz se encuentra en inglés, es posible pasarla a español haciendo click en el botón y seleccionando el idioma que queramos:



Empezamos contando que los programas en Scratch consisten en asignar una secuencia de bloques debajo del bloque de color marrón que dice “al recibir empezar”:



Cada bloque representa una acción a realizar, y cuando uno los encarta verticalmente, forman una **secuencia de comandos**, es decir, una tarea que se hace en un orden establecido, que se ejecuta desde el primer bloque hasta el último (de arriba hacia abajo).

Para ejecutar un programa hacemos clic en la banderita verde.



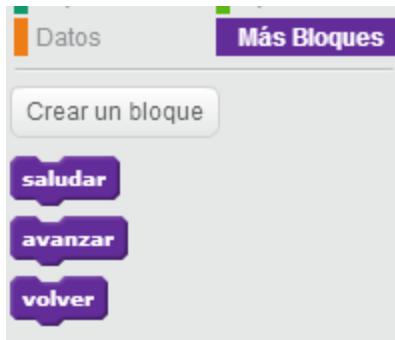
(el botón rojo sirve para detener el programa en cualquier momento)

Eso ejecutará toda la secuencia que hayamos asignado al bloque anterior:



Así, el gato realiza la secuencia “avanzar, saludar, volver”.

Actividad: Luego de explicar y mostrar cómo funciona, se pide realizar otra secuencia: avanzar, avanzar, saludar, volver, saludar (por ejemplo). Sólo deben usar los bloques de la categoría “más bloques”:



De esta forma, podemos ver que **un programa es una descripción que le damos a la computadora para que realice lo que le indicamos**.

Observar también que distintas secuencias producen resultados distintos:



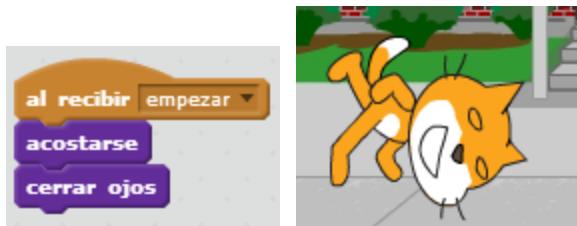
Ahora vamos a armar una secuencia que represente la tarea de “dormirse”, que consistirá en primero acostarse y después cerrar los ojos.

Para ello tenemos algunos bloques más:



Actividad: Pedir que realicen una nueva secuencia utilizando estos últimos bloques.

La secuencia de “dormirse” será:



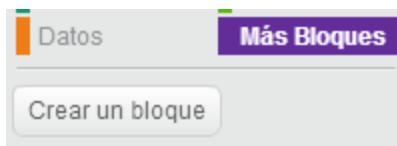
Hasta ahora utilizamos bloques que ya vienen con el problema, llamados **primitivas**, pero sería interesante crear nuestros propios bloques. A los nuevos bloques los llamaremos **procedimientos**. Así, la definición de nuevos procedimientos son nuevas acciones que le explicamos a la computadora cómo realizar.

A continuación veremos cómo crear un bloque nuevo (procedimiento) que represente la acción de dormirse, y que será usado así:



Esta nueva acción va a indicar mejor que dormirse consta de los pasos anteriormente descritos.

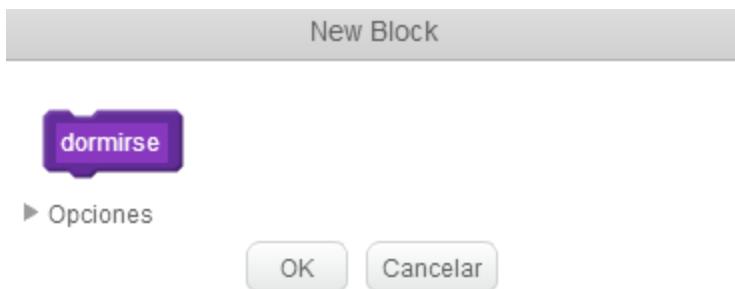
Para hacer esto vamos a “Más bloques” y presionamos el botón “Crear un bloque”:



Se nos abrirá una ventana así:



En esa ventana completamos el nombre del nuevo bloque:



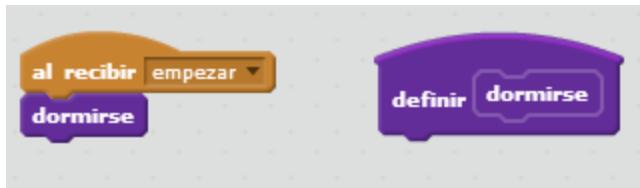
Cuando presionamos “OK” nos aparece lo siguiente en el editor:

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar



¿Qué es este bloque “definir dormirse”? Es la parte en donde le explicamos a la computadora qué significa para nosotros “dormirse”. Si no le explicamos nada, la computadora no hará nada cuando le pidamos ejecutar el bloque “dormirse”.

Es conveniente mostrar a los alumnos que esto pasa:

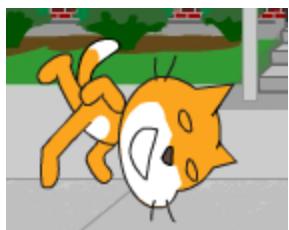


En este caso, como todavía no le explicamos nada acerca de “dormirse”, la computadora no hace nada cuando presionamos la banderita verde.

Lo que haremos a continuación es enseñarle al gato (al autómata) a “dormirse”, de manera que cuando usemos el nuevo bloque



El gato se dormirá



Actividad: Dejar a los alumnos definir dormirse como la secuencia de “acostarse” y “cerrar los ojos”.

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

La definición entonces quedará así:



Actividad: Ahora que sabemos definir nuevos bloques, definir el procedimiento “despertarse” que hace que el gato abra los ojos y se levante. Luego de definirlo, hacer que el gato avance un paso, se duerma, se despierte, salude y vuelva a su lugar.

Solución:



Cierre

Como cierre es importante hacer un repaso rápido de todo lo aprendido hasta el momento:

- Los comandos representan **acciones** y, si los ejecutamos, la computadora realiza dicha acción.
- Las **secuencias** de comandos son series de acciones que se realizan una a continuación de la otra **en orden**.
- Los comandos incluidos originalmente en el autómata se llaman **primitivas**.
- Podemos definir nuevos comandos, que llamamos **procedimientos**, para explicarle a la computadora cómo realizar nuevas acciones.

Lightbot

Objetivo

Introduciremos el concepto de programa como solución a un problema.

Al finalizar esta clase queremos que los alumnos comprendan:

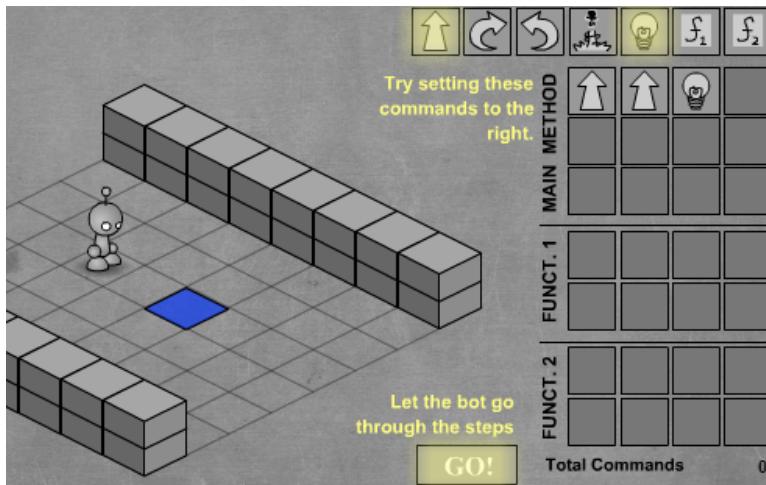
- Que los programas pueden resolver problemas específicos.
- Que es necesario pensar el problema y luego proponer una solución a través de la programación.
- Que existen formas más simples de resolver problemas si encontramos el patrón que permite descomponerlos (independientemente de que soluciones complicadas resuelvan el problema).
- Que no hay un único camino para resolver un problema.

Presentación

Daremos a los alumnos que hasta el momento realizamos actividades de índole recreativa, que nos daban total libertad para jugar con los bloques de Scratch. A partir de ahora, empezaremos a usar esos mismos bloques para resolver problemas. Habrá una secuencia de bloques que será una solución clara y otras que no lo serán. Pero para introducir el tema, primero jugaremos a un juego, en el que tenemos que programar para pasar sus niveles.

Desarrollo

Presentamos el juego Lightbot. En cada nivel el objetivo es lograr que el robot prenda todas las luces (ubicadas en los cuadrados azules del piso). Nos brindan una serie de acciones en forma de íconos (similar a los bloques en Scratch) y una grilla en donde ubicarlas. Además nos dan espacio para definir dos procedimientos (en el juego los llama “funciones”), aunque en los primeros niveles no necesitamos hacer uso de esto.



Las instrucciones que presenta el juego son las siguientes:



Avanzar.



Girar a la derecha.



Girar a la izquierda.



Saltar. Permite saltar por encima de un sólo bloque, o bien bajar de uno.



Prender luz.

Sin mencionar el espacio que nos permite definir nuevas tareas, presentamos la siguiente actividad.

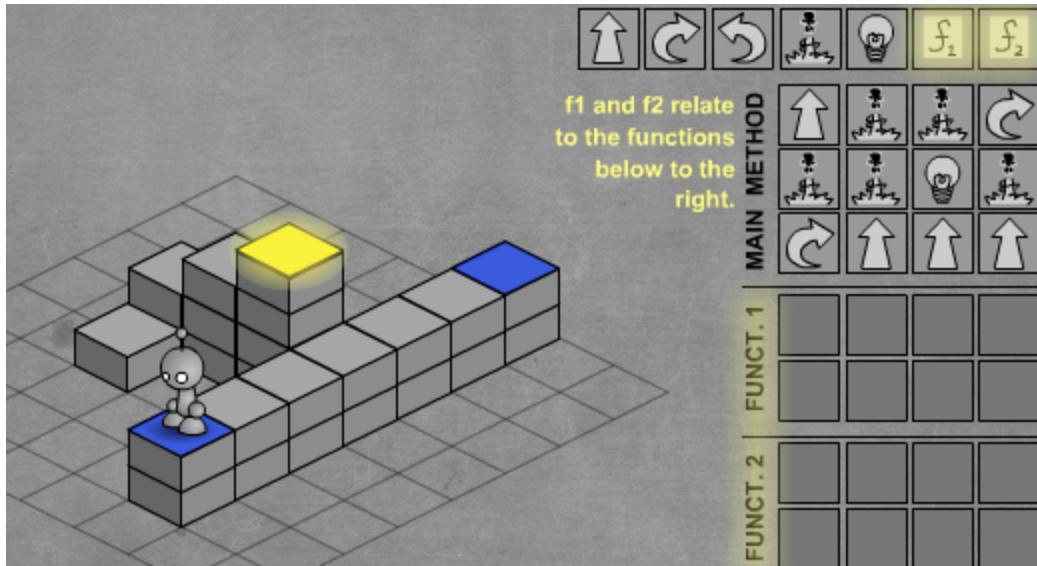
Actividad: Dejar que los alumnos avancen hasta el nivel 6, ayudándolos cuando se bloquen. Pedir a los que terminan primero que ayuden a los que se atrasan más, diciéndoles que les expliquen cómo avanzar, pero sin resolverles la tarea.

Es relativamente sencillo avanzar hasta el nivel 5 inclusive, sin embargo, en el nivel 6 ya necesitamos usar las subtareas que nos deja definir el juego, por falta de espacio.

Actividad: Dejar que los alumnos intenten resolver el nivel 6 sin usar subtareas y que vean que esto no es posible.

Así, llegamos a este punto sin usar subtareas y podemos ver que no alcanzan los espacios

de la sección “Main method”:



Aquí introducimos el concepto de procedimiento, o “función”, como lo llaman en el juego.

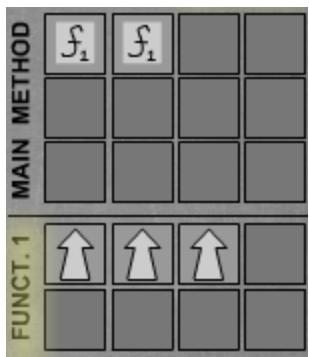
Los grupos de casilleros llamados “Funct. 1” y “Funct. 2”, forman un espacio que nos da el juego para extender las acciones que puede realizar el robot. Así como en Scratch nos permitían crear nuevos bloques, en este juego nos permiten definir dos tareas nuevas, aunque para esto nos dan un espacio limitado, ya que es parte del juego pensar la forma de ahorrarnos casilleros.

Entonces, podemos usar el espacio “f1”, por ejemplo. Al usarla le decimos al robot que realice todo lo que hayamos definido en “f1”:



En este caso el robot se moverá tres pasos adelante, ya que “f1”, significa (esta definido como) “avanzar” tres veces.

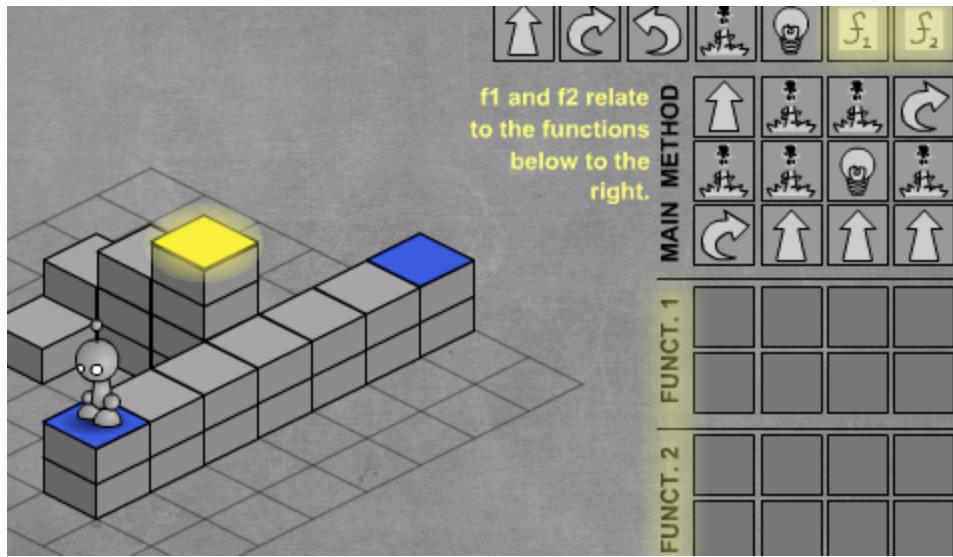
Si tenemos entonces



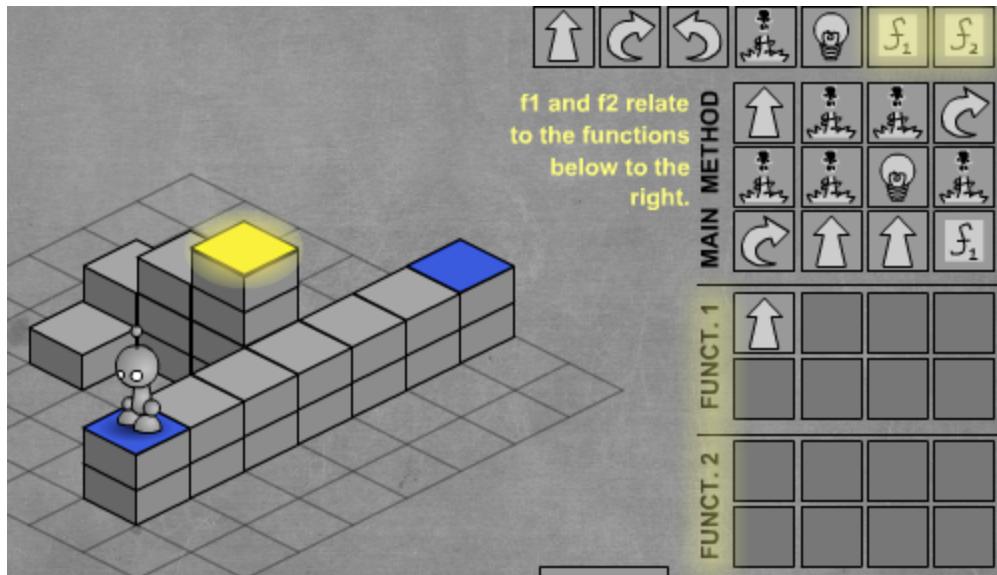
El robot se va a mover 6 veces hacia delante, ya que decimos que haga dos “f1”. Observar que en lugar de invertir 6 espacios, dentro de Main Method invertimos solo 2, y que cada “f1” vale por 3 avanzar. De esta manera la idea a partir de ahora es identificar patrones que nos permitan ahorrarnos casilleros, mediante la definición de nuevas tareas.

Actividad: Si tuviésemos que inventarle un nombre a “f1” que describa la acción que permite realizar, ¿cuál sería?

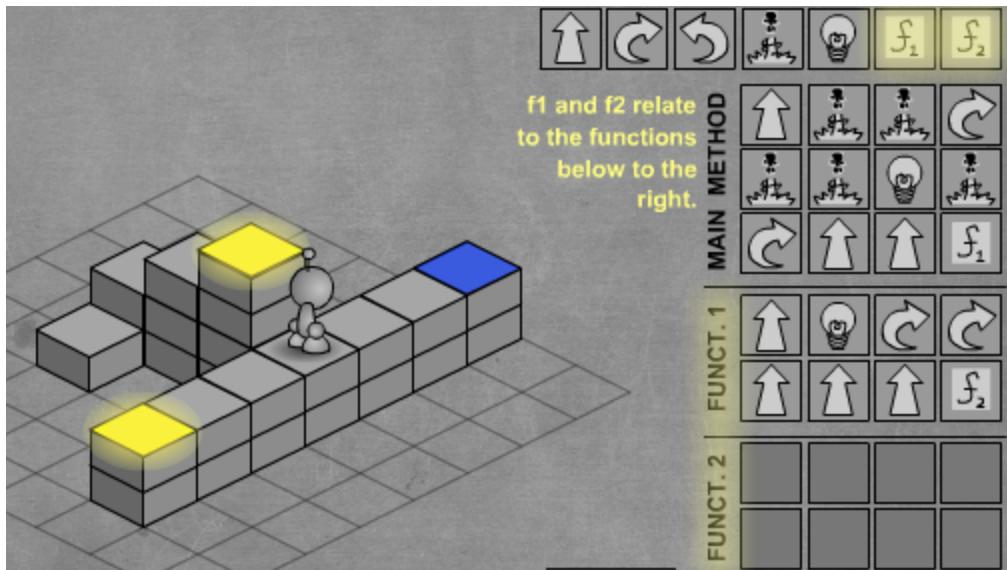
Ahora que sabemos esto, podemos volver al problema en el que nos quedamos sin celdas:



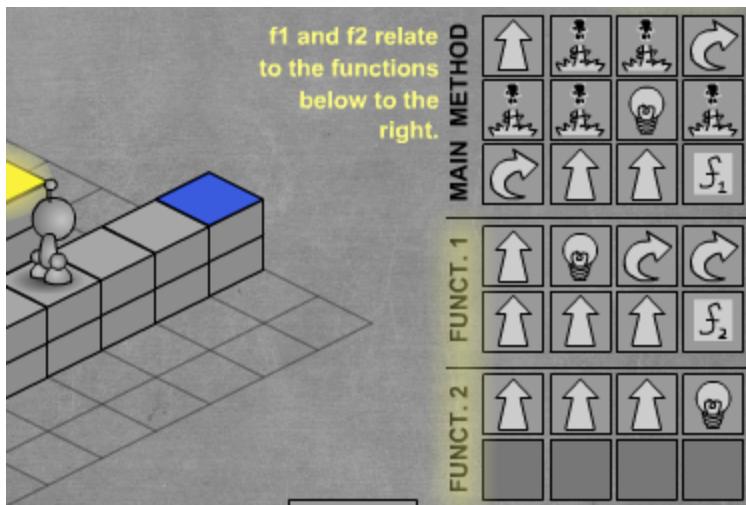
Una opción (seguida por muchos alumnos) para seguir indicando acciones al robot, resulta de continuar lo que hará este diciendo “f1”:



A partir de esto podemos seguir hasta completar “f1”:

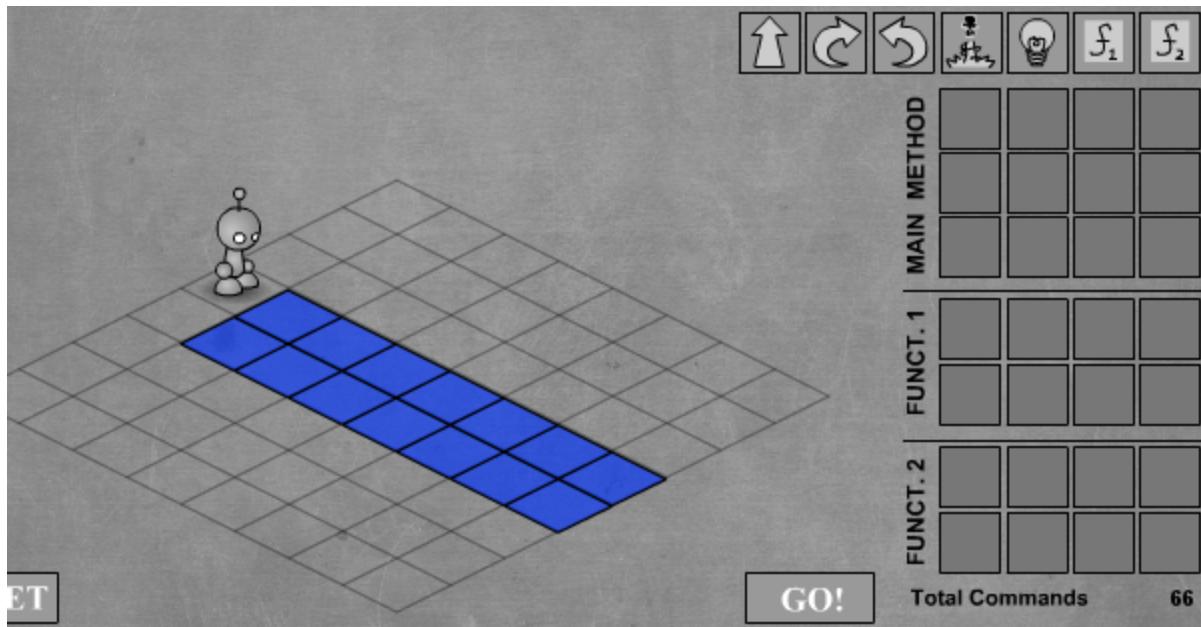


Como nos quedamos sin espacio incluimos en “f2” las acciones restantes:



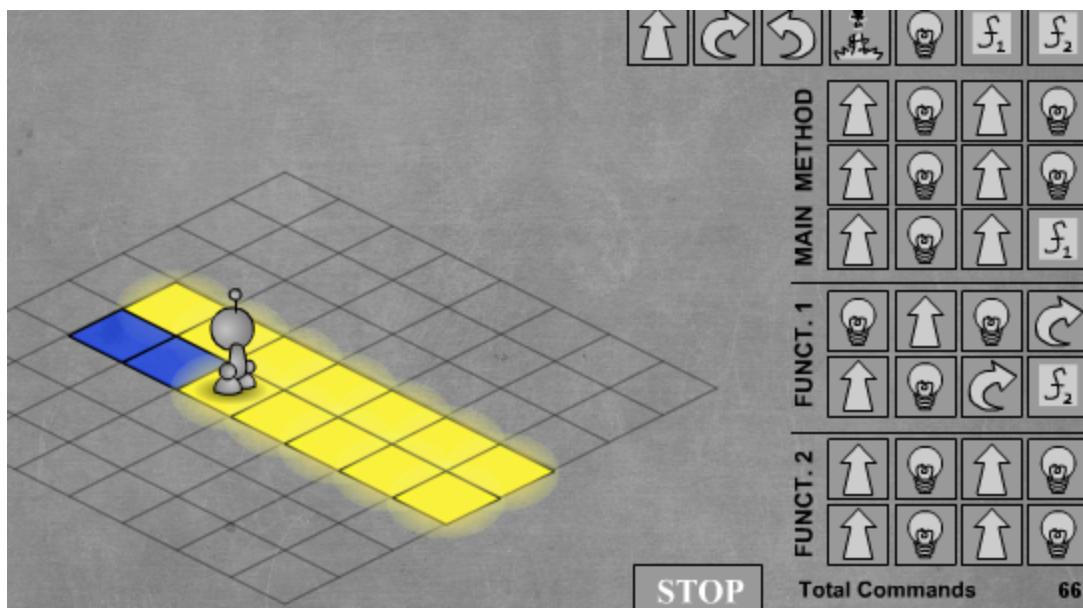
En esta solución en realidad no se pensó en dividir inteligentemente el problema. Si bien llegamos a prender todas las luces, con esta forma de utilizar las subtareas no vamos a llegar muy lejos en los niveles más complejos, dado que en algunos casos ni siquiera alcanzan las celdas usando las subtareas 1 y 2.

En el siguiente nivel nos vamos a encontrar con lo siguiente:



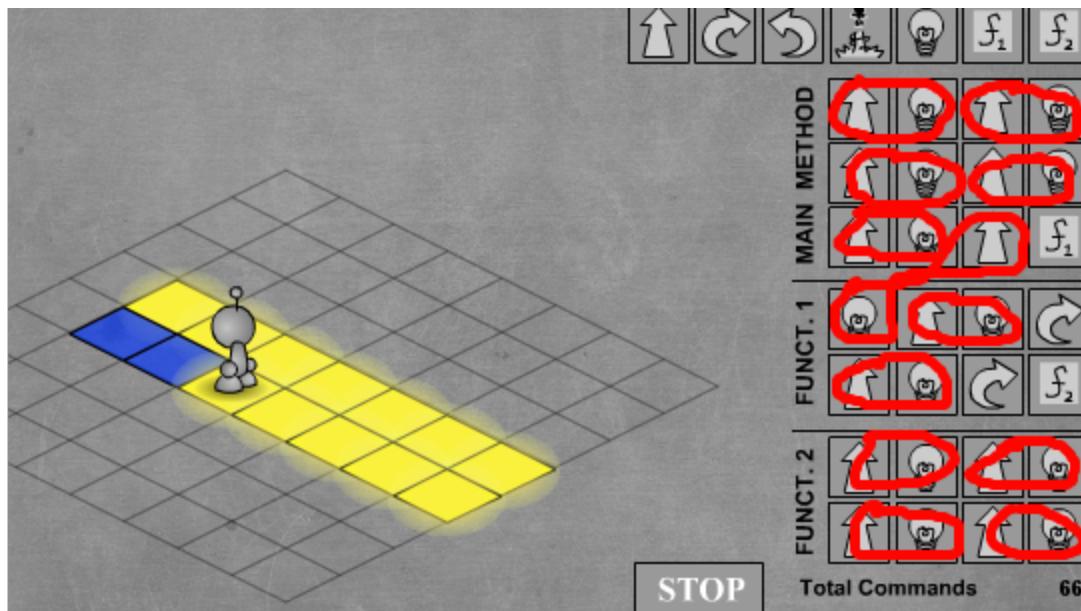
Actividad: Dejar que los alumnos intenten resolver este nivel por su cuenta, y frenar cuando la mayoría esté encarando una solución que no resuelve el problema.

Si intentamos repetir lo que hicimos en el nivel anterior (algo que hacen muchos alumnos), vamos a llegar a la siguiente solución:

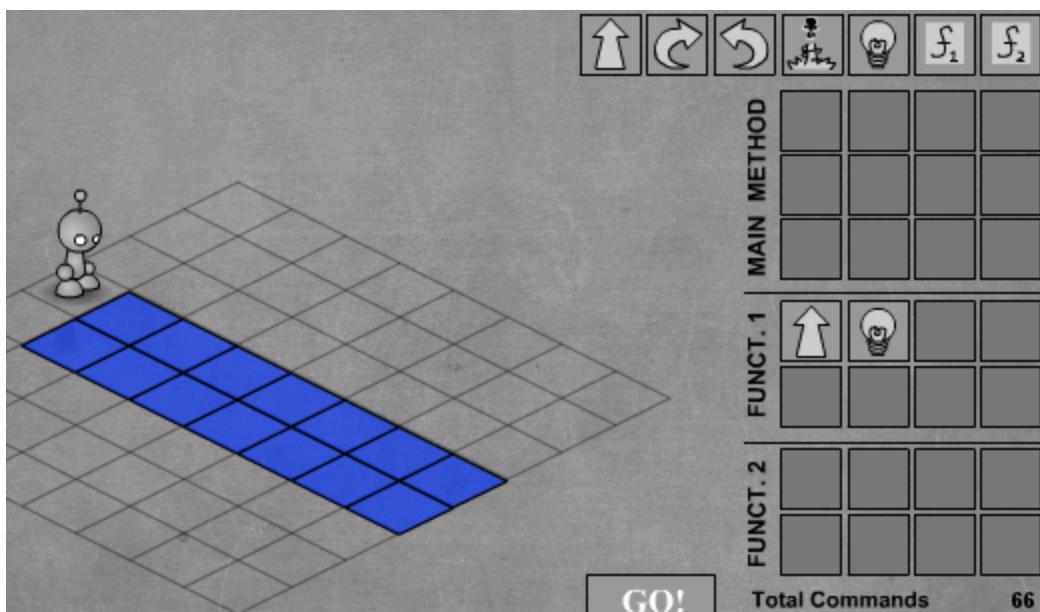


Esto es exactamente a lo que nos referimos en párrafos anteriores. Intentamos usar las

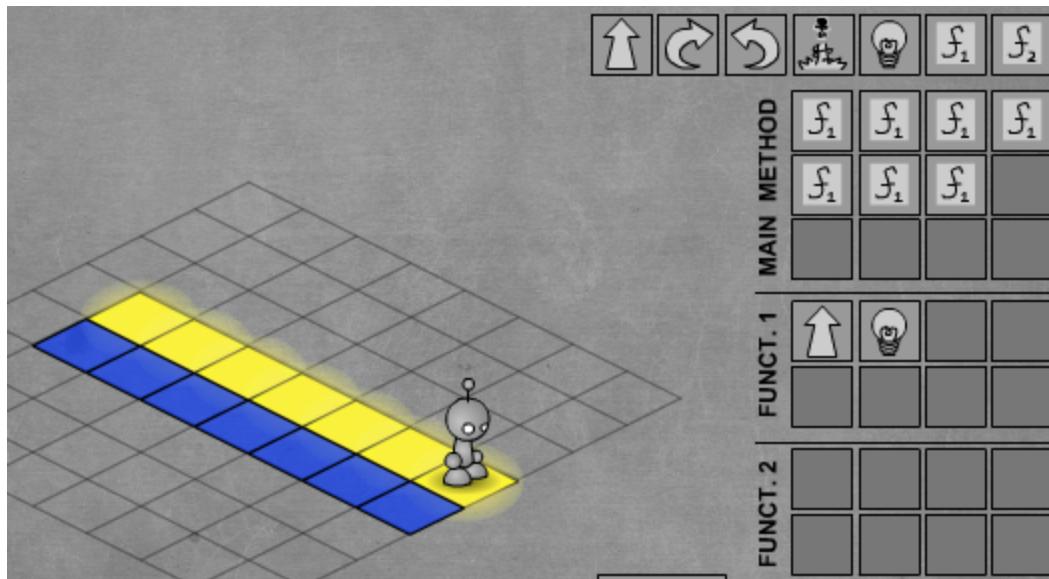
procedimientos para extender lo que podíamos escribir, pero eso no fue suficiente. Por el contrario, debemos pensar mejor para qué usamos los procedimientos. Cada función en realidad tiene que cumplir con un propósito bien definido, y ser usada en secuencias de acciones que se repiten muchas veces dentro del programa. En este caso podemos ver algo que se repite demasiado: “avanzar” y “prender luz”.



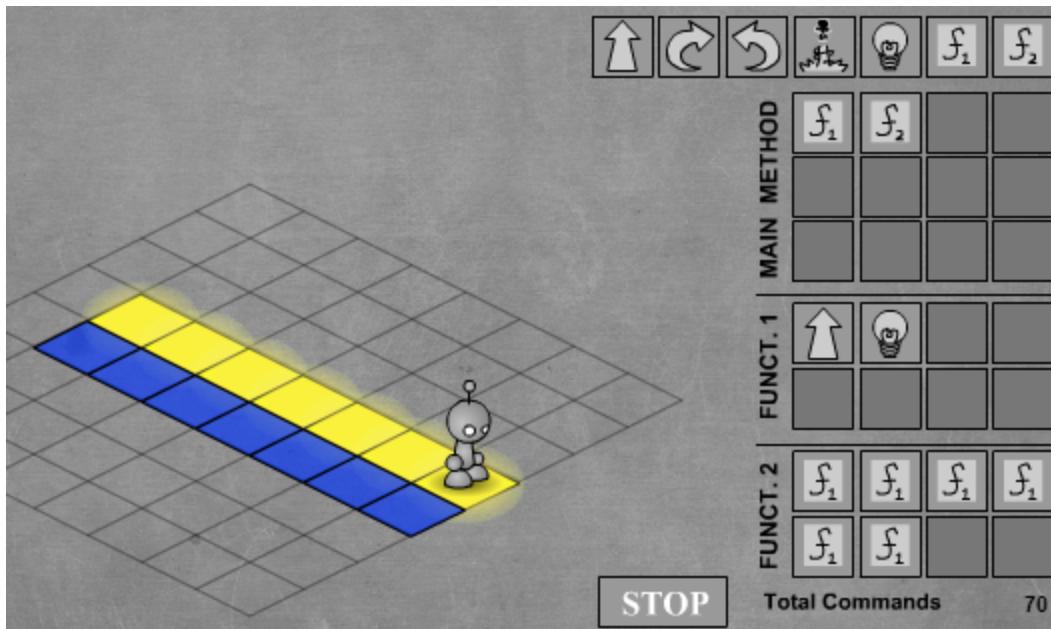
Teniendo en cuenta esto vamos a armar otras subtareas. Primero vamos a borrar todo lo que hicimos hasta ahora, y armar una función que sea exactamente “avanzar” y “prender luz”:



Contando con esta función, cada vez que necesitemos “avanzar” y “prender luz”, vamos a decir simplemente “f1”:

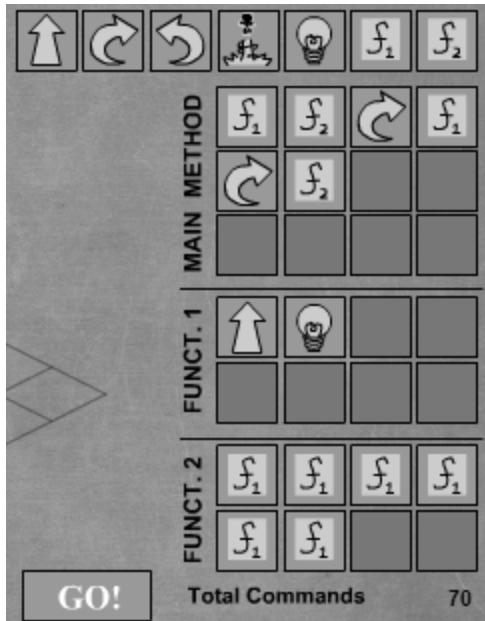


Lo que podemos notar rápidamente, es que podemos repetir lo que hicimos para la primera fila de luces, ahora para la segunda. Pero para eso, podemos utilizar “f2”:



Notar que en “f2” hacemos 6 veces “f1”, porque veremos que en la segunda fila con 7 nos caeríamos fuera de la fila.

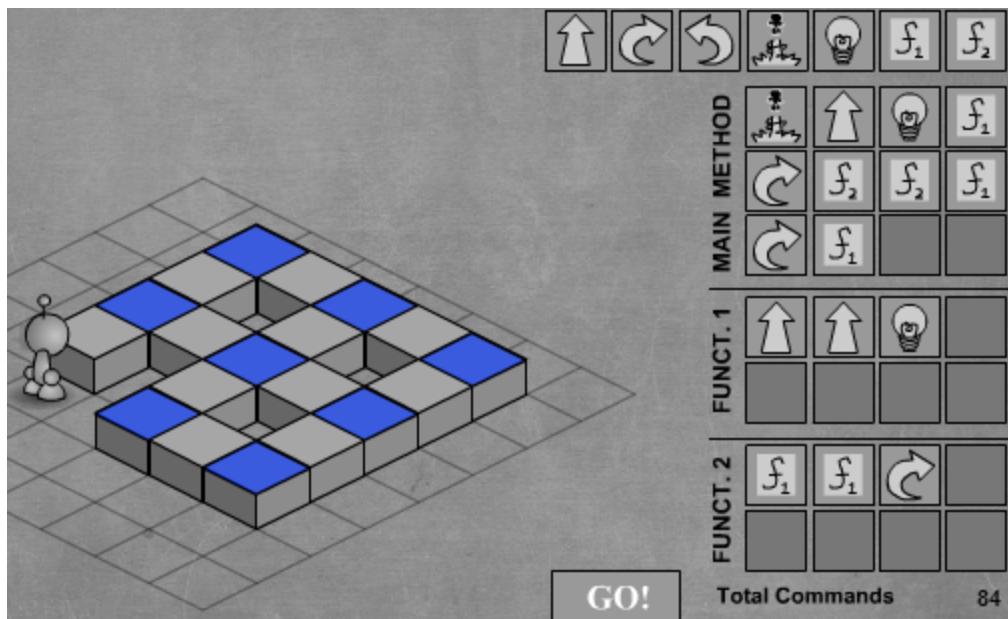
Entonces, simplemente hacemos lo siguiente para completar todo el nivel:



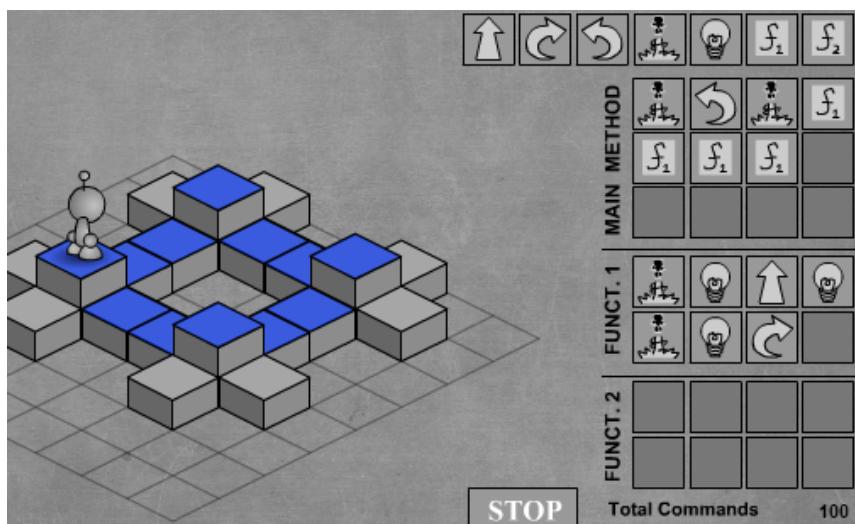
Los próximos niveles requieren de esta habilidad para resolverse.

Actividad: Avanzar un par de niveles más, hasta que se termine el tiempo de la clase (pensar en dejar unos 10 minutos para realizar el cierre de la clase).
 Las soluciones y patrones para los niveles siguientes son mostrados a continuación.

En este nivel lo que más se hace es avanzar dos veces y prender luz:



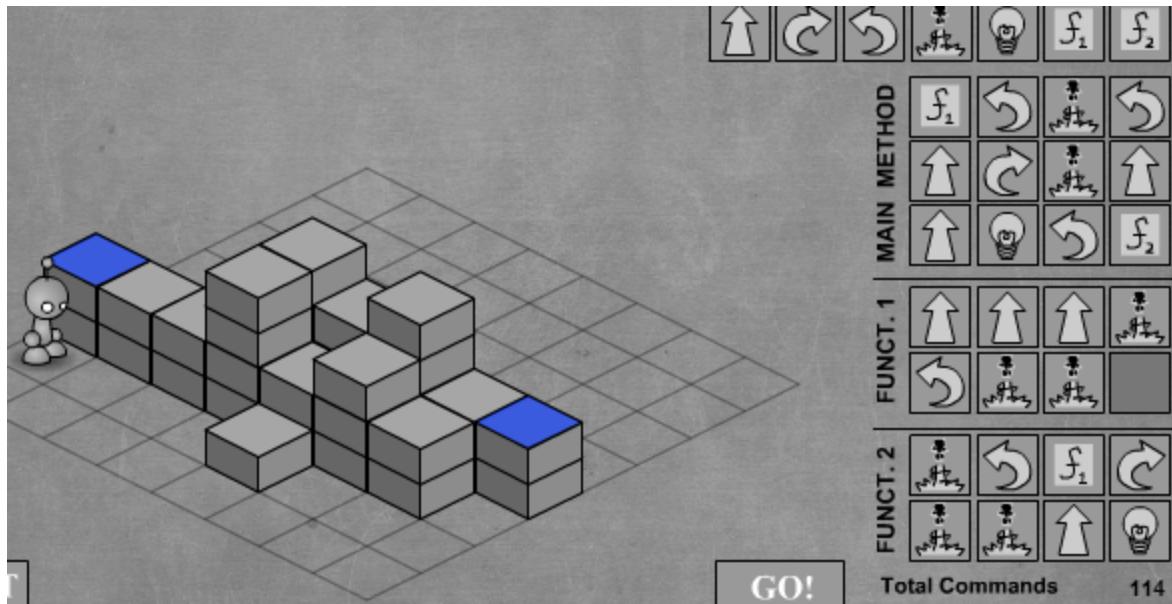
Notar que el siguiente nivel es un cuadrado, por lo que si resolvemos un lado con “f1”, simplemente repetimos eso 4 veces, un “f1” por cada lado.



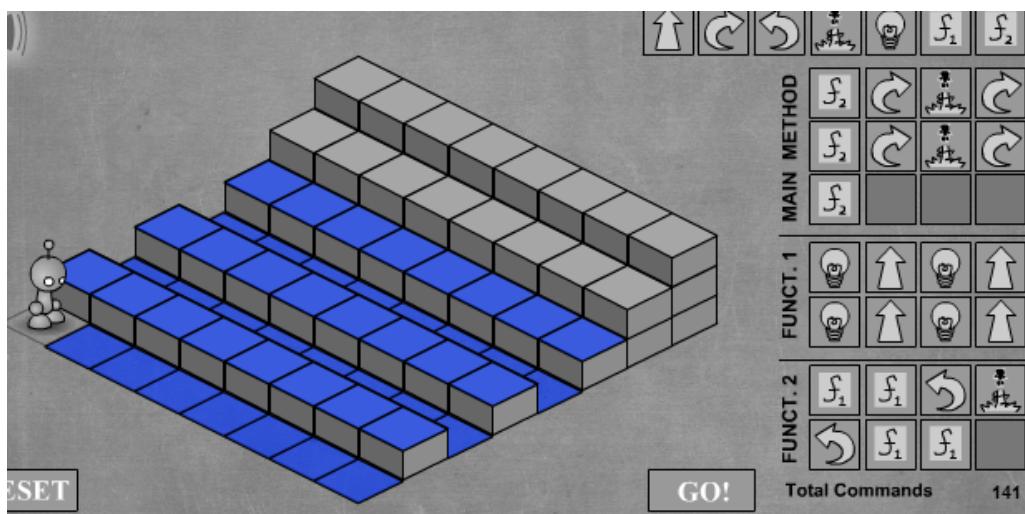
Este es uno de los niveles más difíciles, ya que lo que se hace dos veces es ir hasta el punto

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

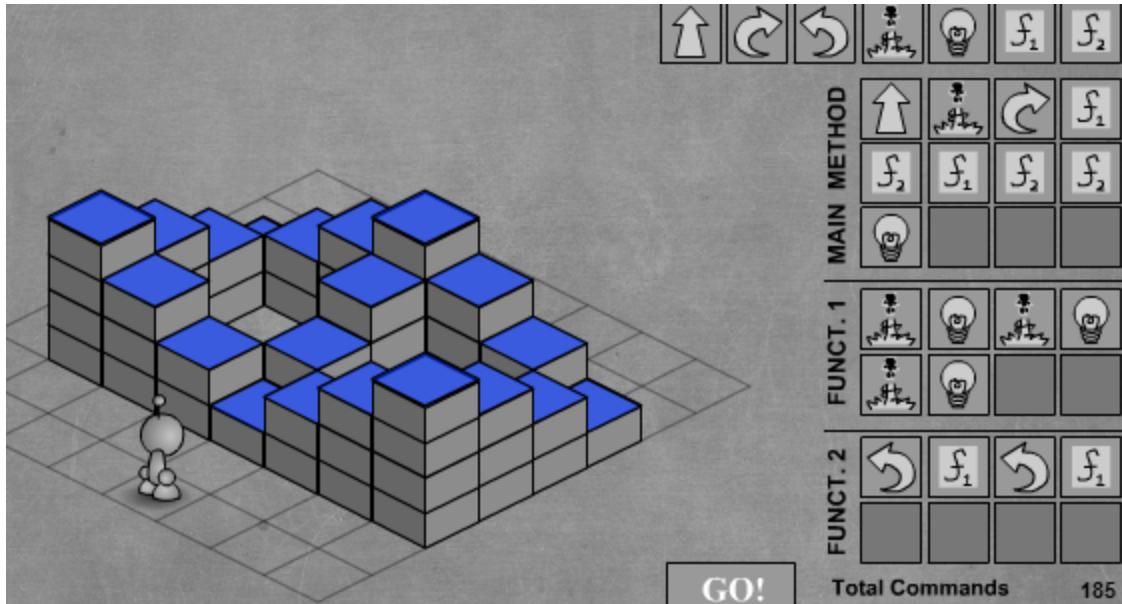
en el que decidimos si ir hacia una luz o la otra. Lo que conviene hacer es hacer una función que vaya hasta este punto y primero ir hacia la luz de la izquierda. Cuando la prendemos, lo que tenemos que hacer es saltar para ubicar el robot en donde comienza el nivel. Luego volvemos a ejecutar la función que nos lleva hasta el centro, y prendemos la luz de la derecha con los casilleros que sobran.



En este nivel conviene hacer una función que prenda 4 luces contiguas, ya que todas las filas tienen 8 luces menos la primera. En lugar de tratar a la primer fila como algo distinto nos conviene tratarla como si empezara como una luz, aunque no la tenga. Se ejecuta un prender luz en una celda que no contiene una, pero es una jugada válida en este juego.



Este último nivel es fácil de resolver, solo que puede pasarnos que apaguemos una de las luces que prendimos con anterioridad. Para arreglar esto simplemente volvemos a ejecutar un prender luz para volver a prenderla y pasar de nivel.



Cierre

Pensar qué es un programa luego de entender la lógica del juego y relacionarlo con las actividades anteriores en Scratch. Entender que en este caso se sabe exactamente si el programa resuelve o no el problema planteado, y que en los casos anteriores no había un problema bien definido a resolver.

Lo nuevo que vimos es el método de dividir un problema grande en partes más pequeñas, que se conoce en programación como **división en subtareas**. Esto consiste en:

- Identificar pequeñas tareas fácilmente explicables y asignarles un buen nombre para que se entienda qué representan
- Combinar estas pequeñas tareas para definir el programa que resuelve el problema

No me canso de saltar

Objetivos

A través de esta actividad queremos que los alumnos comprendan:

- Que cuando una tarea se vuelve repetitiva podemos reducirla al uso de una repetición
- Que si sabemos cuánto hay que repetir esa tarea, usamos una repetición simple y la tarea que haya que repetir
- Que las computadoras puede repetir una tarea cuantas veces sea necesario

Desarrollo

Tenemos el siguiente escenario:



El objetivo es hacer que el gato salte 30 veces para llegar a la última piedra.

Para eso sólo nos dan la siguiente primitiva:



Cuando ejecutamos una vez el saltar, el gato pasa de una piedra a otra y nos avisa que le falta una piedra menos:



Actividad: permitir a los alumnos intentar resolver la actividad con lo aprendido hasta el momento.

Los alumnos seguramente intenten arriba a una solución que posea muchas veces el comando saltar:



El problema es que esta solución es completamente incómoda, además que si el gato luego tiene que saltar otra cantidad de veces, se vuelve muy difícil visualizar cómo modificar esta secuencia.

Luego de que el problema sea entendido introducimos el bloque “repetir”, ubicado en la categoría Control:

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar



Lo traemos hasta el editor y vemos que posee dos partes. En una nos deja elegir la cantidad de veces que deseamos repetir algo:



Y luego nos deja un hueco en donde podemos enganchar una secuencia de comandos:



Actividad: Luego de esta explicación dejar que los alumnos intenten resolver el resto de la actividad.

La solución es la siguiente:



Cierre

Como cierre se deberían observar los siguientes aspectos:

- Las computadoras pueden repetir una tarea cuantas veces sea necesario.
- El bloque repetir nos permite ahorrar mucho código y simplificar lo que queremos expresar. Luego si hay errores o el problema cambia, es fácil cambiar el numerito o darnos cuenta que la secuencia es incorrecta.
- El bloque consiste de un número y la secuencia que deseamos repetir. Es saber ese número para poder usar este bloque.
- A partir de ahora siempre que una secuencia de comandos se repita una cantidad fija de veces, utilizaremos el bloque repetir.

Programar en papel cuadriculado

Presentación

En esta actividad vamos a programar en papel y nuestros programas representarán pasos para realizar un dibujo.

Desarrollo

Para esta actividad necesitaremos hojas cuadriculadas (también pueden ser cuadrículas dibujadas sobre hojas en blanco, pero es más fácil que ya sean cuadriculadas). Vamos a utilizar una hoja para programar y otra hoja para dibujar. Pondremos a cada una el título que le corresponde, “Programas” y “Dibujos” (respectivamente).

El conjunto de comandos (instrucciones) es el siguiente:

- — Mover un cuadrado adelante
- ← — Mover un cuadrado atrás
- ↑ — Mover un cuadrado arriba
- ↓ — Mover un cuadrado abajo
- ↖ — Pintar casillero

Además, delante de cada comando podemos decir cuántas veces debe ejecutarse. Por ejemplo, lo siguiente significa “pintar hacia la derecha 2 cuadros”:

(↖→) 2

Esto incluye al cuadrado sobre el que estamos parados.

Programaremos de forma similar al Lightbot, pero de arriba hacia abajo, poniendo **un comando por renglón**.

(↗↑) 3

(↗→) 3

(↗↓) 3

(↗←) 3

Esta secuencia forma un cuadrado.

Sin embargo, a diferencia de Lightbot, a cada procedimiento le asignaremos un nombre. Así, cada programa que hagamos tendrá esta forma:

“nombre del procedimiento”

definición del procedimiento

Para ilustrar, veamos el siguiente procedimiento:

“Dibujar ventana”

(↗↑) 3

(↗→) 3

(↗↓) 3

(↗←) 3

A su vez, los procedimientos definidos pueden utilizarse en otros procedimientos

“Dibujar ventana doble”

[Dibujar ventana]

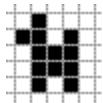
→ 4

[Dibujar ventana]

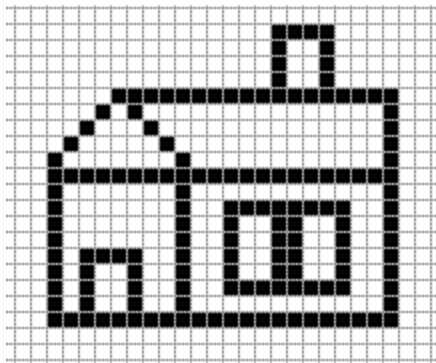
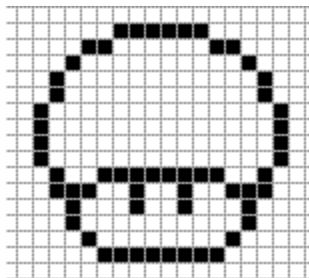
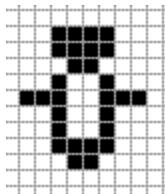
Como podemos observar, para utilizar un procedimiento debemos escribir su nombre entre corchetes y lo utilizamos como si fuese uno de los comandos ya existentes.

El nombre que inventemos debe ser corto, aunque legible y descriptivo. Si nuestro procedimiento hace muchas cosas y nos cuesta asignarle un nombre, significa que no estamos separando bien el problema.

Actividad: Hacer un programa que describa el siguiente dibujo.



Actividad: Realizar más ejercicios, hasta que los alumnos comprendan los conceptos involucrados para dividir los dibujos en distintas partes y definir los programas.



Variante: que cada alumno piense su propio dibujo, lo programe y se lo pase al compañero para que lo reproduzca. Deben intercambiarse sólo los códigos y no el dibujo final.

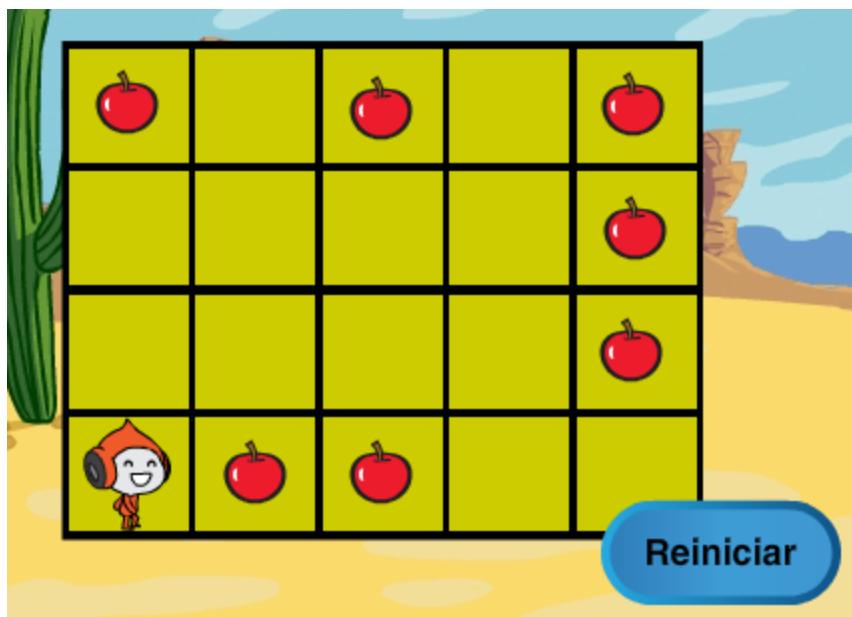
Cierre

Como cierre sería interesante hacer algunas preguntas como las siguientes.

- ¿Se dificultó más programar en papel?

- ¿Es mejor hacer varias subtareas o programar todo el dibujo en una sola? ¿Por qué?

El marciano en el desierto



Objetivo

Comenzaremos a resolver actividades de resolución de problemas usando Scratch.

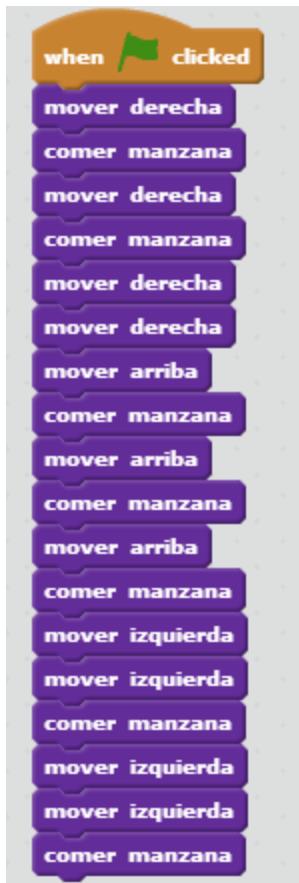
Desarrollo

Haremos la actividad del “Marciano en el desierto”. El objetivo es comer todas las manzanas que hay en el tablero utilizando las siguientes instrucciones:



Actividad: Dejar que los alumnos intenten resolver el problema como se les ocurra.

Una solución primitiva puede ser la siguiente:



Pero podemos ver que en esta solución se repiten las siguientes secuencias:



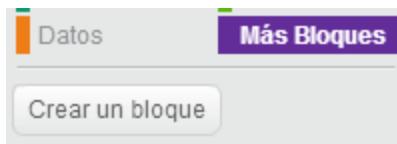
Actividad: Habiendo marcado las cosas que se repiten, pedir a los alumnos que vuelvan a resolver el problema, pero ahora utilizando el bloque “repetir”.

Así, podemos mejorar la solución utilizando el bloque “repetir” y llegamos al siguiente resultado:



No contentos con el resultado, podemos utilizar la herramienta vista en clase llamada “procedimiento”, que consiste en crear nuevos bloques y explicarle a la computadora cómo realizar cosas que hasta ahora no sabe hacer.

Para hacer esto vamos a “Más bloques” y hacemos clic en “Crear un bloque”:



Esto nos abre una ventana así:



► Opciones

OK

Cancelar

En esa ventana completamos, por ejemplo:



► Opciones

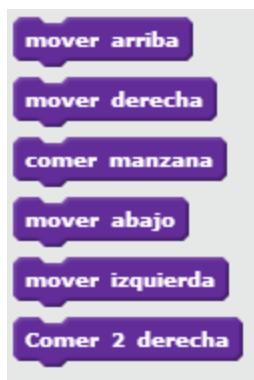
OK

Cancelar

Cuando presionamos "OK" nos aparece lo siguiente en el editor:



La creación de este bloque nos agregó una instrucción al final de las que ya estaban:



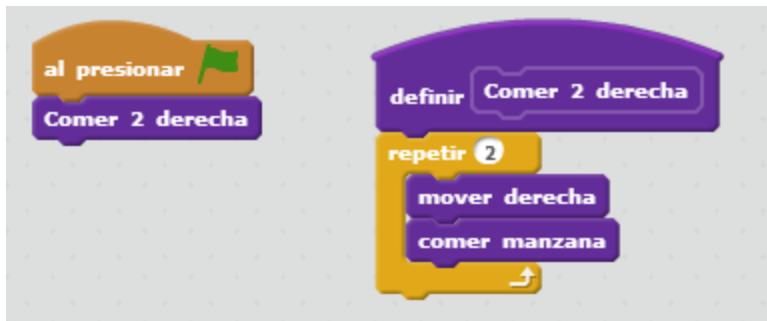
Esta nueva instrucción puede ser utilizada como cualquiera de los bloques existentes.



Pero con esto todavía no le explicamos a la computadora qué significa “Comer 2 derecha”. Para hacer esto completamos su definición con lo que queremos que haga:



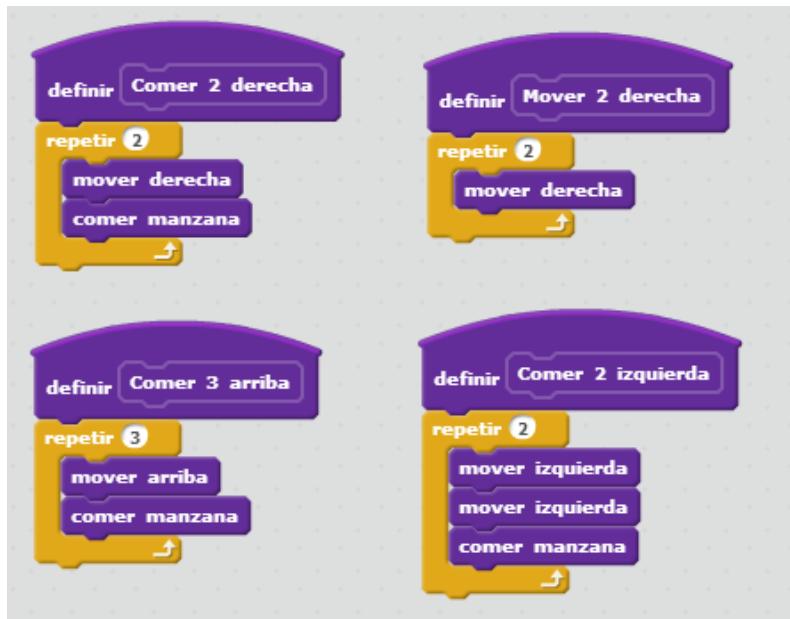
Con esto logramos lo siguiente:



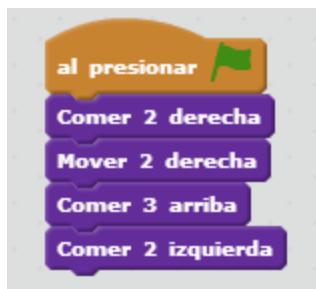
Cuando la computadora ejecute el bloque “Comer 2 derecha” va a ir a su definición y va a realizar lo que le hayamos explicado, de la misma forma que el robot de Lightbot y sus subtareas.

Actividad: Pedir a los alumnos que completen los demás procedimientos.

De esta manera, definimos todos estos bloques nuevos:



Y el programa finalmente nos queda así:



Cierre

Si bien para la computadora la solución sigue siendo la misma (el robot se come las manzanas de la misma forma que antes), para los humanos esta descripción es mucho más fácil de entender. Lo conversamos entre todos diciendo que esta sería la forma con la que le explicarían a sus amigos cómo comer todas las manzanas del tablero de manera muy simple. Luego cada pequeña tarea contendrá los detalles de cómo se realiza específicamente. Además esta forma de resolver el problema es muy útil cuando el problema se vuelve

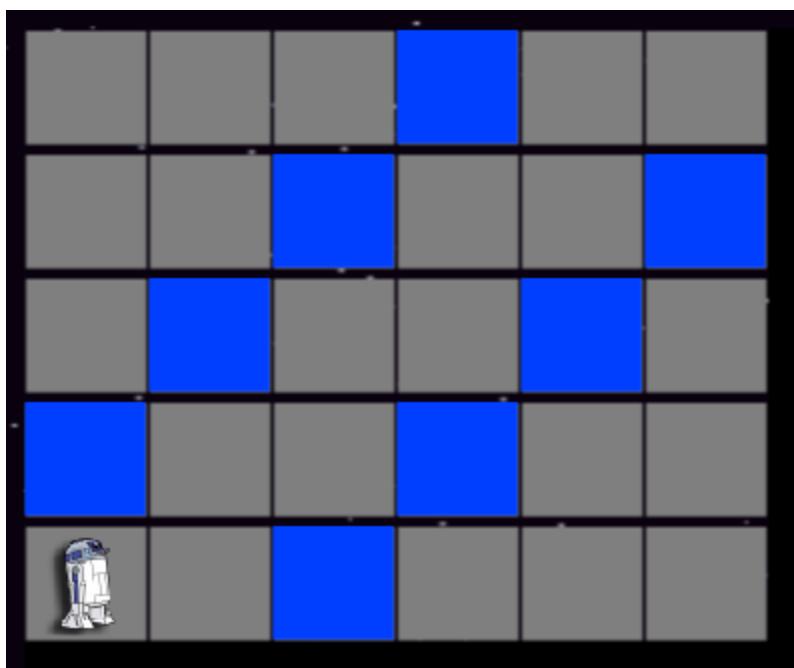
complejo.

[Lightbot en Scratch](#)

Objetivo

El objetivo de esta y las siguientes actividades es seguir practicando actividades en donde deben partir un problema en procedimientos y resolverlo.

Desarrollo



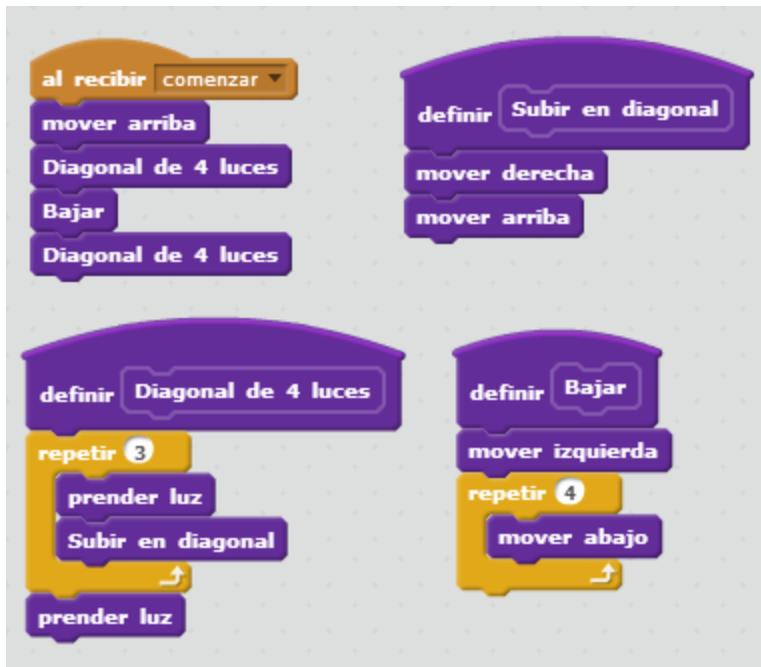
La idea en esta actividad es que los alumnos se den cuenta de que les conviene crear un procedimiento que mueva al robot en diagonal ya que eso es lo que se repite a menudo. Luego pueden crear dos procedimientos más:

- Uno para prender una diagonal de 4 luces
- Otro para pasar a la siguiente diagonal de luces

Observar que ambas diagonales son idénticas, solo que ubicadas en lugares diferentes.

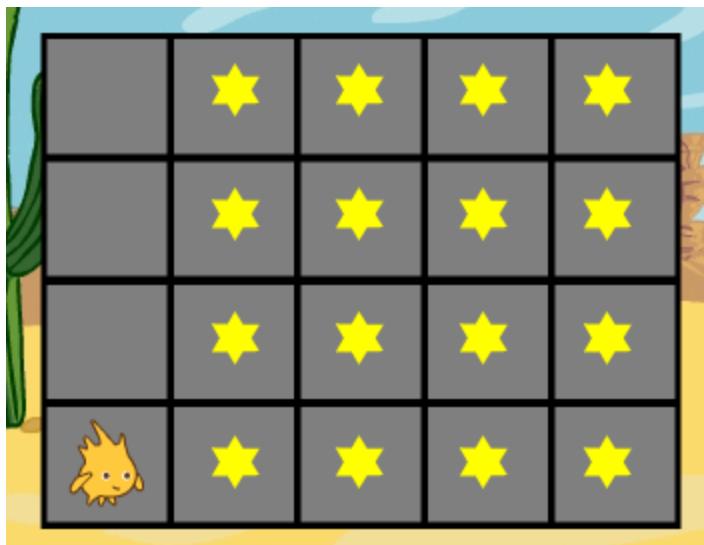
Actividad: Dejar que los alumnos intenten resolver el problema.

La solución puede ser la siguiente:

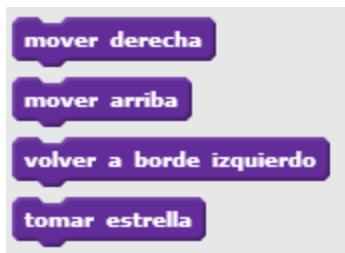


Si vemos que no están creando procedimientos, es importante guiarlos y decirles cosas como “esto es como en Lightbot, si no aprendés ciertas cosas, cuando sea más difícil no vas a saber qué hacer”. Si no se les ocurre cómo partir el problema, mostrarle a cada uno por separado que podría dividirlo en dos diagonales.

El recolector de estrellas



Boob es un extraterrestre que se alimenta de estrellas. Su objetivo es tomar todas las estrellas del tablero, contando con las siguientes primitivas:

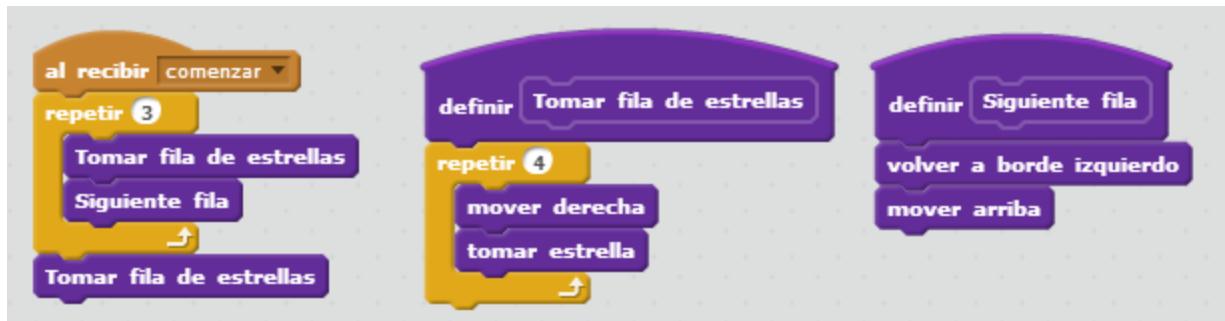


No tenemos instrucciones para movernos hacia abajo o la izquierda. El bloque “Volver al borde izquierdo” simplemente nos regresa a la primer celda desde el borde izquierdo, respetando la fila en la que nos encontramos.

¿Cuál será nuestra estrategia? Consumir fila a fila todas las estrellas.

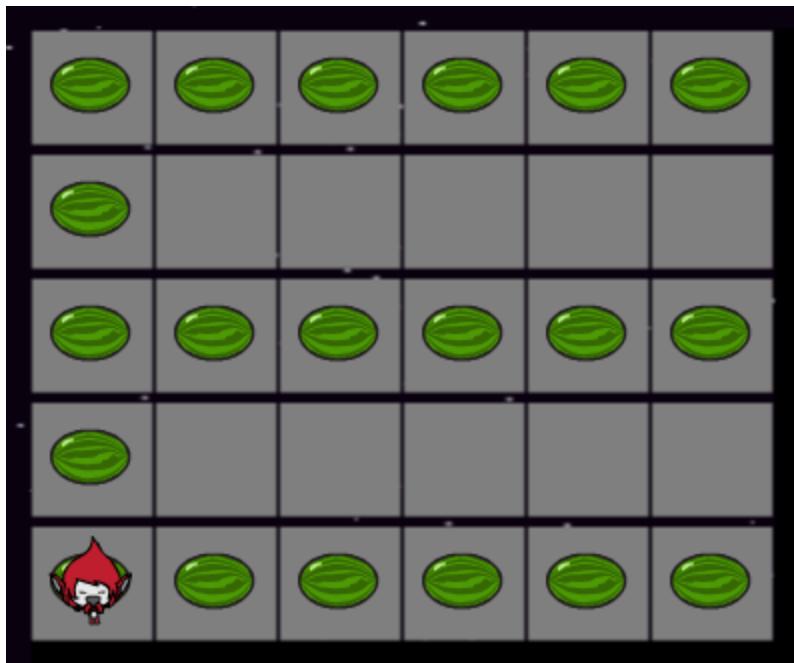
Actividad: Dejar que los alumnos resuelvan el problema.

La solución es la siguiente:



Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

María, la come sandias



El objetivo es morder todas las sandías (maría come las sandías por la mitad).

Las primitivas son las siguientes:



Actividad: discutir entre todos la estrategia a usar.

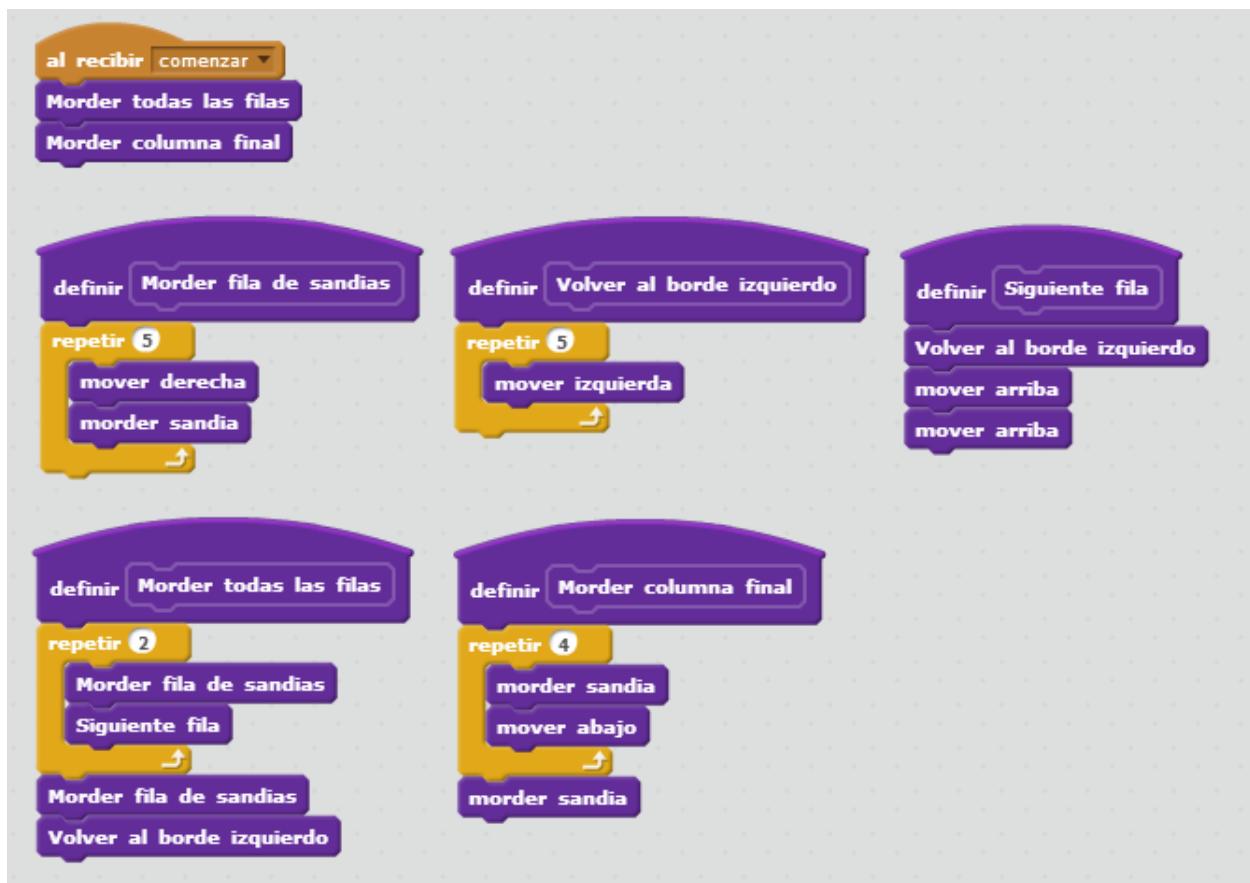
Hay varias formas de resolver este problema. Una puede ser definir un procedimiento que coma una fila de sandías, y luego de comer las 3 filas, comer la columna sobrante. En este caso usamos al menos 2 procedimientos, uno para comer una fila y otro para comer la columna final.

Con esta misma estrategia también podemos comer primero la columna y luego las 3 filas. Es indistinto.

Lo que sería interesante definir además de estos procedimientos es uno más que nos permita volver al principio de la fila (el borde izquierdo del tablero). Esto en la actividad anterior era una primitiva, y ahora no contamos con ella. ¿Podemos definirla nosotros mismos? La respuesta es sí, y conviene hacerlo.

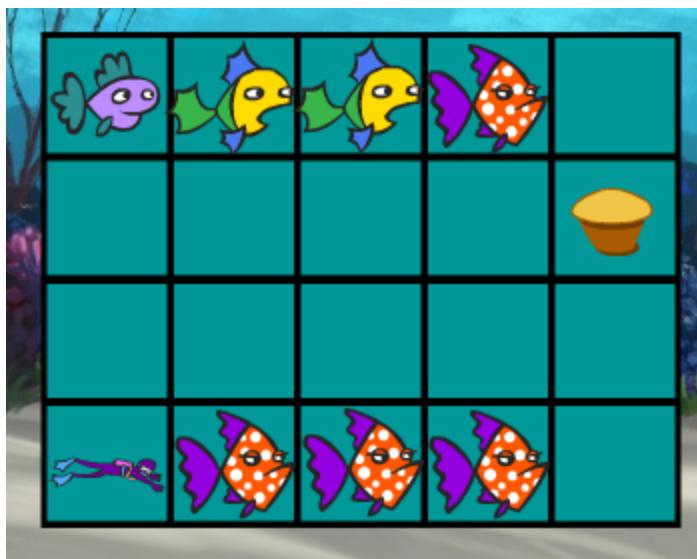
Actividad: Luego de esta discusión dejar que los alumnos intenten plantear toda la solución. Si se complica, primero definir el procedimiento de comer una fila, luego definir el procedimiento de volver al inicio de la fila, luego otro que consuma 3 filas, y el última que consuma la columna.

Una solución es la siguiente:



Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

Alimentando a los peces



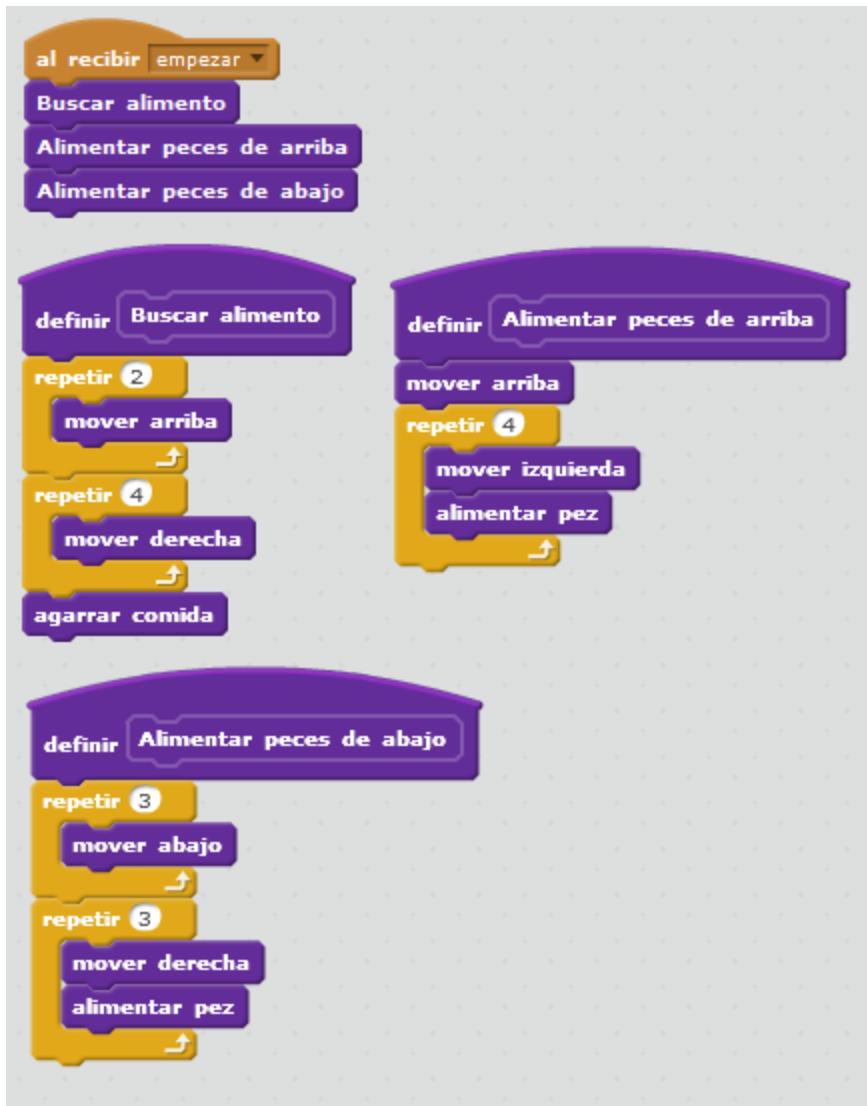
En esta actividad debemos ir a buscar la comida y luego pasar por cada pez para alimentarlo. Si tocamos un tiburón, nos come.

Las instrucciones a utilizar son las siguientes.



Actividad: Discutir entre todos qué procedimientos se podrían crear. Por ejemplo, podría crearse un procedimiento para ir a buscar el alimento y otro para alimentar a los peces. Pensar también en qué partes sería útil usar un ciclo “repetir”.

Una solución muy conveniente es la siguiente:



Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

Instalando juegos



Mika es la encargada de la biblioteca. Le pidieron que instale un videojuego en todas las computadoras del lugar. Es un proceso tedioso, así que quiere automatizarlo.

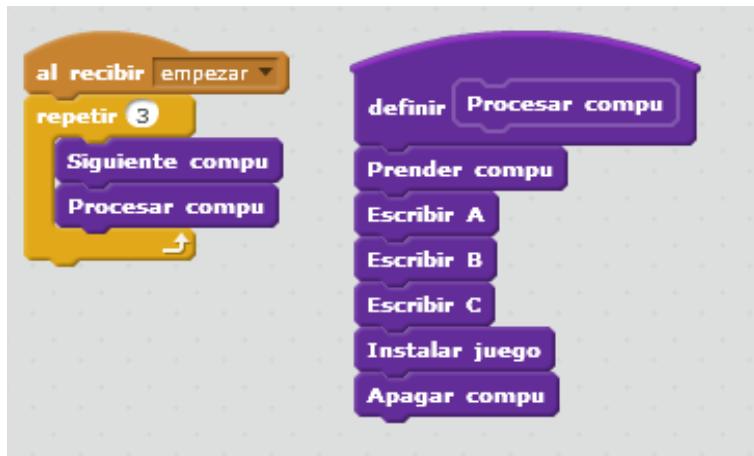
Estas son las instrucciones primitivas:



A una computadora primero hay que prenderla e ingresar la contraseña, que es ABC. A continuación podemos instalar el juego, y al terminar debe quedar apagada.

Actividad: definir un procedimiento que permita realizar el proceso descrito y definir un programa que lo ejecute por cada computadora de la biblioteca.

La solución es la siguiente:



Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

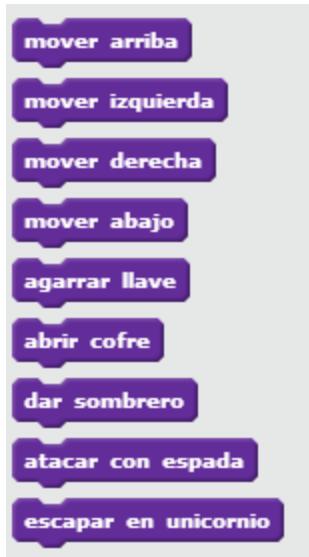
La gran aventura del mar encantado



Esta actividad pretende explotar al máximo la técnica de división en subtareas. El objetivo principal es escapar en el unicornio con la princesa, pero para eso, tenemos que recorrer una compleja aventura:

- Tenemos que ir a buscar la llave
- Con la llave tenemos que ir al cofre y que nos dará un sombrero
- Tenemos que darle este sombrero al mago para que nos regale una espada
- Con esa espada podemos ir a luchar contra el caballero oscuro y rescatar a la princesa
- Con la princesa tenemos que ir hasta el unicornio y escapar

Las primitivas que nos dan son las siguientes:



¿Qué puede salir mal?

- No podemos agarrar la llave si no estamos en la casilla donde está la llave
- No podemos abrir el cofre sin la llave y sin estar en el cofre
- No podemos darle el sombrero al mago si no tenemos el sombrero y estamos con el mago
- No podemos atacar con espada al caballero si no tenemos la espada y estamos en la casilla del caballero
- No podemos escapar con la princesa en el unicornio si no estamos con la princesa y en el unicornio.

Podemos pasar por encima de cualquier figura sin problemas (no tenemos que esquivar ningún obstáculo).

Actividad: Dejar que los alumnos resuelvan la actividad como se les ocurra. Tener escritos los pasos a seguir para que todos puedan visualizarlos cómodamente y no se pierdan. Aconsejar en todo momento que si no dividen en distintos procedimientos la tarea puede volverse compleja.

Una solución es la siguiente:



Reparando la nave



El marciano llamado Tino tiene una misión en el espacio, pero tuvo que realizar un aterrizaje de emergencia en un planeta desconocido. Tuvo suerte, porque cerca del aterrizaje se encontró con dos materiales (carbón y hierro) que le van a servir para repararse nave y continuar su aventura.

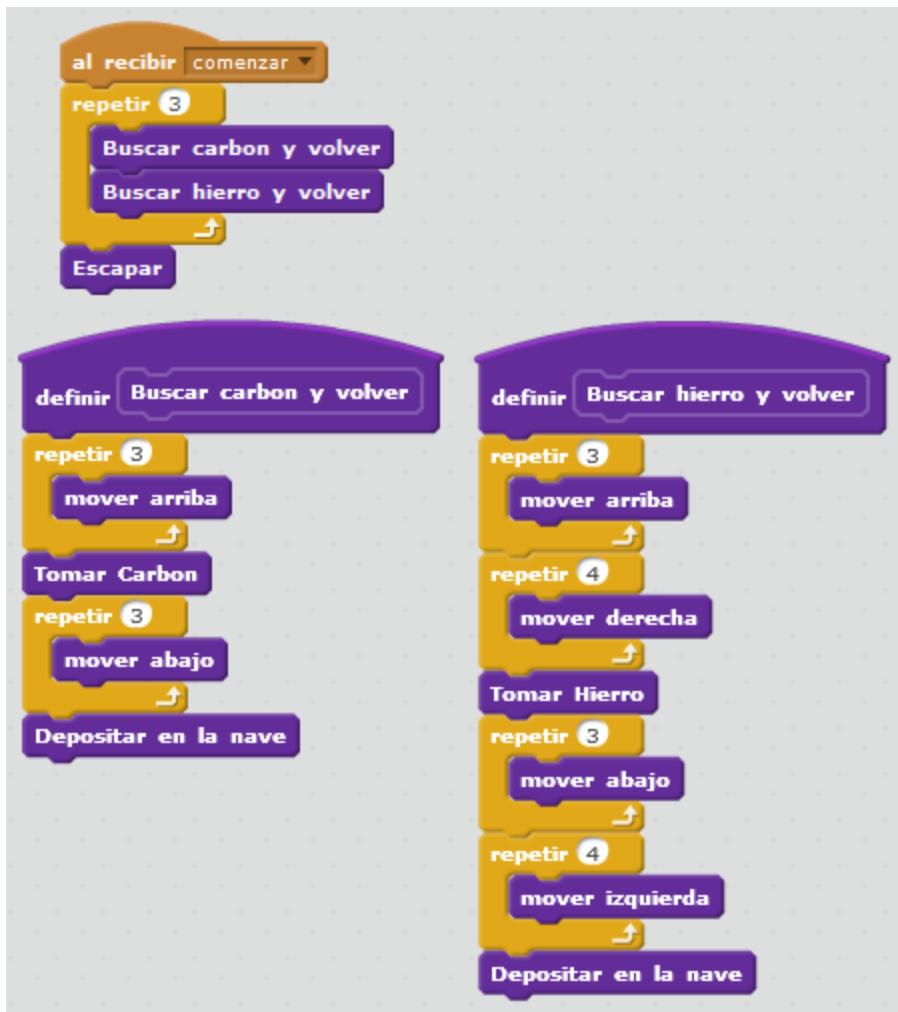
Las primitivas son las siguientes:



Debemos buscar 3 porciones de carbón y 3 porciones de hierro. El marciano sólo puede tomar un pedazo de carbón o hierro por vez, así que tendrá que realizar 3 viajes por cada material que necesita. Cuando termine de recolectar los recursos suficientes podrá escapar del planeta.

Actividad: hacer un programa que resuelva este problema.

Existen varias soluciones que utilizan correctamente los conceptos vistos hasta el momento. Una de esas posibles soluciones es la siguiente:



Alternativas condicionales

Objetivos

En estas clases vamos a aprender a recorrer secuencias de tamaño fijo junto con alternativas condicionales.

Salida del aula cambiante

Hasta el momento creamos programas que permitían resolver situaciones en escenarios conocidos. Por ejemplo, cuando movíamos al robot sabíamos al momento de hacer el programa, donde estaban las luces, las dimensiones del tablero, la posición original del robot, etc. ¿Qué pasaría si nuestro programa tuviese que funcionar para distintas posiciones originales del robot o distintas ubicaciones de las luces?

Planteando el escenario

Volvamos al ejemplo del aula. Tenemos ahora un aula con dos puertas (o una puerta y una ventana). Sabemos que cada día una y solamente una de las puertas estará abierta, pero no sabemos cuál en cada día. Queremos que el programa que nos permite salir del aula funcione para cualquier día.

Para ello vamos a contar con una nueva posibilidad para programar nuestro autómata: el bloque **Si<condición> Entonces<secuencia de acciones>**. El bloque **Si** nos permite ejecutar una secuencia de acciones si se cumple determinada condición. Además existe una variante de los bloques **Si** que se denomina que **Si<condición> Entonces<secuencia de acciones 1> Si No <secuencia de acciones 2>**. En esta variante se ejecuta una secuencia de comandos indicada a continuación del **Si No** en caso de que la condición sea falsa (si es verdadera se ejecutan las instrucciones a continuación del **Si**).

Por otra parte, el docente puede apreciar si una puerta está trabada o no desde cualquier posición (no necesita acercarse).

Actividad: Construir grupalmente una solución que guíe al docente a la salida, 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 eso efectúe el recorrido que lo lleva a esta puerta o a la otra. En ambos casos, el programa terminará abriendo la puerta y atravesando la salida.

La elección del mono

En esta actividad tenemos un escenario cambiante, algo que se va a repetir en las próximas actividades de aquí en más.

Podemos toparnos con este escenario



o con este otro



El objetivo es simple: comer manzana cuando hay manzana y comer banana cuando hay banana. En la casilla a la derecha del mono, hay manzana o hay banana, pero no ambas a la vez.

Las primitivas son las siguientes:



Para poder resolver esta actividad necesitamos un bloque nuevo, llamado “si.. entonces”:



Este bloque se encuentra en la categoría “control”.

Este bloque nos servirá para poder preguntar primero si se cumple cierta condición y sólo en ese caso ejecutar un comando. Pero primero vamos a dividir el problema.

El procedimiento que debemos definir tiene que resolver el problema de comer una manzana o bien una banana, así que lo llamaremos “comer fruta”:



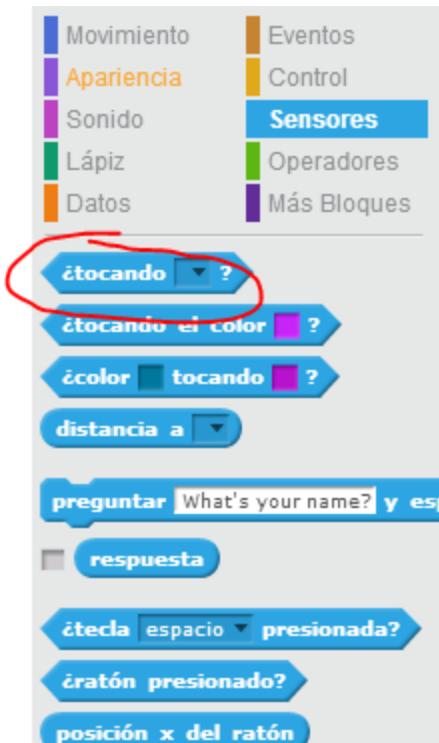
Nuestro programa será tan simple como:



Pero el problema de comer fruta, a su vez puede ser dividido en otros dos:



Para preguntar si estamos tocando una manzana o una naranja vamos a la categoría Sensores y seleccionamos la pregunta “¿tocando ...?”:



Observar que hay una flecha pequeña, que nos permitirá seleccionar de una lista qué objeto deseamos preguntar si estamos tocando. Para ello deberemos hacer clic exactamente sobre la flecha.

Seleccionamos manzana:



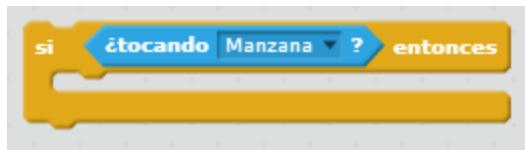
La forma que posee este bloque es idéntica al hueco que posee el bloque “si... entonces”:



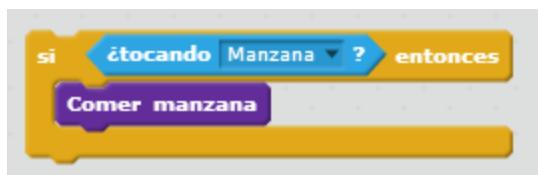
Esto no es casual, ya que el bloque “si... entonces” toma dos cosas. Primero toma una condición, y luego una secuencia de bloques. Si la condición es verdadera, la secuencia se

ejecuta, y sino no ejecuta nada.

Entonces, al combinar estas dos cosas tenemos:



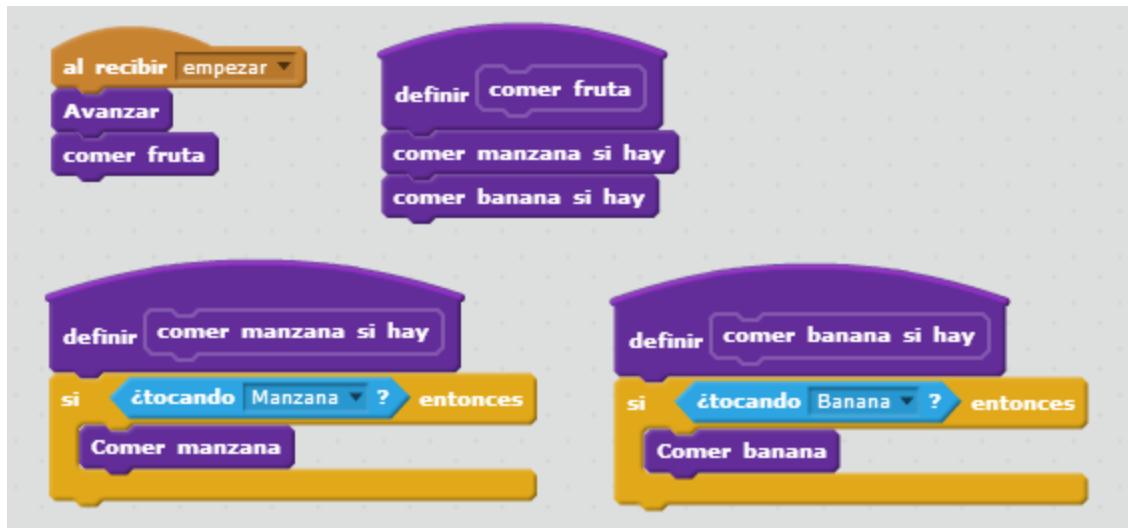
Que se lee como “si Tocando manzana entonces...”. Lo que hay que hacer es elegir qué comando ejecutar si esa condición es verdadera:



Y con esto resolvemos el problema de comer una manzana si estamos en presencia de una:



Actividad: de la misma forma definir el bloque “comer banana si hay” que come una banana en caso de que haya una. Luego, terminar de definir comer fruta y verificar que el problema se resuelve correctamente luego de presionar varias veces la banderita verde.



Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

Laberinto corto

Este problema vuelve a consistir de un escenario cambiante. El objetivo es avanzar un paso en la dirección correcta.

Nos dan dos instrucciones posibles:



Y tenemos dos casos posibles:



El siguiente paso hacia abajo



El siguiente paso hacia la derecha

No sabemos qué caso nos va a tocar, pero sí tenemos cierta información que nos puede ayudar. Debajo del ratoncito hay una flecha que indica hacia donde tenemos que movernos.

Como pista, podemos preguntar qué flecha estamos tocando, mediante alguna de las siguientes preguntas:

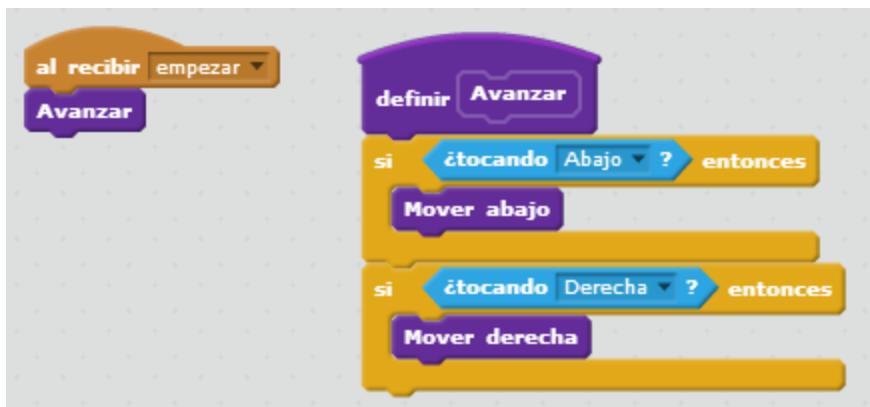


Es evidente que si estamos tocando la flecha para abajo, debemos movernos para abajo, caso contrario, tenemos que movernos hacia la derecha.

Actividad: Con esta información dejar que los alumnos intenten resolver el problema. Como pista, puede recomendarse definir un procedimiento llamado Avanzar, que sea el que sabe elegir qué paso dar haciendo estas preguntas.



Una solución es la siguiente:



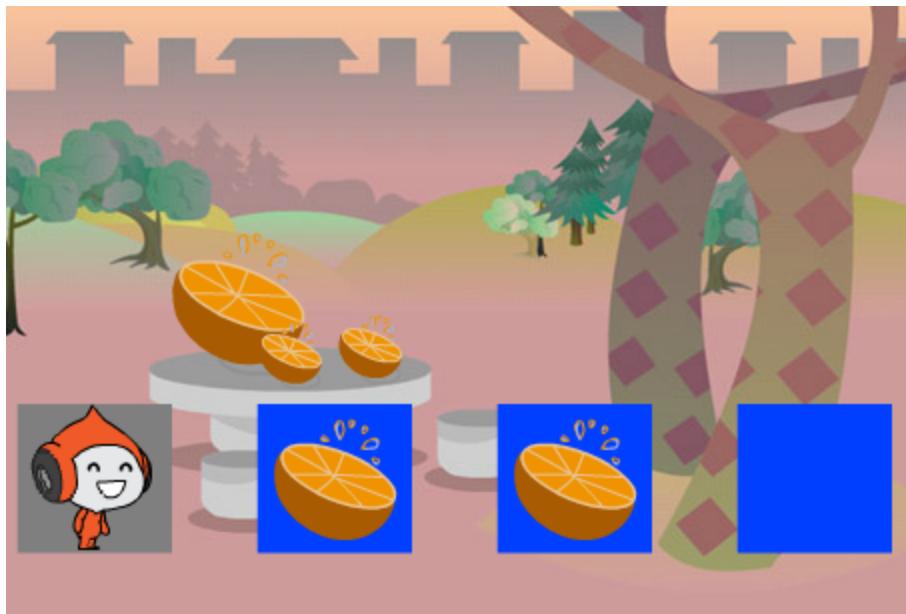
Pero también existe esta otra variante:



Esto funciona porque sólo hay dos posibles, o hay que moverse para abajo, o hay que

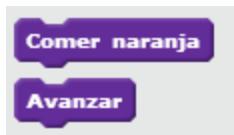
movearse para la derecha. Si la flecha que estamos tocando es para abajo, nos movemos en esa dirección, y sino, como tiene que ser una flecha para la derecha porque no hay más opciones, nos movemos para la derecha.

Tres naranjas



En esta ocasión veremos un tablero, y en algunas casillas habrá naranjas. Empezamos en una casilla gris vacía, y debemos recorrer las casillas azules y comer las naranjas que haya en las mismas.

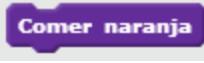
Las instrucciones son simples:



Las casillas azules siempre son 3, pero no sabemos en qué casillas hay naranjas. Sin embargo debemos comer todas las naranjas y no intentar comen naranjas donde no las hay.

Para mostrar la estructura del problema, haremos clic repetidas veces sobre la banderita verde , para mostrar que las naranjas pueden ubicarse en distintas celdas.

Si intentamos comer naranjas en donde no hay, el juego se reiniciará.

En este caso necesitamos una forma de preguntar si la casilla en donde estamos parados tiene una naranja y sólo en ese caso vamos a ejecutar  (caso contrario, no

haremos nada).

Actividad: Dejar que los alumnos intenten resolver el resto del problema por su cuenta.

La solución a este problema es la siguiente:



¿Qué pasa si intentamos hacer todo junto? Tendríamos el bloque “si … entonces” dentro del repetir, y se complica poder razonar por partes el problema. Por eso recomendamos en todas las actividades que siguen hacer pequeños procedimientos que resuelven una parte pequeña, para luego poder usarlos en la solución del problema más grande.

Lightbot recargado



Presionar la banderita verde para poder observar la estructura del problema.

En este caso la longitud de la fila es siempre la misma, algunas celdas poseen luz y otras no. La última celda nunca posee una luz y la primera puede tener o no.

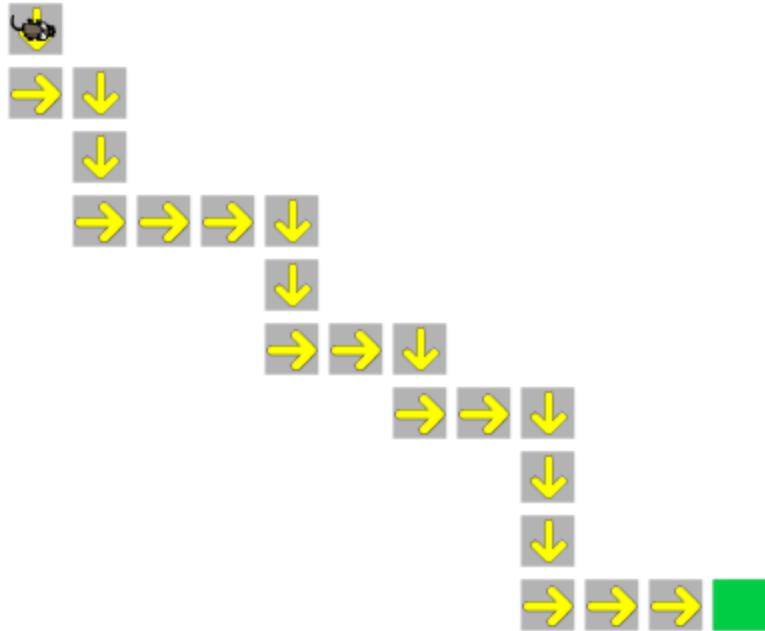
Actividad: Dejar que los alumnos resuelvan el problema.

La solución es la siguiente:



Observar que aunque sean 11 celdas, como en la última no tenemos que hacer nada, repetimos la secuencia 10 veces.

Laberinto largo



Volvemos con el problema del laberinto. Ya resolvimos este problema pero sólo en el caso en que tenemos que avanzar una casilla:



Actividad: ¿Cuántas veces tenemos que avanzar? Intentar resolver el problema sabiendo este dato.

Una solución es:



Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

Repetición condicional

Objetivos

En estas actividades vamos a aprender a recorrer secuencias de tamaño variable junto con alternativas condicionales.

Super Lightbot 1



Super Lightbot es un juego en el que intentaremos prender todas las luces, al igual que en Lightbot, pero con algunas variantes:

- No sabemos cuántas luces vamos a tener que prender
- El juego tiene una estructura: prendemos todas las luces hasta que nos topamos con algo que no es una luz.

Para mostrar la estructura que tiene la fila de celdas, apretar repetidas veces sobre la banderita verde , para mostrar que las filas poseen distinta longitud.

Pregunta a la clase: ¿Por qué no podemos usar una repetición simple para resolver el problema? ¿Podemos resolverlo con lo que ya sabemos o necesitamos algo nuevo?

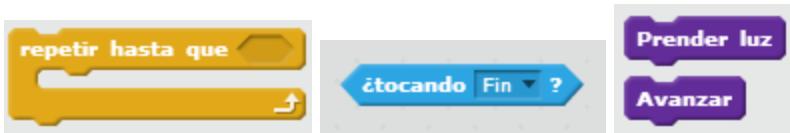
El problema no puede ser resuelto con una repetición simple sencillamente porque no sabemos cuánto tenemos que repetir la tarea:



Si pudiéramos saber el tamaño de la fila cada vez, el problema estaría resuelto con una repetición como esta, pero no contamos con esa información.

Pero, como podemos ver, en este juego podemos prender luces *hasta* que nos topemos con un cuadrado gris. Usaremos esta condición para cortar con la repetición.

Actividad: Utilizar estos bloques para resolver el problema



Actividad: ¿Qué ventajas tiene esta solución con respecto a lo que hacíamos antes para Lightbot, comer manzanas, comer bananas, etc.? ¿Hubiese sido fácil resolver todos los problemas anteriores si contábamos con esta herramienta?

Solución:



Lo que acabamos de usar se llama repetición condicional, y consta de dos partes:

- Una condición que nos sirve para dejar de repetir algo
- La tarea que queremos repetir hasta que la condición se cumpla

A la forma en la que resolvimos el problema la llamaremos **recorrido**, porque justamente lo que hicimos es recorrer todas las celdas procesando (aplicando una tarea) una por una. Los recorridos son muy útiles para cuando un problema posee cierta estructura. Por ejemplo, en este caso la estructura consta de una fila de celdas contiguas. Esta estructura claramente nos está indicando que el problema puede resolverse con un recorrido, siempre y cuando contemos con:

- Una forma de poder cortar con el recorrido (saber el número de celdas o tener identificada la última celda de alguna manera)
- Contar con la acción que queremos aplicar a cada celda (en este caso es Comer Naranja)
- Una forma de poder pasar al “siguiente elemento” (que en este caso es Avanzar), que haga que en algún momento la condición sobre la que estemos trabajando cambie (pasar de no cumplirse, a cumplirse).

El primer y último ítem son extremadamente importantes, porque son requisito para poder

usar la repetición condicional. Si no contamos con alguno de los dos no podremos trabajar correctamente.

Nota para el profesor: Una variante de “repetir hasta que” es “mientras” (while, en inglés), que es la versión utilizada en la mayoría de los lenguajes de programación. Ambas son formas de repeticiones condicionales, sólo que varían en que la primera versión sigue repitiendo algo HASTA que la condición se cumpla, y la otra variante repite algo MIENTRAS la condición se cumpla, o en otras palabras, hasta que deje de cumplirse.

Super Lightbot 2



Presionar la banderita verde para poder observar la estructura del problema.

En este caso tenemos que resolver un problema similar, pero con la siguiente variante: no toda celda antes del final de la fila es una luz que podemos prender.

Si intentamos prender una celda que no es azul (que es gris) nos dará el siguiente error:



En este caso necesitamos una forma de preguntar si la celda que estamos pisando es azul, y

sólo en ese caso utilizar (caso contrario, no hacer nada).

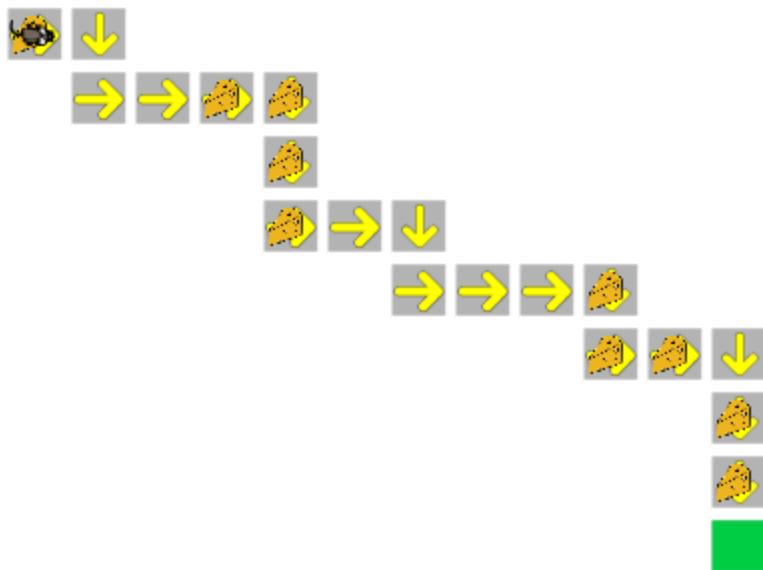
Actividad: Dejar que los alumnos intenten resolver el problema por su cuenta. Si no se les ocurre usar el bloque “si … entonces”, darles como pista que podrían usar eso. La condición sobre la que tienen que preguntar es si .

La solución a este problema es la siguiente:



Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

Laberinto con queso



Volvemos a encontrarnos en un laberinto, pero con las siguientes modificaciones:

- Ahora debemos comer un pedazo de queso sólo en las celdas que posean uno.
- El largo del laberinto es desconocido.

Actividad: definir un programa que resuelva este problema.

Una solución es la siguiente:



Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

El Detective Chaparro



Este es un juego en el que tenemos que descubrir al sospechoso. La forma en que se resuelve es pasar por cada persona interregándola, desde la primera hasta encontrar al sospechoso. Siempre existe el sospechoso, que suplanta la identidad de alguno de los personajes que vemos en pantalla. No sabemos a quién suplanta y por eso debemos recorrer a cada persona, una por una.

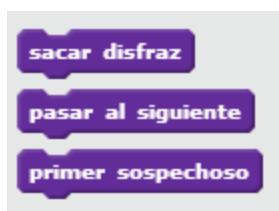
Presionar varias veces la banderita verde para ver qué se nos muestra.

Podemos ver que:

- El detective empieza desde cualquier posición
- La cantidad de sospechosos es siempre la misma

En los juegos anteriores, siempre empezábamos “desde el principio”, pero en este caso debemos ir al primer sospechoso, dado que si avanzamos desde donde estamos parados, vamos a perder a algunos sospechosos que podrían ser el villano.

Contamos con las siguientes instrucciones:



Las últimas dos son claras; la primera lo que hace es fijarse si el villano es el sospechoso ubicado en la posición en donde estamos parados.

Actividad: Sabiendo que siempre hay 7 sospechosos, pero tenemos que cortar el juego apenas encontramos al villano, ¿podemos usar una repetición simple?

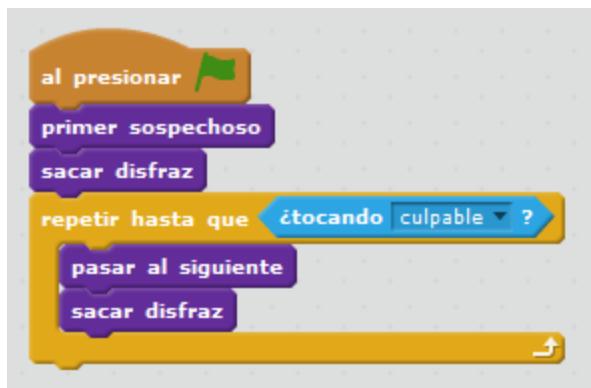
No podemos usar una repetición simple porque aunque sean 7 los sospechosos, el villano lo podemos encontrar ANTES de pasar por los 7. Si decimos:



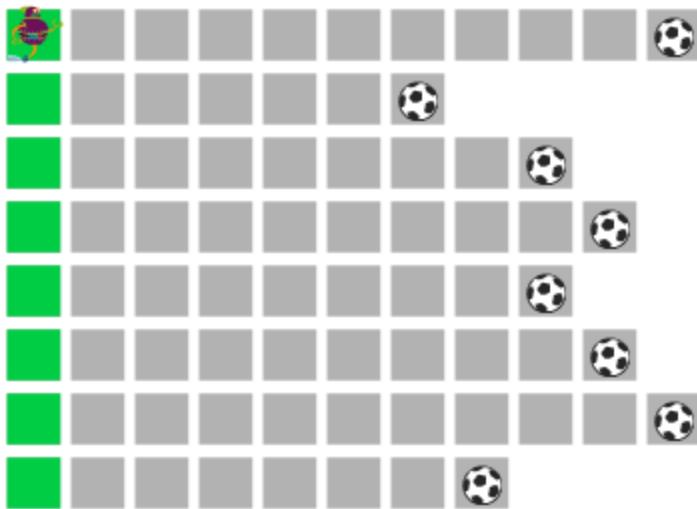
Estamos indicando que vamos a repetir algo exactamente 7 veces. Pero en este caso, podría ser necesario repetir la tarea 7 veces, pero también podría ser necesario hacerlo menos veces si al villano lo encontramos antes del final.

Actividad: Dejar que los alumnos resuelvan la actividad.

La solución es la siguiente:



Fútbol para robots



Esta actividad presenta una solución más compleja que las correspondientes a las actividades anteriores. Separando en subtareas el problema resulta abordable, en caso contrario es demasiado complejo.

Las primitivas son:



Veremos 8 filas de tamaño variable, que debemos recorrer hasta alcanzar la pelota. Una vez ahí debemos ejecutar el comando “patear pelota”. Luego, debemos volver al inicio de la fila ya que sólo es posible pasar a la siguiente fila si estamos parados en una casilla verde.

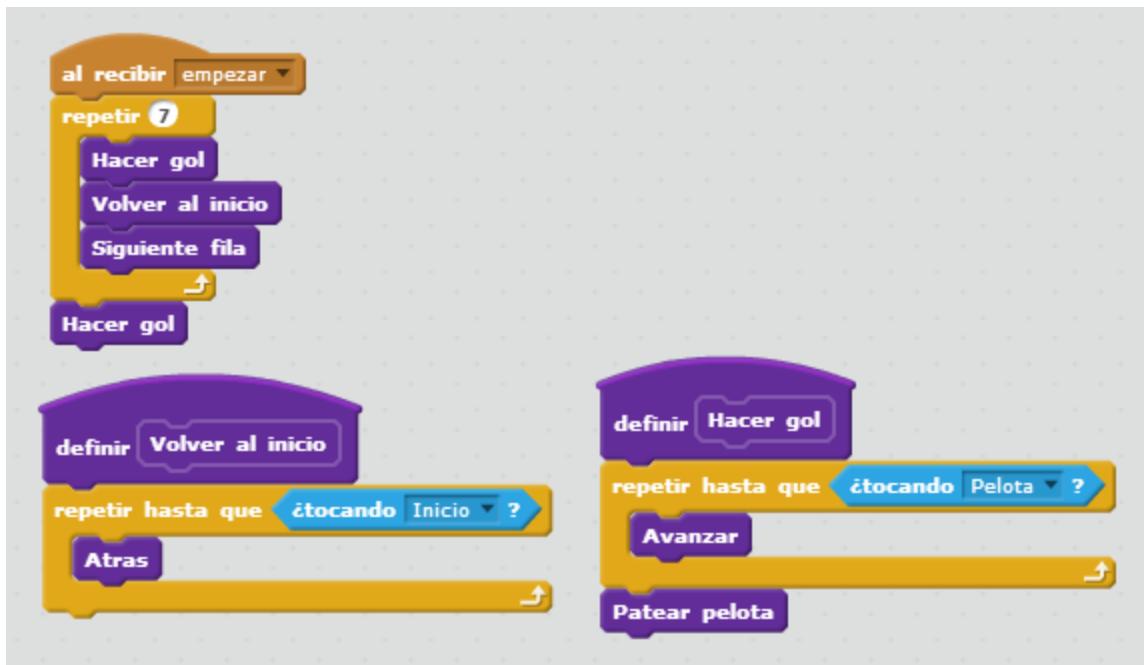
Para resolver cómodamente el problema debemos subdividirlo en problemas más pequeños. Al menos destacamos dos subtareas:

- “Hacer gol”, que dirige el robot hasta la pelota y la patea
- “Volver al inicio”, que retrocede lo necesario para volver a la casilla verde

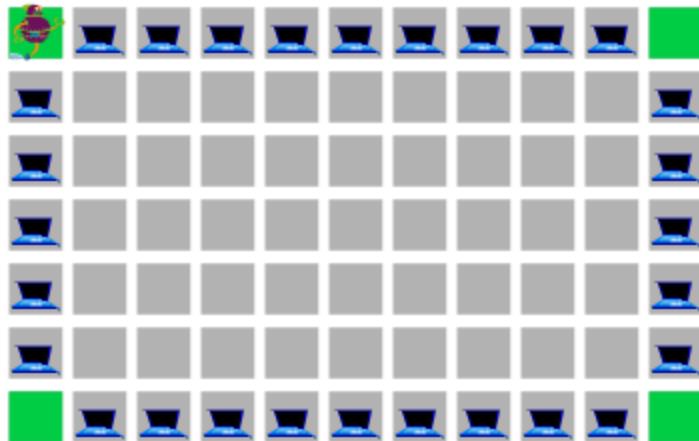
Con estas subtareas podemos definir el programa de forma más simple. Si no pensamos antes en subtareas el problema se puede dificultar.

Actividad: Luego de esta discusión dejar que los alumnos definen estas subtareas y, a continuación, el programa para patear todas las pelotas. Hacer notar a los alumnos que la tareas de hacer un gol es similar a la de los problemas anteriores.

Possible solución:

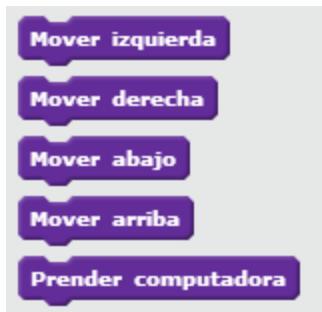


Prendiendo las compus



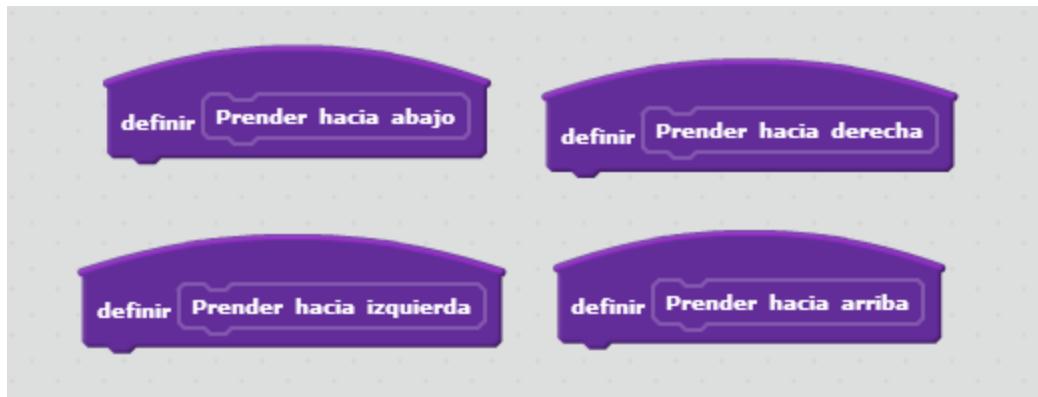
El escenario es un rectángulo de computadoras de tamaño variable, es decir, no conocemos el ancho o el alto del rectángulo, ya que cambia cada vez que presionamos la banderita verde.

Nos dan el siguiente conjunto de instrucciones.

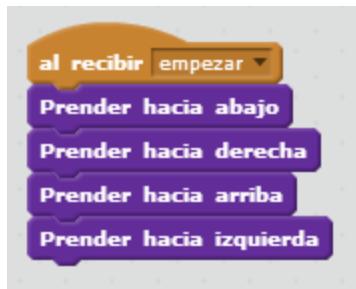


Actividad: Pensar entre todos qué subtareas podemos definir para poder resolver el problema.

Una buena división en subtareas debería llevar una solución como la siguiente:



Cada una de estas subtareas resuelve un lado específico del rectángulo, y nos permiten definir el siguiente programa:



De esta forma prenderíamos todas las computadoras del rectángulo.

Actividad: Definir el código de las subtareas anteriores para que el programa prenda todas las computadoras.

La solución es:



Observar que, como siempre empezamos en una esquina, primero tenemos que movernos hacia un paso, para salir de la esquina actual y alcanzar la siguiente.

El mono que sabe contar



Esta actividad pretende explotar al máximo lo aprendido hasta ahora. Como en las actividades anteriores, debe hacerse clic sobre la banderita verde varias veces para poder ver qué patrón sigue el escenario.

Las instrucciones que nos proveen son las siguientes:



El objetivo es que el mono pase por cada casilla, contando cada manzana y cada banana por la que pasa, pero existen una serie de reglas, que se describen a continuación:

- El mono sólo puede ejecutar “contar manzana” si está parado sobre una manzana

- El mono sólo puede ejecutar “contar banana” si está parado sobre una banana
- El mono sólo puede avanzar o volver hacia atrás si existe una celda a la cual moverse.
- Sólo se puede pasar a la siguiente fila si el mono está parado en una casilla verde.

Resolveremos el problema paso a paso, primero haciéndolo sólo para una de las filas y luego generalizando la solución para todas.

Actividad: Pensar entre todos qué subtareas haría falta para resolver el problema.

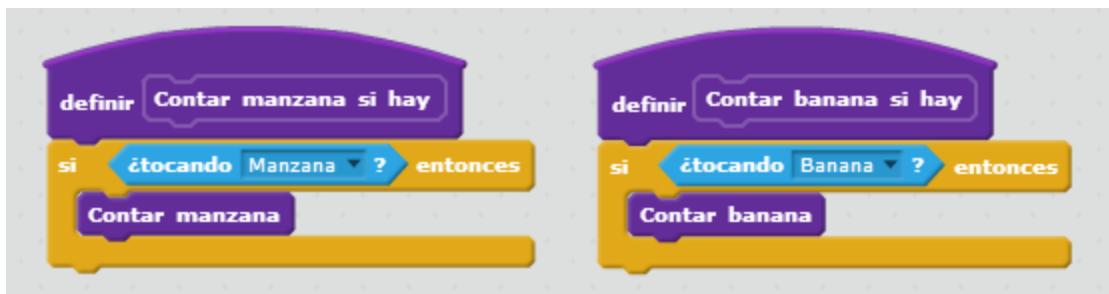
Las primeras subtareas que debemos intentar resolver son:

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

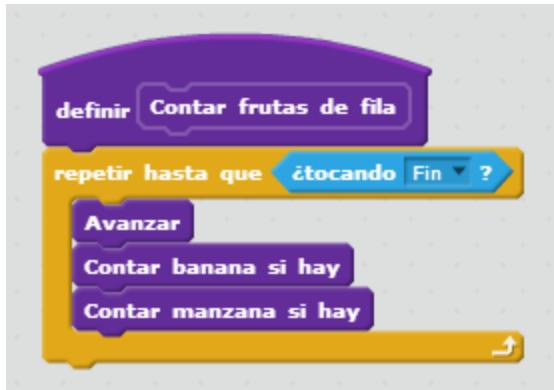
Actividad: Dejar que los alumnos intenten definir cada una de estas subtareas.

Si tenemos estas cuatro subtareas resueltas, el resto del juego es muy fácil. Empezaremos a definir una por una.

Primero comenzamos por lo que haríamos en una casilla en donde puede haber fruta:



La tarea de recorrer toda la fila también es bastante fácil de resolver, dado que debemos repetir las dos subtareas que acabamos de definir y avanzar hacia delante, hasta toparnos con el color violeta:



Volver a la casilla verde también es sencillo:



Actividad: Definir el programa que resuelve el problema teniendo estos procedimientos definidos

Ahora lo que debemos observar es que la cantidad de filas es siempre la misma, que es 5. Lo primero que surge pensar es que deberíamos usar un “repetir 5”, pero hay que tener en cuenta un detalle: si bien contamos frutas en 5 filas cambiamos de fila solo 4 veces, por lo que la última hilera debe ser tratado en forma diferenciada a las anteriores. Entonces, en realidad 4 veces haremos algo como “Recorrer fila, volver a casilla verde, siguiente fila” y en la última sólo haremos “Recorrer fila”. La definición de este procedimiento queda así:



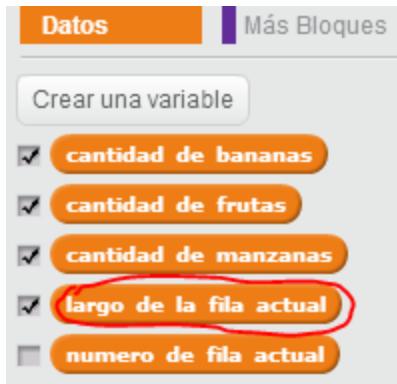
El mono cuenta de nuevo



Esta es una variante de la actividad anterior. El objetivo sigue siendo el mismo y las reglas también, pero existe un problema, no contamos con que la última celda sea de color violeta.

Actividad: Si la última celda es igual a todas las de la fila, no podemos utilizar un “repetir hasta que”, ¡¿porque cuál sería la condición para deternernos?! Por mucho que pensemos en realidad no existe una condición que nos permita frenar. Si no podemos utilizar una repetición condicional, ¿qué dato necesitamos para poder utilizar una repetición simple?

Haya surgido o no la respuesta correcta, debemos ir a la parte de datos:



Ese bloque llamado “largo de la fila actual”, es justamente la cantidad de casillas que posee la fila en donde estamos parados. Debemos tener en cuenta que como la primera casilla no se cuenta, lo que indica el número es la cantidad de celdas hacia la derecha del mono.

Veamos un ejemplo.



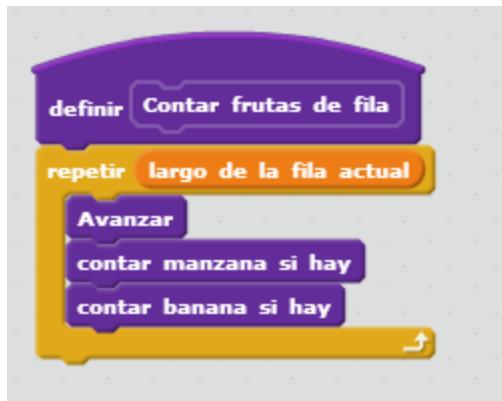
En este caso “largo de la fila actual” vale 5, porque el mono está parado sobre una fila que posee 6 casillas en total, contando la casilla verde.

Este dato lo podemos usar junto al bloque repetir, y con eso podemos recorrer la fila aún sin tener la condición de corte que necesitábamos.



Actividad: Dejar que los alumnos terminen de resolver la actividad.

La solución de esta actividad es igual a la anterior, sólo que “Contar frutas de fila” se resuelve como:



Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

El superviaje



Somos superman, y tenemos que llegar a alguna ciudad.

Contamos con una primitiva:



Cuando ejecutamos este comando la distancia se acorta en una unidad:



La ciudad está a una distancia de entre 50 y 200 KM, pero no sabemos en cada caso cuánto exactamente. La distancia cambia cada vez que empezamos el problema (observar esto presionando varias veces la banderita verde).

También nos proveen como dato la distancia restante hasta la ciudad. Este dato lo podemos encontrar en la categoría datos:



Este dato es un número, como lo era el “largo de la fila actual” en la actividad anterior.

Actividad: Sabiendo esto, permitir a los chicos resolver la actividad como les sea posible.

Esta actividad puede resolverse de dos maneras, con una repetición simple y también con una repetición condicional:



Solución con repetición simple



Solución con repetición condicional

Actividad: Es interesante discutir cuál es mejor. Preguntar a los chicos con qué versión se quedarían y por qué les parecería mejor.

Siempre que sabemos cuántas veces tenemos que repetir algo, lo correcto es usar una repetición simple, porque de lo contrario tenemos que pensar cómo armar la condición que permite cortar la otra variante.

Cierre

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a pfactorovich@fundacionsadosky.org.ar

Como cierre sería interesante repasar cada tipo de repetición y entender qué resuelve una y qué resuelve la otra, y cuando es posible resolver el problema con ambas qué conviene elegir.

Parametrización de soluciones

Canciones y estribillos

El objetivo de esta actividad es mostrar cómo los estribillos de las canciones representan procedimientos. Luego aprovecharemos esto para presentar un tema nuevo, llamado parametrización.

Como sabrán un estribillo es una pieza musical que se repite a menudo. ¿Nos suena a algo? ¿Qué hacíamos en Lighbot cuando veíamos que un patrón se repetía? ¿Qué hicimos en el ejercicio de prender dos diagonales de luces iguales? ¿Qué hicimos en el ejercicio de tomar 4 columnas de estrellas?

Vimos que los procedimientos cumplen dos funciones:

- Ayudar a separar nuestras ideas en distintas subtareas más simples para poder plantear una solución más compleja (metodología que se conoce como división en subtareas)
- Reunir una secuencia de pasos bajo un nombre que nos permita ejecutar toda esa secuencia con sólo utilizar el nombre que hayamos asignado

En esta actividad nos vamos a concentrar más en el último ítem, ya que el objetivo de las canciones es ahorrarse trabajo definiendo un estribillo que es posible reutilizar en distintas partes.

Veamos un ejemplo con la canción de María Elena Walsh, “El reino del revés”.

Primero ver el siguiente [link](#) para ver la canción entera.

Seguir estos pasos:

- Marcar el párrafo entero que se repite (puede haber partes repetidas dentro de los párrafos pero nos importan los párrafos enteros dentro de la canción)
- Extraer lo que se repite, ubicarlo en otra parte y ponerle de título *[Estribillo]*
- En cada lugar en donde aparece el estribillo entero, reemplazarlo por el título *[Estribillo]*

La solución debe quedar así:

[Estribillo]

Vamos a ver cómo es
el Reino del Revés.

Canción:

Me dijeron que en el Reino del Revés
nadie baila con los pies,
que un ladrón es vigilante y otro es juez
y que dos y dos son tres.

[Estribillo]

Me dijeron que en el Reino del Revés
cabe un oso en una nuez,
que usan barbas y bigotes los bebés
y que un año dura un mes.

[Estribillo]

Me dijeron que en el Reino del Revés
hay un perro pekinés
que se cae para arriba y una vez
no pudo bajar después.

[Estribillo]

Me dijeron que en el Reino del Revés
un señor llamado Andrés
tiene 1.530 chimpancés
que si miras no los ves.

[Estribillo]

Me dijeron que en el Reino del Revés
una araña y un ciempiés
van montados al palacio del marqués
en caballos de ajedrez.

[Estribillo] x 2

Al terminar hay que observar dos cosas:

- ¡El estribillo se repetía muchas veces! Eso significa que si no lo reducimos a un título tenemos que escribir todo el párrafo una y otra vez.
- Al final el estribillo se repite dos veces, así que tranquilamente podemos usar el concepto de repetición que ya conocemos y decir “x 2”, dando a entender que se repite dos veces lo mismo.

Observar que cuando llega el momento de cantar el estribillo, no decimos literalmente “estribillo”, sino que vamos a la definición del estribillo y cantamos la letra que lo forma.

¿A qué se parece todo esto? Como decíamos al principio, la metodología es la misma que con programas, vemos lo que se repite, le ponemos un nombre a lo que se repite y de ahí en más usamos ese nombre y no toda la secuencia.

Veamos otra clase de ejemplo. Hay canciones en donde existe una parte que se repite, pero cambia ligeramente en algo.

Actividad: En la famosa canción “un elefante se balanceaba…”, ¿qué es lo que se repite y qué es lo que varía?

Seguramente todos coincidan en que la canción es siempre la misma salvo por el número de elefantes.

Podemos abstraer eso de la siguiente manera. La canción empieza para un elefante, y a partir de ahí recién se repite lo mismo una y otra vez. Entonces empezamos escribiendo:

“Un elefante se balanceaba sobre la tela de una araña, como veía que resistía fueron a llamar a otro elefante”

El resto es así:

“2 elefantes se balanceaban sobre la tela de una araña, como veía que resistía fueron a llamar a otro elefante”

“3 elefantes se balanceaban sobre la tela de una araña, como veía que resistía fueron a llamar a otro elefante”

“4 elefantes se balanceaban sobre la tela de una araña, como veía que resistía fueron a llamar a otro elefante”

“5 elefantes se balanceaban sobre la tela de una araña, como veía que resistía fueron a llamar a otro elefante”

¿Ven lo que se repite? Todo menos un número. Lo que podemos hacer es primero marcar con un hueco lo que cambia cada vez:

“_ elefantes se balanceaban sobre la tela de una araña, como veía que resistía fueron a llamar a otro elefante”

Esto representa un esquema o esqueleto que nos permite decir “elegí el número que quieras para usar en ese hueco, y vas a tener la canción para la parte en donde se dice esa cantidad de elefantes”

Si viene alguien que no conoce la canción, tenemos que indicarle de alguna manera que ese hueco se llena exactamente con un número. Para eso, en lugar en lugar de poner una línea, vamos a poner un nombre subrayado. En este caso, ¿qué nombre puede recibir el hueco? Podemos llamarlo “número”, “cantidad”, etc.

Entonces finalmente conseguimos algo así:

“cantidad elefantes se balanceaban sobre la tela de una araña, como veía que resistía fueron a llamar a otro elefante”

La canción sería algo como:

[Estríbillo cantidad]

“cantidad elefantes se balanceaban sobre la tela de una araña, como veía que resistía fueron a llamar a otro elefante”

Canción:

“Un elefante se balanceaba sobre la tela de una araña, como veía que resistía fueron a llamar a otro elefante”

[Estríbillo 2]

[Estríbillo 3]

[Estríbillo 4]

... (así para todos los números que siguen)

Lo que llamamos informalmente “hueco”, en programación se conoce como **parámetro**, y al proceso de agregar un parámetro a un procedimiento. se lo conoce como **parametrización**.

Otra canción que algunos pueden conocer es la que dice “El viejo Mc Donald tenía una granja...”. A los que no la conocen pueden ver el siguiente [video](#).

La letra de la canción es la siguiente:

El viejo Mc Donald 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 Mc Donald tenía una granja, IA IA IO

El viejo Mc Donald 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 acá
por todos lados oink oink
El viejo Mc Donald tenía una granja, IA IA IO

El viejo Mc Donald 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 Mc Donald tenía una granja, IA IA IO

El viejo Mc Donald tenía una granja, IA IA IO
Y en su granja había un gallito, IA IA IO
con un kikiri de aquí, un kikiri de allá
por todos lados kikiri kikiri
El viejo Mc Donald tenía una granja, IA IA IO

Observar que tiene 4 partes que se repiten casi iguales.

Actividad: Discutir qué se repite y qué varía en esta canción.

La conclusión a la que deben arribar es que la canción se repite 4 veces igual salvo por el animal y el sonido que hace.

Actividad: Modificar la letra de la siguiente manera:

- Marcar cuáles serán los parámetros dentro del estribillo y poner un nombre a cada tipo de de parámetro. En otras palabras hay un parámetro para el animal y para el sonido que produce el animal (intentar que los chicos piensen ellos mismos en el nombre de cada parámetro, pero si tienen complicaciones ayudarlos).
- Armar un estribillo con los parámetros identificados
- Formular la canción como 4 llamados estribillos en los que le pasamos en cada llamado qué animal y qué sonido deben usarse.

La solución es la siguiente:

[Estríbillo animal sonido]

El viejo Mc Donald tenía una granja, IA IA IO
Y en su granja tenía un/una animal, IA IA IO
Con un sonido por aquí, con un sonido por allá,
en todos lados sonido sonido
El viejo Mc Donald tenía una granja, IA IA IO

Canción:

[Estríbillo vaca muu]

[Estríbillo cerdito oink]

[Estríbillo caballo eee]

[Estríbillo gallito kikiri]

Cierre

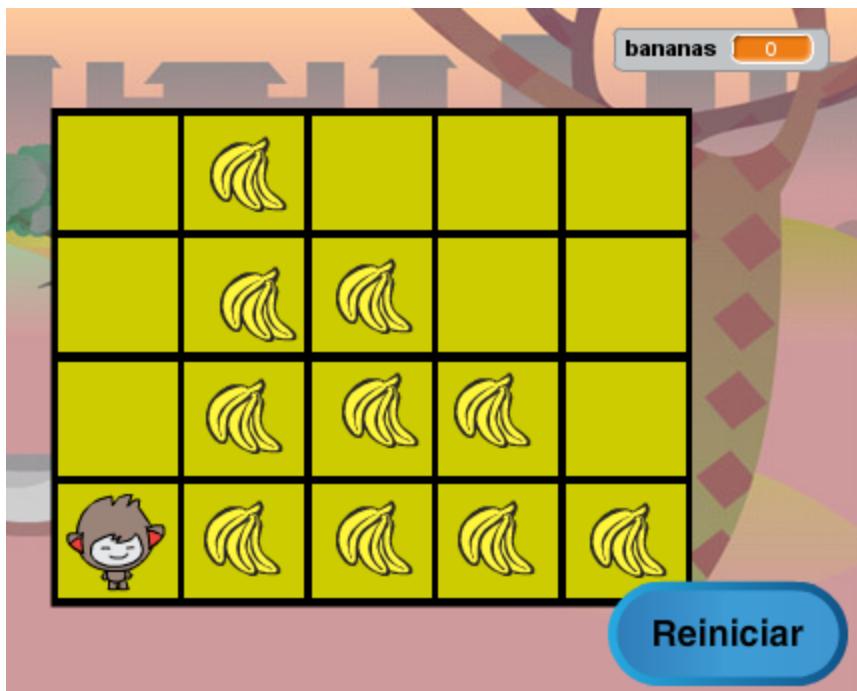
En esta actividad vimos una herramienta muy poderosa utilizada en programación para poder definir procedimientos que difieren en un dato, que nos permite definir como parámetro el dato que varía de un llamado a otro, y permitirnos recién en el llamado elegir qué parámetro queremos en ese momento.

En las próximas actividades veremos cómo este concepto se aplica específicamente a la programación.

[El planeta de Nano](#)

Desarrollo

Presentamos la actividad “El planeta de nano”.



Al igual que con la actividad de comer manzanas, en este caso hay que comer bananas. Pero las instrucciones en este caso son distintas:

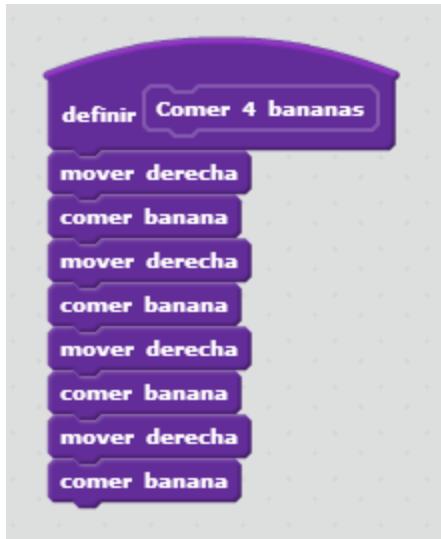


No tenemos instrucciones para movernos hacia abajo o la izquierda. El bloque “Volver al borde izquierdo” simplemente nos regresa a la primer celda desde el borde izquierdo, respetando la fila en la que nos encontramos.

Actividad: Dejar que los alumnos intenten resolver el problema.

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

Si tuviésemos que contar a una persona una forma de resolver el problema, diríamos que hay que comer distintas filas de bananas. Entonces, un procedimiento, para empezar a comer las primeras 4, podría ser “Comer 4 bananas”:

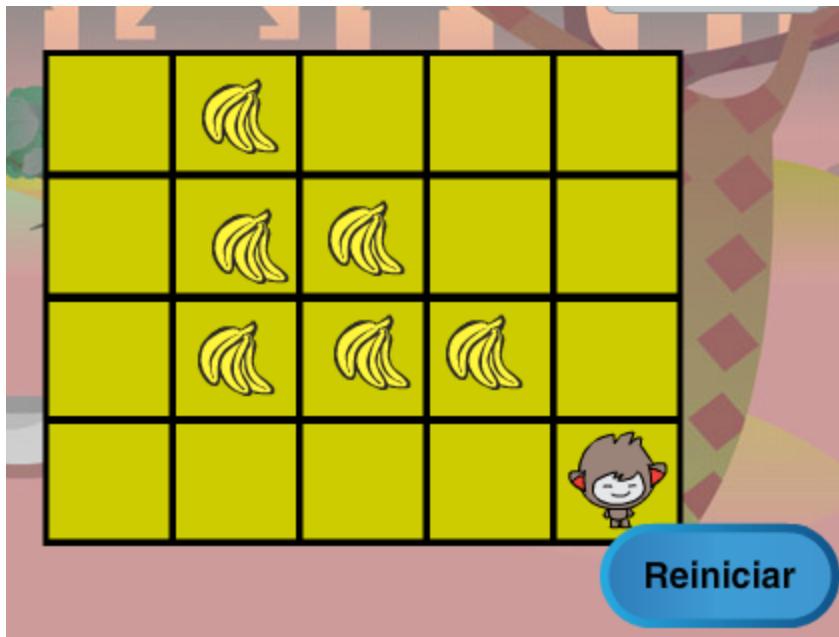


O mejor aún, usando un “repetir”:

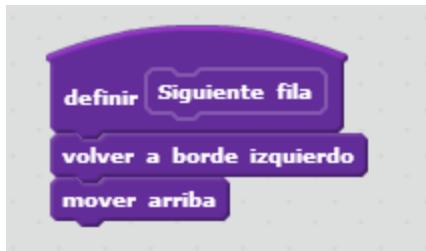


Esto nos deja el tablero en el siguiente estado:





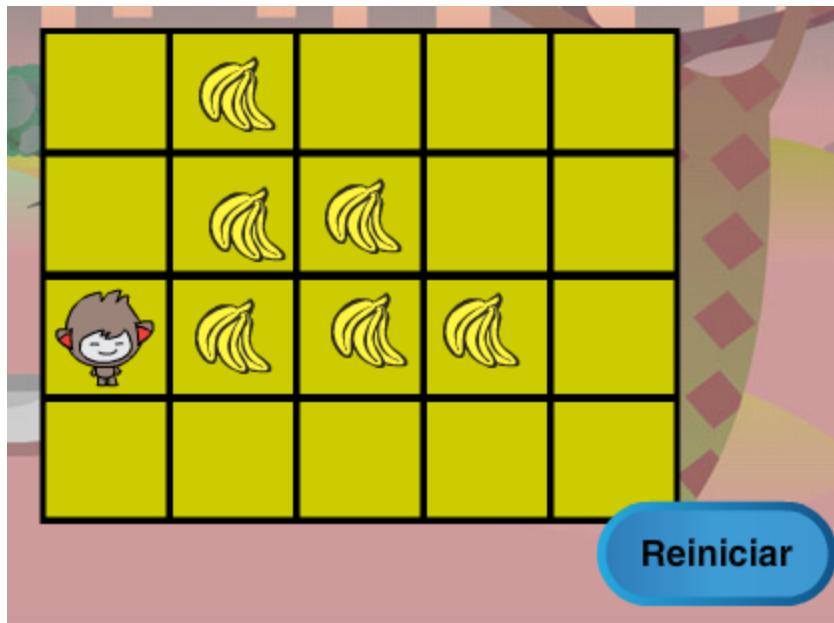
Ahora lo que hace falta es un procedimiento que nos permita pasar a la siguiente fila de bananas:



Y el programa nos queda así:



Con el siguiente tablero:



Es fácil ver que debemos hacer otros procedimientos similares para las demás filas:



Son muy similares a “Comer 4 bananas”, salvo por el número de bananas que come cada uno.

En resumen nos queda el siguiente programa final:



Podríamos haber comido las bananas como quisiéramos, pero esta solución es fácil de explicar y de leer, sobre todo para nosotros que escribimos la solución y para otro que quiera leerla como nosotros. A la computadora le es indistinto, siempre y cuando todas las bananas hayan desaparecido.

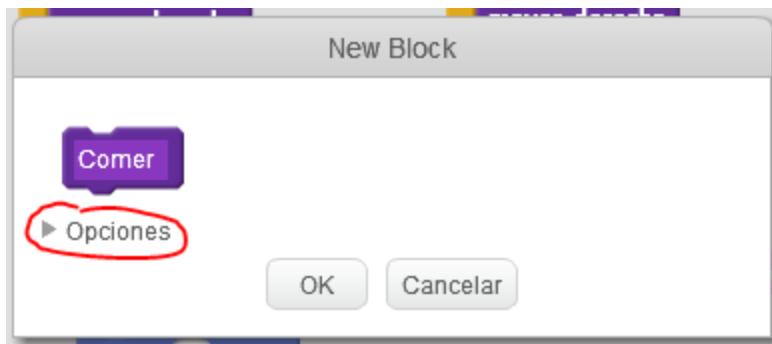
Lo que es interesante ahora es pensar una solución mejor, dado que esta tiene un problema: repite muchas veces algo **muy similar** que sólo cambia en un **valor**.

Sería conveniente que en lugar de crear 4 procedimientos distintos que sólo cambian en un número, hagamos lo siguiente:

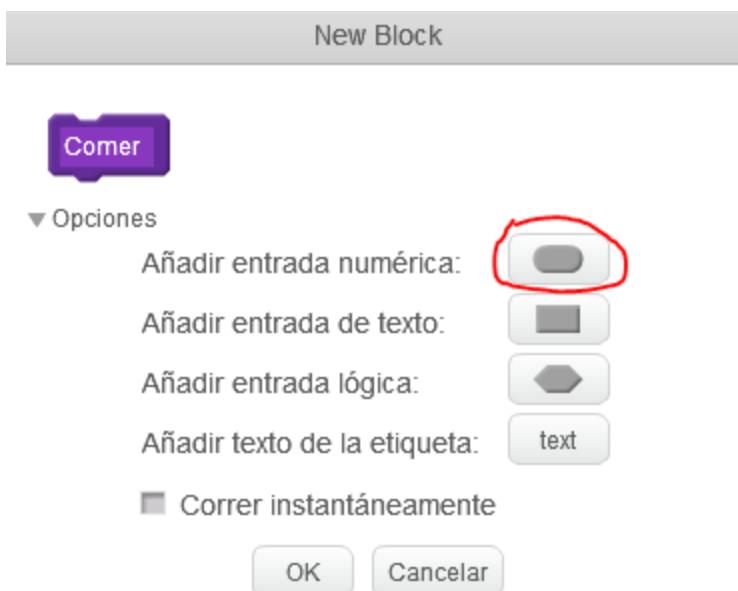


Lo que queremos hacer ahora, es indicar nosotros mismos que nuestro comando va a recibir un dato, y que ese dato va a decir, en el caso de las bananas, cuántas de estas hay que comer.

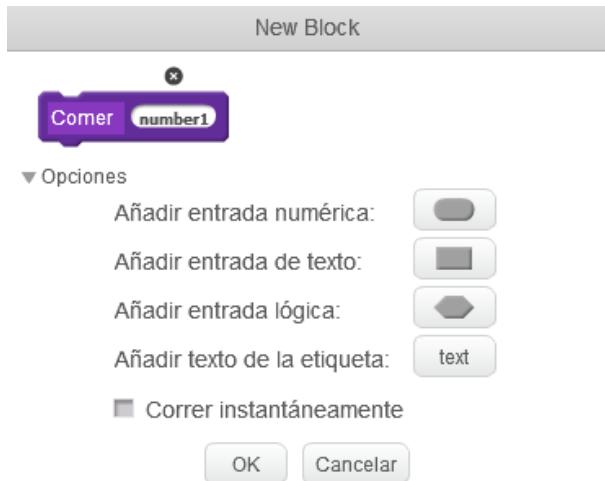
¿Cómo hacemos esto? Primero debemos crear un **parámetro**. Para ellos crearemos un nuevo bloque e iremos a “opciones”:



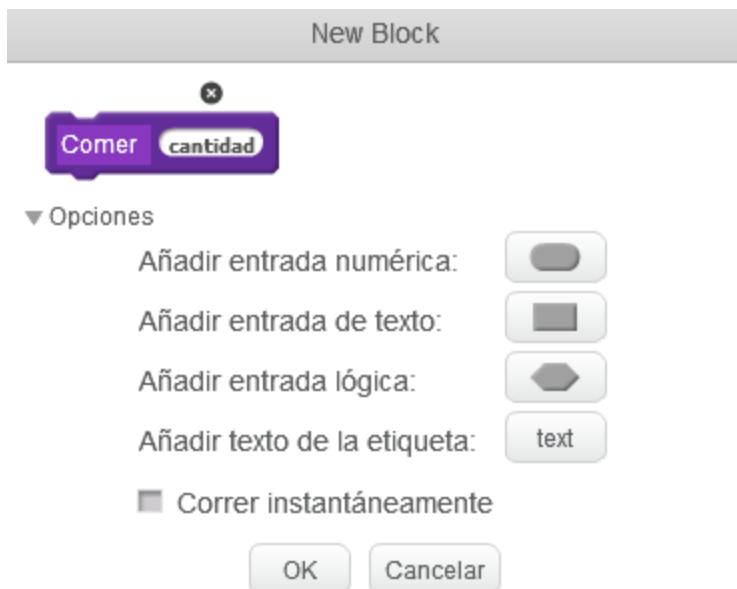
Luego de escribir “Comer” presionamos el botón que dice “Añadir entrada numérica”:



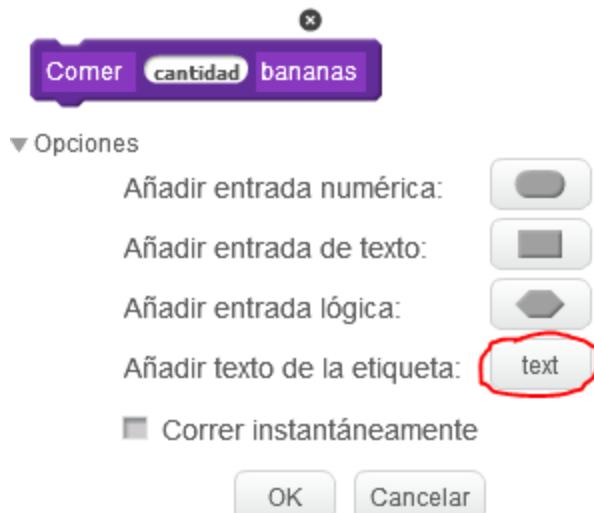
Al presionar ese botón nos agregará un hueco en blanco con un nombre por defecto:



A continuación le daremos nuestro propio nombre al hueco que nos creó, que en este caso puede llamarse “cantidad”, para indicar que va a representar la cantidad de bananas a comer:



Como retoque final podemos seguir añadiendo texto a continuación del hueco a través del botón “Añadir texto de la etiqueta:”



Presionamos OK y veremos lo siguiente:



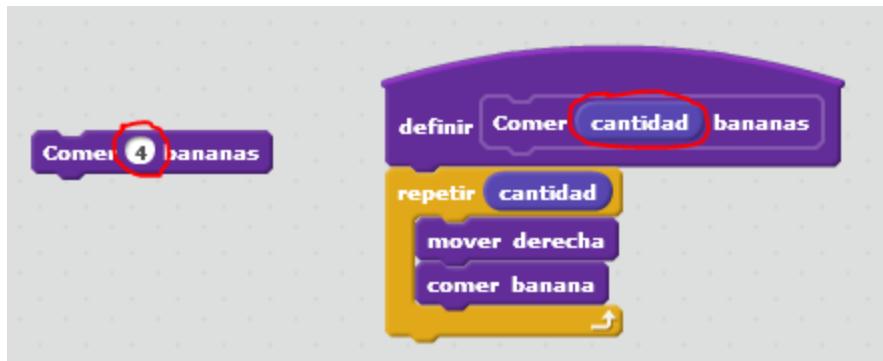
El bloque azul dentro de la definición, llamado “cantidad”, es el parámetro del cual hablábamos antes de empezar a crear este nuevo bloque. Definiremos este procedimiento genérico de la siguiente manera:



Observar que arrastramos el bloque azul y lo ubicamos en el lugar que debe ocupar dentro del procedimiento. En lugar de escribir un número concreto de veces que se va a repetir la secuencia de avanzar y comer banana, con esto estamos indicando que esa cantidad va a

ser arbitraria y dependerá de lo que ingrese el usuario.

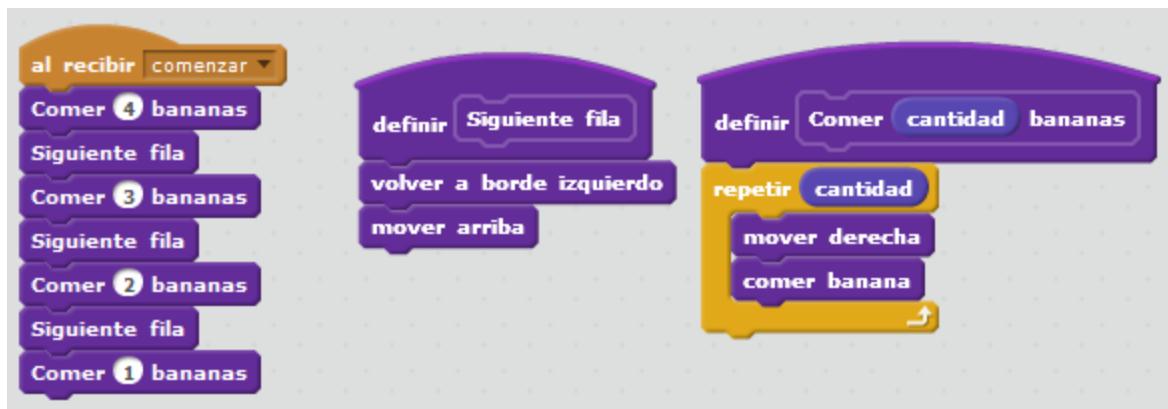
De esta forma los parámetros nos permiten relacionar lo que el usuario ingresa en el hueco en blanco con el sitio que debe ocupar al momento de utilizar ese valor:



Cuando nosotros escribimos finalmente “4”, por ejemplo, ese número concreto va a reemplazar al parámetro en la definición del bloque, y lo va a hacer en los todos los lugares en donde aparezca el parámetro, que en este caso es “cantidad”.

Este nuevo procedimiento al ser genérico nos permite eliminar a todos los demás que eran instancias particulares de este, lo que da como resultado una reducción el código que tenemos escrito.

Actividad: Hacer que este procedimiento nuevo sirva para comer las 4 diferentes filas de bananas.



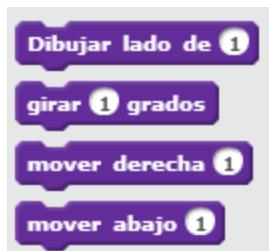
Dibujando figuras

Presentación

Para las actividades de este apartado utilizaremos el [siguiente escenario](#).

Lo que haremos en las siguientes actividades es crear figuras, casi todos polígonos regulares (no hace falta usar esta terminología para trabajar). A los alumnos podemos contarles que dibujaremos distintas figuras, como cuadrados, triángulos, y muchas más que no se imaginan.

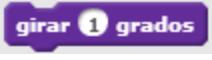
Disponemos de las siguientes primitivas:



Para dibujar un lado simplemente usaremos el bloque . Este bloque nos deja indicar cuál será la longitud del lado a dibujar. Por ejemplo:



El robot siempre que iniciamos el programa comienza desde la misma posición, y si se sale de la pantalla porque la figura que intentamos hacer es muy grande, se frena y dice que no puede continuar.

Además utilizaremos el bloque  . Este bloque nos permite indicar cuánto queremos girar el robot. Si dibujamos un lado y luego giramos el robot, el próximo lado que dibujará estará inclinado tanto como hayamos girado.

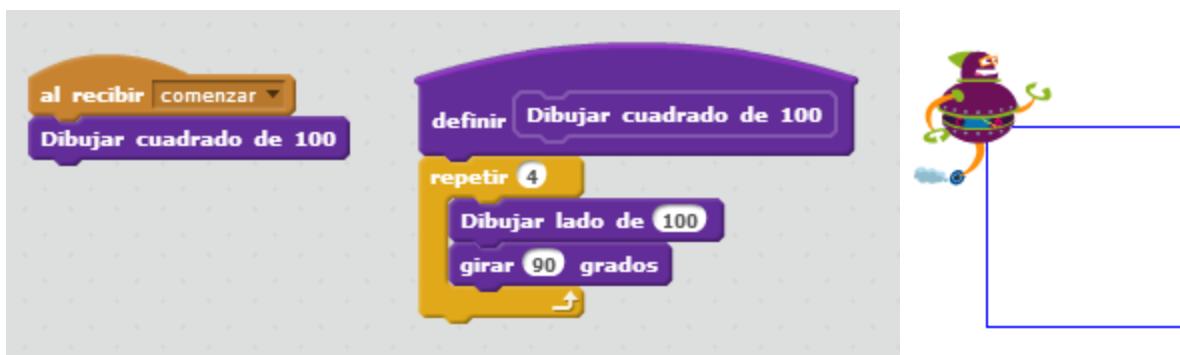
Actividad: pensar entre todos cómo sería posible construir un cuadrado utilizando estos dos procedimientos. Hacer ver a los alumnos que los cuadrados poseen ángulos de 90 grados, es decir, ángulos rectos. Y mostrar que tienen 4 lados y 4 ángulos.

Lleguen o no lleguen los alumnos a una solución, mostramos cómo dibujar un cuadrado para una longitud de 100 por lado:

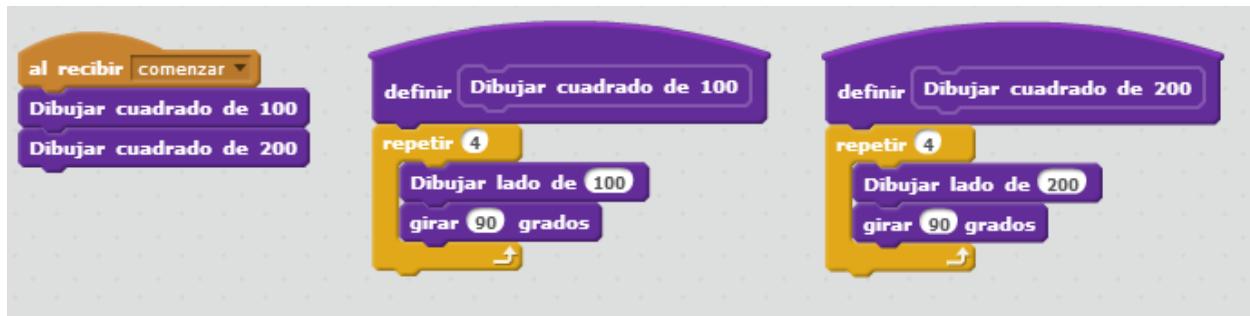


No es muy importante si los alumnos no están familiarizados con el uso de grados para medir ángulos, simplemente les decimos que la cantidad a ingresar es 90 para crear una esquina de un cuadrado.

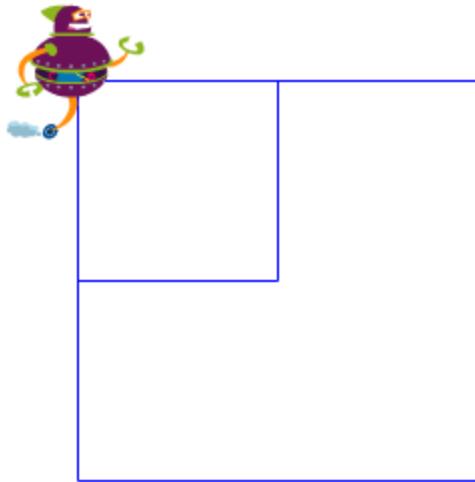
Además podemos (y deberíamos) asignar un nombre a esta tarea:



Actividad: Pedir a los alumnos que crean un procedimiento que esta vez dibuje un cuadrado de 200 por lado.



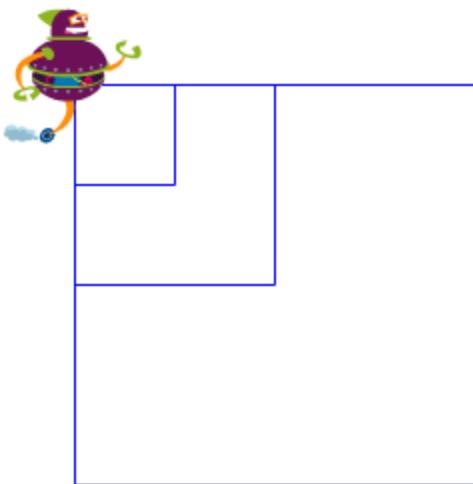
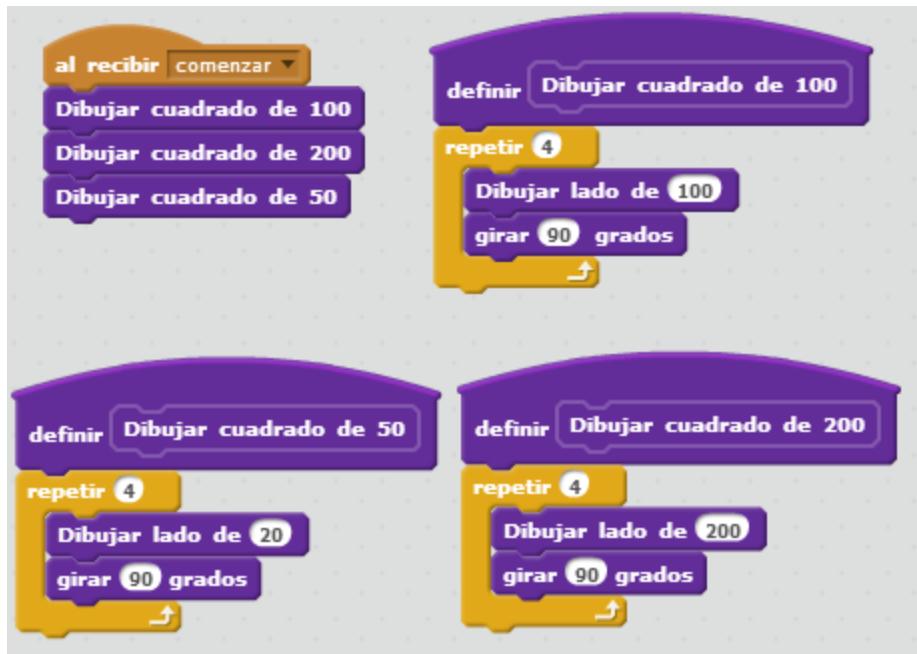
Podemos ver que el resultado es un cuadrado dentro de otro. El cuadrado más chico es el de 100 y el más grande es el de 200.



Los alumnos deberían darse cuenta que hicieron lo mismo sólo que cambiaron un 100 por un 200.

Actividad: Entre todos definir un procedimiento que ahora dibuje un cuadrado de 50 por lado.

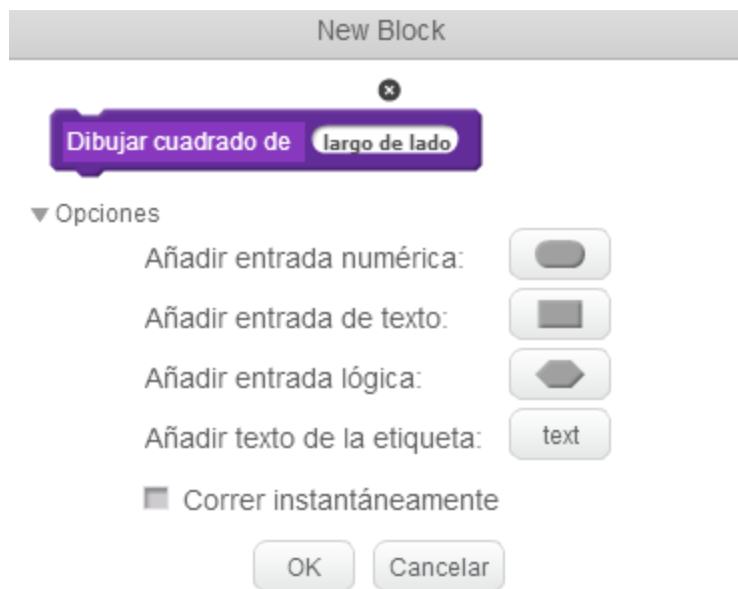
Lo mismo cuando pedimos que piensen uno de 50:



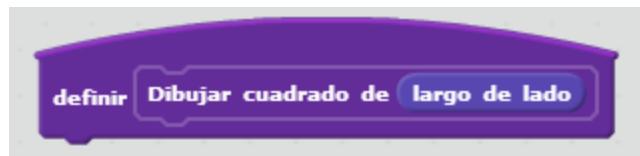
Si estamos haciendo repetidamente la misma acción, ¿habrá alguna forma mejor de hacer esto? ¿Qué nos convendría tener?

Por ejemplo, el comando girar nos permite indicarle cuánto girar **girar (1) grados**, el comando mover **Dibujar lado de (1)** permite indicar de cuánto dibujar un lado, etc. Queremos una solución similar, que nos permita elegir de qué tamaño queremos para nuestro cuadrado.

Para hacer esto crearemos un procedimiento, que contendrá un **parámetro** llamado “largo del lado”:



Esto nos creará un bloque como el siguiente:



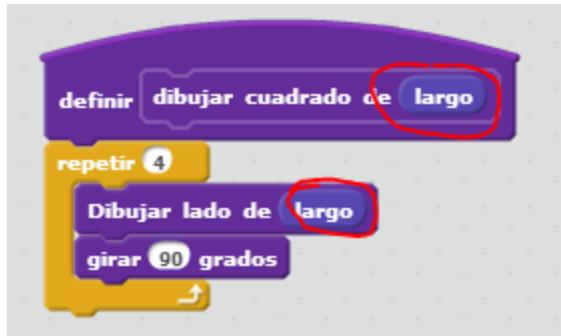
Y uno así para usar:



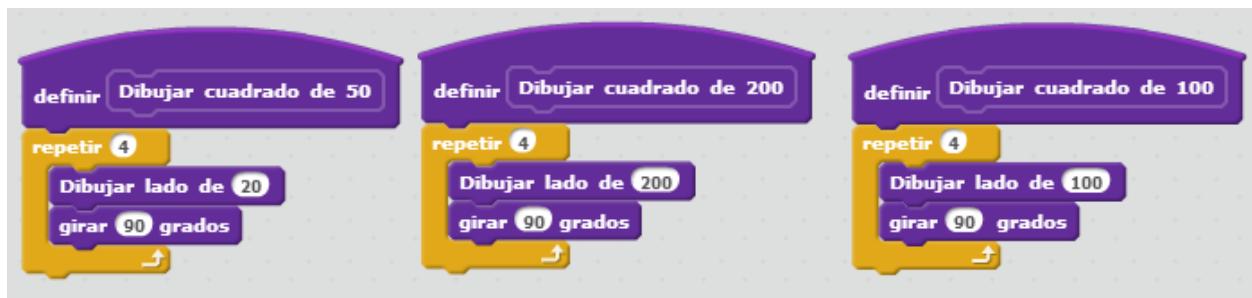
Ese hueco que aparece para elegir un dato, es el mismo que aparece en los otros bloques que estuvimos usando:



Lo que haremos ahora es definir cómo dibujar un cuadrado, pero en lugar de indicar la longitud del lado, arrastramos el bloque que hace de parámetro en vez de elegir un valor fijo:



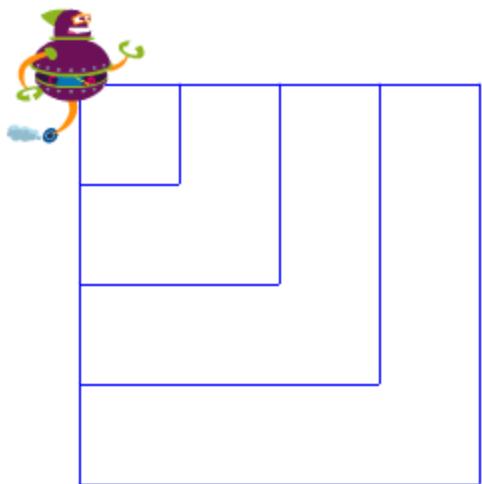
Los procedimientos anteriores ahora están de más y simplemente los borramos:



Actividad: dibujar cuadrados de los tamaños 50, 100, 150 y 200 usando el nuevo procedimiento parametrizado.

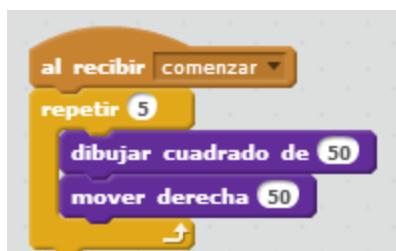
El resultado es el siguiente:



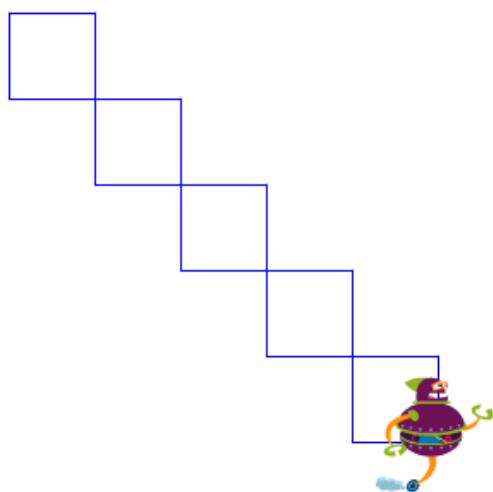


Es importante notar que utilizamos un sólo procedimiento para hacer cuatro cosas similares (dibujar cuadrados) pero que se diferencian en algo (el tamaño de cada cuadrado). Agregar un parámetro nos permitió justamente representar esta diferencia y trasladarla al usuario del bloque, que es el que decidirá de qué tamaño quiere dibujar cada cuadrado.

Actividad: Sobre la misma actividad de generar un cuadrado, se nos pide hacer 5 cuadrados de 50 cada uno, uno al lado del otro:



Actividad: Modificar el ejercicio anterior para que los cuadrados se dibujen en diagonal:



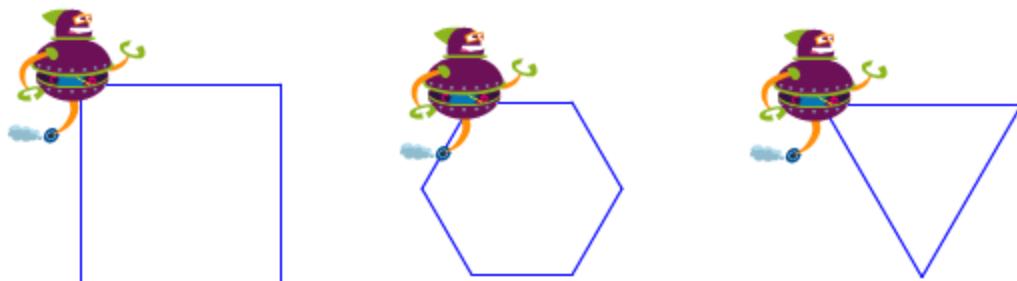
Solución:



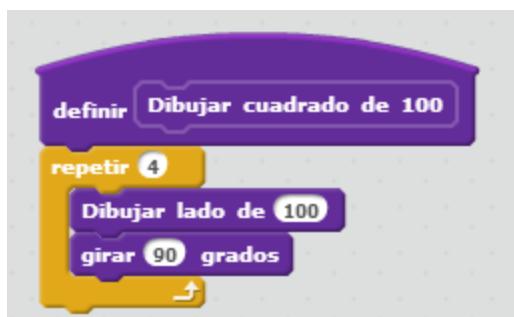
Más figuras

Se realiza sobre la misma actividad anterior.

En esta actividad se nos piden distintas figuras:



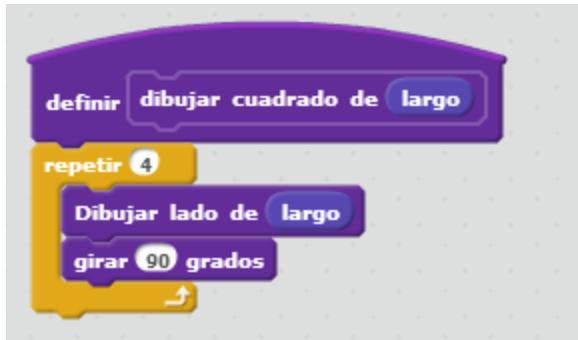
Repasemos cómo haríamos un cuadrado de 100 en Scratch:



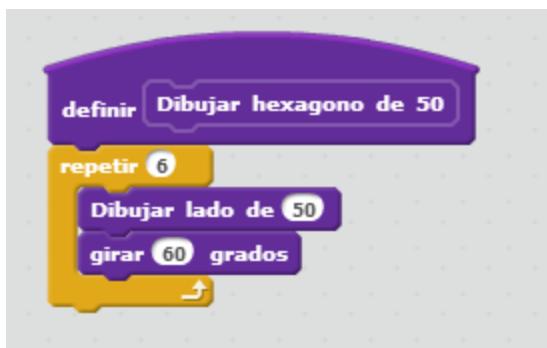
Analicemos cierta información::

- La suma de la extensión de todos los lados es el perímetro del cuadrado. En este caso es 400, porque es 4×100 (4 veces 100).
- Independientemente del tamaño del cuadrado, la suma de todos sus lados da 360, porque tenemos 4 giros, cada uno de 90 grados.

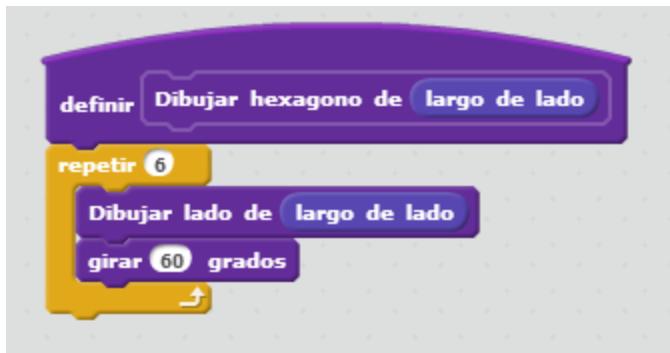
La mejora que hicimos a ese procedimiento es agregar un parámetro en lugar de decirle de cuánto es cada lado:



Para ver esto, probar el siguiente procedimiento:



Actividad: Reproducir este código en el editor de Scratch y parametrizar el largo de cada lado del hexágono.



Actividad: Responder entre todos, ¿cuántos lados tiene esta figura? ¿Cómo calculamos el perímetro? ¿Cuánto gira por cada lado? ¿Cuánto da la suma de todos sus giros?

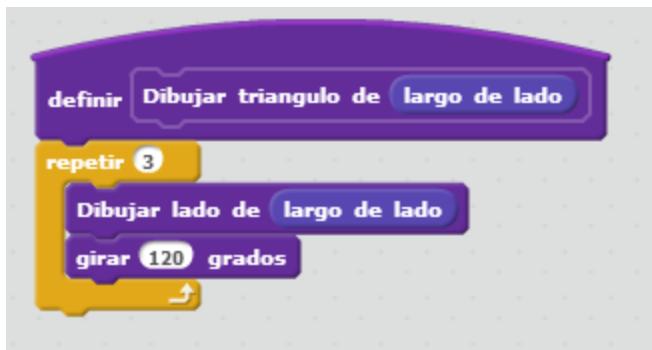
Para conocer el perímetro simplemente hacemos 6×50 , porque tenemos un lado de 50 repetido 6 veces. Pero lo curioso es que si hacemos 60×6 , nos vuelve a dar 360, como con el cuadrado. Esto es así porque este tipo de figuras siempre suma 360 grados en giros. No

importa si tiene 3, 4, 6 u 8 lados, la suma de todos los giros debe dar 360.

Si se quiere una figura de cierta cantidad de lados, sólo debemos hacer $360 / \text{cantidad de lados}$, para saber cuánto debemos girar por cada lado. Por ejemplo:

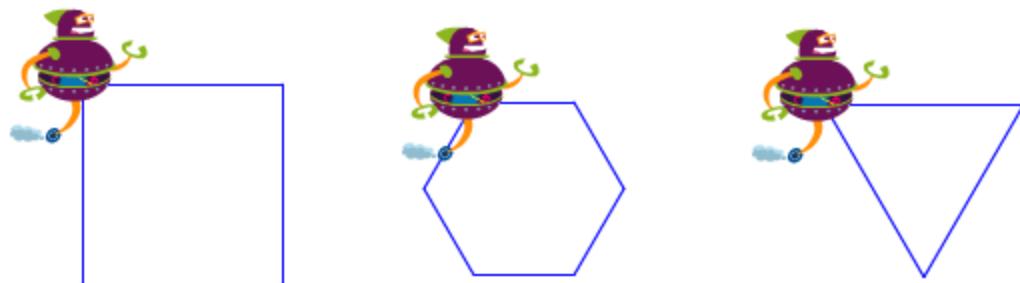
- El cuadrado tiene 4 lados, $360 / 4$ da 90.
- El hexágono tiene 6 lados, $360 / 6$ da 60.

Actividad: Sabiendo que todas las figuras deben sumar 360 grados en giros, ¿cómo haríamos una figura de 10 lados? ¿Y una figura de 8 lados? ¿Y una figura de 3 lados, o sea, un triángulo? Definir cada uno de estos procedimientos parametrizando el largo de cada lado, como hicimos en la actividad anterior.



Figuras regulares (opcional)

Tanto el robot como el programador están cansados de hacer tareas tan repetitivas. Los jefes se la pasan pidiendo tal o cual figura, todas con el mismo patrón: los lados y los ángulos se repiten cierta cantidad de veces. El problema es que siempre hay que estar haciendo la misma cuenta con los ángulos.



Por ejemplo, si nos piden hacer una figura de 4 lados, sabemos que vamos a girar 90 grados por cada uno. Si nos piden hacer una figura de 6 lados, vamos a girar 60 grados por cada uno

En otras palabras, siempre estamos haciendo la misma cuenta:

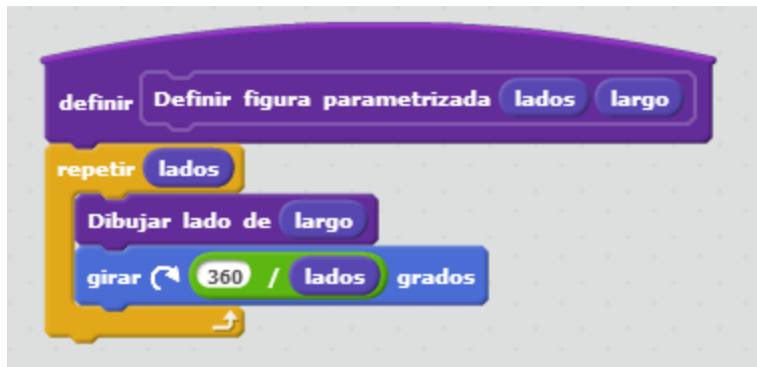
$$360 / 4 = 90$$

$$360 / 6 = 60$$

$$360 / \text{cantidad de lados} = \text{cantidad de giro por lado}$$

Además sería interesante poder elegir la extensión que tiene cada lado.

Esto se podría ver reflejado en un procedimiento que dibuje cualquier polígono regular:



Actividad: reproducir esto en Scratch. Notar que se utiliza la expresión dividir (categoría de operadores).

La fiesta de Drácula



Dracula organizó una fiesta con los fantasmas, brujas y monstruos de su ciudad, pero tiene un problema. Los foquitos del pasillo en donde comenzó la fiesta no tienen el color que le gustaría. La fiesta no va a comenzar si primero no se arregla esta situación.

Las primitivas son las siguientes:

Hay 3 focos. Para cambiar cada foco hasta conseguir el color esperado, debemos ejecutar la primitiva “cambiar color” una cierta cantidad de veces. Con el primer foco debemos ejecutar “cambiar color” 5 veces, con el segundo debemos ejecutarla 8 veces y con el tercero 12 veces. El resultado será el siguiente:

Luego de haber cambiado correctamente los focos, podemos ejecutar “empezar fiesta” para que Dracula y sus amigos comiencen a bailar. Sin embargo, si los focos no tienen el color correcto, la fiesta no empezará.

Actividad: definir un procedimiento con un parámetro que indique la cantidad de veces que ejecuta la instrucción primitiva “cambiar color”. Utilizar este procedimiento para cambiar el color de cada foco y resolver el problema.

La solución es la siguiente:

Salvando la navidad

Papá Noel tiene que recuperar los regalos que cayeron de su saco.

Las filas son siempre las mismas.

Actividad: Resolver este problema definiendo un único procedimiento que nos permita comer cualquier fila, pero sin utilizar una repetición condicional.

Para poder definir ese procedimiento debemos parametrizarlo para que podamos proveerle el largo de una fila determinada. La solución es la siguiente:

Prendiendo las compus parametrizado

Una primera observación interesante es que los procedimientos creados en la solución anterior son prácticamente iguales salvo por la dirección del movimiento. Vale la pena preguntarle a los alumnos qué podemos usar cuando distintos procedimientos necesarios son similares salvo por una diferencia mínima.

Una forma de resolver la actividad anterior consiste en definir un procedimiento que reciba por parámetro una dirección y se mueva hacia esa dirección:

Lo que resta ahora es preguntar qué dirección recibimos y relacionarla con cada comando particular:

Ahora en lugar de tener estos 4 procedimientos

Podemos hacer uno solo que reciba también una dirección:

Y nuestro programa queda tan simple como

Actividad: guiar a los alumnos hacia esta solución, y que todos la reproduzcan en sus computadoras.

Lightbot cuadrado

Esta actividad es similar a la de prender un cuadrado de computadoras. Las únicas diferencias radican en que ahora el cuadrado es de tamaño fijo y en un lado determinado no sabes qué celdas poseen luces y cuáles no. Sólo las esquinas nunca poseen luces.

Comenzaremos definiendo la subtarea “Mover hacia”, idéntica a la actividad anterior.

Una vez terminada, debemos definir una subtarea que nos permita procesar cualquier lado del cuadrado:

Actividad: Completar la definición de las subtareas mencionadas y luego definir un programa que utilizando sólo la última subtarea pueda prender todas las luces del cuadrado.

La solución es la siguiente:

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

Mario bros

Mario tiene que recuperar todas las monedas. Utilizaremos una técnica similar a la actividad anterior, en el que definiremos una subtarea que nos permita agarrar monedas hacia una dirección, y que además nos permita indicar cuántas.

Se cuenta con las siguientes primitivas:

Y queremos definir una subtarea así:

Además esa subtarea utilizará a esta otra, que permitirá movernos un único paso hacia la dirección que queramos:

Actividad: Completar las definiciones de las subtareas indicadas y resolver la actividad utilizándolas donde corresponda.

Una solución es la siguiente. Para la subtarea “Mover hacia”, debemos dar esta definición:

Luego la subtarea “Agarrar *cantidad* monedas hacia *direccion*” podemos definirla así:

Y con esto el programa que podemos definir es el siguiente (agregando una subtarea más para comer las monedas del centro):

Interactividad

La interactividad es un elemento indispensable cuando hablamos de juegos, ya que es lo que nos permite que los programas realicen acciones en base a lo que el usuario presione en el teclado, el mouse, un joystick o cualquier dispositivo que permita ingresar datos a la computadora.

Esta sección se divide en dos partes. Primero haremos dos actividades introductorias que no poseen interactividad, pero que nos permitirán introducir elementos de la lógica de los juegos que vamos a programar. Luego programaremos partes de algunos juegos conocidos por los chicos, y otros que diseñamos nosotros.

La música del loro

Tenemos dos escenarios y un mismo problema. Necesitamos hacer que el loro se acerque a un determinado instrumento y realice una acción con este. Pero sucede que en el escenario a veces hay un micrófono, y a veces un tambor. Si estamos ante la presencia de un micrófono, debemos cantar, y si es un tambor debemos tocarlo.

El otro problema es que tampoco sabemos en qué posición aparece el instrumento.

Contamos con las siguientes primitivas:

Actividad: Permitir que los chicos intentar resolver el problema.

Como no sabemos a qué distancia se encuentra el instrumento vamos a terminar introduciendo un “repetir hasta que”:

Pero el problema es que no podemos fijar como condición que vamos a tocar un “tambor” o un “micrófono”, porque en el escenario puede aparecer cualquiera de estos. Lo que

precisamos es combinar ambas condiciones, ya que la respuesta es que debe frenar cuando se encuentra un tambor o cuando se encuentre un micrófono. Esto lo hacemos con el operador “o” de la categoría de operadores:

Con esto ya podemos hacer que el loro avance hasta el objetivo, sea cual sea:

Lo que deberíamos hacer ahora es que el loro cante o toque el tambor. ¿Pero por cuál de las dos condiciones se detuvo? Tampoco lo sabemos. ¿Cómo debemos solucionarlo?

Actividad: Permitir a los chicos intentar resolver lo que resta del problema.

La solución consiste en preguntar si estamos tocando un tambor. Si esto es verdad, simplemente tocamos el tambor, y sino, podemos cantar, porque tiene que ser un micrófono:

Debemos notar que cuando nos detuvimos en el “repetir hasta que”, nos tuvimos que haber detenido porque alguna de las dos condiciones se cumplía, sino hubiésemos seguido avanzando. Y como esto es verdad, luego podemos preguntar si nos detuvimos por haber tocado un tambor, o un micrófono, porque no puede ser otro el caso. Simplemente debemos preguntar esto porque no sabemos por cuál de las dos condiciones nos hemos detenido.

Esta técnica nos servirá más adelante para modelar la lógica de ciertos juegos.

Homero Simpson

Homero tiene hambre, y piensa en comer todas las hamburguesas y rosquillas que tiene por delante. El problema es que en cada fila no sabemos si va a tener una rosquilla o una hamburguesa.

Las primitivas son:

Actividad: Definir un programa que resuelva este problema, utilizando recordando la técnica vista en la actividad anterior.

La solución es la siguiente:

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

El juego más difícil del mundo

Nuestro primer juego está basado en otro del mismo nombre. Lo pueden encontrar en www.el-juego-mas-dificil-del-mundo.com.

El juego es simple. Manejamos un cuadrado rojo, y debemos llegar a la salida (el cuadrado verde), esquivando las bolas azules.

Si en Scratch presionamos la banderita verde vamos a ver que los obstáculos (las bolas azules) se mueven de un lado a otro, pero no podemos mover al personaje. Todas las actividades en las que definamos juegos consistirán en que los alumnos programen dicha lógica.

Las primitivas que trae este juego son las siguientes:

Cada juego tendrá una lógica diferente. En este en particular sucede que:

- Si tocamos la salida ganamos
- Si tocamos una bola azul perdemos

Debemos pensar que mientras no hayamos ganado o perdido, podemos continuar jugando. Eso significa que la técnica que vimos en actividades anteriores puede utilizarse:

¿Qué sucede en este juego mientras no hayamos ganado o perdido? Podemos movernos hacia algunas de las posibles direcciones. Esto lo haremos preguntando por una condición que no utilizamos hasta ahora, llamada “¿tecla ... presionada?”, de la categoría sensores:

Esta condición nos permite preguntar si se ha presionado una tecla en particular.

Por ejemplo, podemos relacionar esta condición con movernos hacia abajo:

Y si esto lo ponemos dentro de la repetición obtenemos lo siguiente:

Actividad: definir una subtarea que permita moverse hacia alguna de las direcciones posibles en base a la tecla presionada.

La solución es la siguiente:

Ya podemos jugar un poco, porque si tocamos una bola o la salida el juego el programa termina (termina la repetición y no hay más comandos en la secuencia).

¿Qué podemos hacer luego de la repetición? Lo mismo que hicimos en actividades anteriores, preguntar si lo que pasó es que perdimos, o si pasó que ganamos.

Y con esto hemos completado el juego en su totalidad.

En esta instancia podemos invitar a los chicos a cambiar los disfraces de los objetos del juego, para que noten que podemos transformar este juego en otro, sólo cambiando los dibujos y manteniendo la lógica. Eso lo podemos hacer seleccionando cada objeto del juego y yendo a “disfraces”:

Por ejemplo, podemos hacer que el personaje sea una bruja, los obstáculos murciélagos y el objetivo una varita que hay que recuperar:

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

En búsqueda de la llave

En este juego debemos hacer que el buzo agarre la llave, esquivando al tiburón.

La lógica es simple:

- Avanzamos al presionar alguna tecla mientras no hayamos ganado o perdido.
- Perdemos si el tiburón nos toca
- Ganamos si tocamos la llave

Actividad: programar este juego utilizando las técnicas vistas en la actividad anterior:

La solución es la siguiente:

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

Pacman

En esta actividad programaremos una versión simplificada del viejo y conocido Pacman.

Contamos con las siguientes primitivas:

Si comenzamos el juego podremos notar que sucede lo mismo que en la actividad anterior:

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

nuestro personaje no se mueve y no podemos ganar ni perder.

En este caso, podremos ganar si sumamos 20 puntos (este dato se encuentra en la categoría datos). Si nuestros puntos son iguales a 20, ganamos el juego. Y como contrapartida, podemos perder si tocamos un fantasma.

Mientras no hayamos ganado o perdido, podremos movernos y comer puntos. El Pacman se mueve distinto al juego anterior: avanza todo el tiempo (sin necesidad de presionar una tecla) y podemos hacer que apunte hacia distintas direcciones para que avance hacia otro lado. Además mientras nos movemos su disfraz va cambiando, y eso lo hacemos con la acción “siguiente disfraz”. Si en el camino se topa con un punto, simplemente lo come. En resumen el movimiento se define como:

1. Avanzar
2. Apuntar hacia la dirección que indique el jugador con el teclado
3. Pasar al siguiente disfraz.
4. Comer punto si chocamos uno.

Si llegamos a ganar o perder, la repetición principal del juego se detendrá, y como en la actividad anterior, debemos indicar qué pasa en cada caso. Para el caso de perder ya existe una primitiva que define qué pasa, pero para el caso de ganar, invitaremos a los alumnos a definir lo que quieran (puede decir una simple frase, o hacer algo más complicado).

Actividad: definir un programa con la lógica del juego siguiendo todas las reglas que describe el enunciado anterior.

Una solución es la siguiente:

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

La defensa de la Tierra

En este juego manejaremos un tanque que debe defender a la Tierra de una invasión alienígena. El juego consiste en destruir todas las naves enemigas que podamos, pudiendo recibir como máximo hasta 5 disparos, en cuyo caso perdemos.

Las primitivas son las siguientes:

Debemos programar la siguiente lógica del juego:

- Mientras no hayamos perdido podemos movernos y disparar. Además si en el transcurso recibimos un disparo debemos restarnos una vida.
- Perdemos cuando la cantidad de vida sea 0. Este dato lo podemos conseguir en la

Este cuadernillo es una versión preliminar del material definitivo. Por consultas, escribir a
pfactorovich@fundacionsadosky.org.ar

categoría “datos”.

Actividad: Programar el juego en base a la lógica descrita anteriormente.

Una solución es la siguiente: